APLIKASI *WEB SERVICE* PADA *E-COMMERCE* DENGAN MENGGUNAKAN SOAP DAN WSDL

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Teknik

Disusun oleh :

FIRMAN FAIZAL RAHMAN

NIM. 0310630050

DEPARTEMEN PENDIDIKAN NASIONAL

UNIVERSITAS BRAWIJAYA

FAKULTAS TEKNIK

JURUSAN TEKNIK ELEKTRO

MALANG

2009

APLIKASI *WEB SERVICE* PADA *E-COMMERCE* DENGAN MENGGUNAKAN SOAP DAN WSDL

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Teknik

Disusun oleh :

FIRMAN FAIZAL RAHMAN

NIM. 0310630050

Telah diperiksa dan disetujui oleh
Dosen Pembimbing :

Raden Arief Setyawan, ST., MT.	**Himawat Aryadita, ST., M.Sc.**
NIP. 132 231 713	**NIP. 132 327 625**

APLIKASI *WEB SERVICE* PADA *E-COMMERCE* DENGAN MENGGUNAKAN SOAP DAN WSDL

Disusun oleh :

FIRMAN FAIZAL RAHMAN

NIM. 0310630050

Skripsi ini telah diuji dan dinyatakan lulus pada
tanggal 7 Agustus 2009

Majelis Penguji :

Ir. Heru Nurwasito, M.Kom.	**Ir. Sutrisno, MT.**
NIP. 131 879 033	**NIP. 131 653 479**

Arief Andy Soebroto, ST., M.Kom.

NIP. 132 231 567

Mengetahui :

Ketua Jurusan Teknik Elektro

Ir. Heru Nurwasito, M.Kom.

NIP. 131 879 033

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ

Atas rahmat Allah Yang Maha Penyayang , tugas akhir ini kupersembahkan untuk Ayah dan Ibu tercinta serta saudaraku atas doa dan kasih sayangnya. Terima kasih untuk teman-teman yang memberikan inspirasi kehidupan dan orang-orang di sekelilingku yang kusayang dan menyayangiku.

KATA PENGANTAR

Puji Syukur kehadirat Allah SWT karena dengan berkat rahmat dan karunia serta ridlo-Nya penyusunan skripsi ini dengan judul "Aplikasi *Web Service* pada *E-Commerce* dengan Menggunakan SOAP dan WSDL" dapat diselesaikan. Penulis menyadari bahwa kajian ini tidak akan mencapai titik akhir penyelesaian tanpa bantuan berbagai pihak, karenanya penulis mengucapkan terima kasih kepada :

1. Keluarga besar Bapak Drs. H. Abdul Bari, BBA. (alm) untuk seluruh do'a dan dukungannya yang telah diberikan kepada Ananda selama studi hingga terselesaikannya skripsi ini.
2. Ir. Heru Nurwarsito, M.Kom. selaku Ketua Jurusan Teknik Elektro, Fakultas Teknik, Universitas Brawijaya.
3. Rudy Yuwono, ST, MSc. selaku Sekretaris Jurusan Teknik Elektro, Fakultas Teknik, Universitas Brawijaya.
4. Raden Arief Setyawan, ST., MT. selaku dosen pembimbing pada penyusunan skripsi ini.
5. Himawat Aryadita, ST., M.Sc. selaku dosen pembimbing pada penyusunan skripsi ini.
6. Bapak dan Ibu dosen serta karyawan Jurusan Teknik Elektro, Fakultas Teknik, Universitas Brawijaya.
7. Astika Citra Parnawati dan keluarga. Terima kasih atas segala perhatian, doa, kasih sayang, serta dukungan semangatnya.
8. Teman, sahabat, dan saudaraku yang telah bersama menjalani hari dengan suka dan duka. Terima kasih atas pelajaran hidup yang sangat berharga, semua kenangan tidak akan bisa terlupakan.
9. Serta semua pihak yang tak dapat disebutkan satu persatu yang telah turut membantu baik secara langsung maupun tidak langsung dalam penyelesaian skripsi ini.

Tiada gading yang tak retak, tersadar bahwa skripsi ini sangat jauh dari kesempurnaan. Karenanya, segala kritik dan saran yang sifatnya membangun dari pembaca tentang isi skripsi ini akan diterima dengan senang hati. Akhir kata, penulis berharap, semoga skripsi ini dapat bermanfaat bagi semua pihak yang membutuhkan.

Malang, Juli 2009,

Penyusun

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR ALGORITMA

DAFTAR ISTILAH

Actor	Kumpulan peran yang saling berkaitan dimana pengguna dari *use case* bermain ketika berinteraksi dengan *use case* [BOR-05].
Class	Sebuah deskripsi dari kumpulan obyek yang berbagi atribut, operasi dan relasi yang sama [BOR-05].
Object	Manifestasi konkrit dari sebuah abstraksi, sebuah entitas dengan ikatan dan identitas yang tergambar dengan baik yang mengenkapsulasi keadaan dan tingkah laku, sebuah *instance* dari sebuah kelas [BOR-05].
UML	*Unified Modeling Language,* sebuah bahasa grafis untuk visualisasi, spesifikasi, konstruksi, dan dokumentasi artifak-artifak dari sistem perangkat lunak [BOR-05].
Use Case	Sebuah deskripsi dari kumpulan aksi yang berurutan termasuk varian, yang dijalankan oleh sebuah sistem yang menghasilkan sebuah hasil berupa nilai yang dapat diamati pada seorang aktor [BOR-05].
Web Service	Sebuah sistem *software* yang dirancang untuk mendukung interoperabilitas interaksi antara mesin ke mesin pada sebuah *network*.
E-Commerce	Cara berbelanja atau berdagang secara *online* atau *direct selling* yang memanfaatkan fasilitas internet dimana terdapat *website* yang dapat menyediakan layanan "*get and deliver*"

ABSTRAK

FIRMAN FAIZAL RAHMAN. 2009. : Aplikasi *Web Service* pada *E-Commerce* dengan Menggunakan SOAP dan WSDL. Skripsi Jurusan Teknik Elektro, Fakultas Teknik, Universitas Brawijaya. Dosen Pembimbing : Raden Arief Setyawan, ST., MT. dan Himawat Aryadita, ST., M.Sc.

Perdagangan merupakan kegiatan yang dilakukan manusia sejak awal peradabannya. Sejalan dengan perkembangannya, cara dan sarana yang digunakan untuk berdagang senantiasa berubah dengan tuntutan untuk memberikan kemudahan bagi konsumen. *E-Commerce* merupakan media perdagangan *online* yang banyak digunakan oleh masyarakat karena memiliki beberapa kelebihan dalam penggunaanya. Hal tersebut mengakibatkan semakin banyaknya yang menggunakan *E-Commerce*. Sehingga memberikan tantangan bagi masyarakat dalam memperoleh informasi. Dengan adanya tantangan itulah, dibutuhkan suatu teknologi yang dapat mengintegrasikan potongan – potongan informasi dari berbagai *E-Commerce* hanya dengan mengakses sebuah situs *E-Commerce* saja. Teknologi tersebut adalah *Web Service*, dimana *web service* menawarkan kemudahan untuk menjembatani berbagai *E-Commecrce* tersebut tanpa mempermasalahkan perbedaan teknologi dan *platform* yang digunakan oleh masing – masing sumber.

Perancangan dan pengimplementasian Aplikasi *Web Service E-Commerce* dilakukan dengan menggunakan bahasa pemrograman XML *Web Service* (SOAP dan WSDL), PHP, dan basis data MySQL. Aplikasi *Web Service E-Commerce* terbagi menjadi dua, yakni Aplikasi *Web Service* dan *Website E-Commerce*. Aplikasi *Web Service* terdiri atas SOAP *server*, SOAP *client*, dan WSDL. Sedangkan *website E-Commerce* terdiri atas *Website* Utama *E-Commerce* dan *Website* Administrasi *E-Commerce*.

Pengujian Aplikasi *Web Service E-Commerce* yang dilakukan dengan menggunakan metode pengujian unit, pengujian integrasi, dan pengujian validasi. Dari metode pengujian tersebut menunjukkan bahwa aplikasi *Web Service E-Commerce* telah berhasil diuji tanpa adanya kesalahan sama sekali. Pengujian Aplikasi *Web Service E-Commerce* juga dilakukan dengan menguji waktu akses *query* basis data dan performansi *web server* Apache HTTP. Hasil dari pengujian waktu akses *query* basis data menunjukkan bahwa basis data sistem memiliki waktu akses *query* terlama pada 0.012 ms dengan 1000 data *entry*. Sedangakan untuk hasil performansi *web server* Apache HTTP menunjukkan bahwa performansi terbaik *web server* terjadi pada 700 koneksi dengan 540 *request* yang dilakukan secara bersamaan.

Kata Kunci : *Web Service, E-Commerce*, XML, SOAP, WSDL.

BAB I
PENDAHULUAN

1.1. Latar Belakang

Perdagangan merupakan kegiatan yang dilakukan manusia sejak awal peradabannya. Sejalan dengan perkembangan manusia, cara dan sarana yang digunakan untuk berdagang senantiasa berubah dengan tuntutan untuk memberikan kemudahan bagi konsumen.

Dalam beberapa waktu terakhir ini, dengan begitu merebaknya media internet, khususnya di Indonesia, banyak perusahaan mulai mencoba menawarkan berbagai macam produk mereka dengan menggunakan internet sebagai media perdagangan. Dan masyarakat sebagai konsumen cenderung memilih cara cepat, mudah, dan murah untuk bertransaksi baik dalam jual beli maupun transaksi lainnya. Sehingga *E-Commerce* menjadi solusi bagi perusahaan untuk pencapaian tujuan finansial melalui modifikasi dan efisiensi proses bisnisnya.

Semakin banyaknya perusahaan yang menggunakan internet sebagai media bisnis dan perdagangan, semakin banyak pula *E-Commerce* yang sejenis beredar di internet. Hal ini memberikan tantangan bagi masyarakat sebagai konsumen dimana dalam mencari informasi yang mereka inginkan harus berlompatan dan menjelajah dari *E-Commerce* satu ke *E-Commerce* yang lain yang kadang kala menjadi pekerjaan yang cukup merepotkan. Sehingga harus membuka beberapa *browser* untuk mengakses masing - masing *E-Commerce* tersebut.

Dengan adanya tantangan itulah, dibutuhkan suatu teknologi yang dapat mengumpulkan informasi dari beberapa *E-Commerce* tersebut hanya dengan menggunakan satu pintu, atau suatu teknologi yang dapat mengintegrasikan potongan – potongan informasi dari berbagai *E-Commerce* hanya dengan mengakses sebuah situs. Teknologi yang dapat menjadi solusi dari tantangan tersebut adalah teknologi *Web Service*, dimana teknologi *web service* menawarkan kemudahan untuk menjembatani berbagai *E-Commecrce* tersebut tanpa mempermasalahkan perbedaan teknologi dan platform yang digunakan oleh masing – masing sumber. Sehingga *web service* dapat diibaratkan sebagai pustaka

yang dapat menyediakan berbagai macam informasi dan layanan ke aplikasi lain antar *web*.

1.2. Rumusan Masalah

Berdasarkan pada permasalahan yang telah dijelaskan pada bagian latar belakang, maka rumusan masalah dikhususkan pada :

1. Merancang dan membuat sebuah aplikasi Web Service E-Commerce dengan menggunakan bahasa pemrograman PHP dan XML Web Service. Modul - modul aplikasi web service yang digunakan berupa pencarian data, katalog data, dan review data yang mendukung aplikasi E-Commerce.
2. Merancang dan membuat suatu basis data Web Service E-Commerce dengan menggunakan basis data MySQL yang dapat digunakan pada aplikasi.
3. Mengimplementasikan dan menguji sistem yang telah dirancang.

1.3. Batasan Masalah

Ruang lingkup penulisan dibatasi pada :

1. Konsep dasar pemrograman PHP dan XML *Web Service* (SOAP dan WSDL) dengan basis data MySQL.
2. Perangkat lunak aplikasi dibuat dengan menggunakan Sistem Operasi Windows XP Professional, Web Server Apache, PHP, dan MySQL.
3. Pembahasan hanya difokuskan pada penerapan *web service*.

1.4. Tujuan Penulisan

Tujuan akhir dari penelitian ini adalah untuk membuat dan merancang aplikasi *Web Service E-Commerce* dengan menggunakan bahasa pemrograman PHP, XML *Web Service*, dan basis data MySQL.

1.5. Manfaat

Manfaat penelitian ini antara lain:

1. Bagi penulis :
   - Menerapkan ilmu yang telah diperoleh di Teknik Elektro konsentrasi sistem informatika dan komputer Universitas Brawijaya.

- Mengembangkan aplikasi *E-Commerce* yang memiliki teknologi *web service* secara tepat sehingga mendapatkan hasil yang maksimal.

2. Bagi pengguna :
   - Mempercepat dan mempermudah proses administrasi basis data dalam jaringan internet secara terpusat.
   - Mempercepat dan mempermudah pengguna dalam memperoleh data dan informasi yang dibutuhkan.

1.6. Sistematika Penulisan

Sistematika penulisan dalam skripsi ini sebagai berikut:

BAB I Pendahuluan

Memuat latar belakang, rumusan masalah, tujuan, batasan masalah, metodologi pembahasan, dan sistematika pembahasan.

BAB II Dasar Teori

Membahas teori-teori yang mendukung dalam perancangan dan pembuatan aplikasi.

BAB III Metodologi

Berisi tentang metode penelitian yang digunakan dalam perancangan dan pengujian sistem.

BAB IV Analisis Kebutuhan dan Perancangan

Membahas tentang analisa kebutuhan dari sistem dan kemudian merancang hal-hal yang berhubungan dengan analisa tersebut.

BAB V Implementasi

Bagian ini berisi penjelasan tentang implementasi yang telah dilakukan (sistem operasi, perangkat keras dan perangkat lunak) dan batasan-batasan implementasi.

BAB VI Pengujian dan Analisis

Bagian ini berisi penjelasan tentang strategi pengujian (unit, integrasi dan validasi) dan teknik pengujian yang dilakukan. Dijelaskan juga seluruh kasus uji beserta hasil pengujiannya.

BAB VII Penutup

Bagian ini berisi kesimpulan dan saran. Kesimpulan didasarkan

atas pengujian dan analisis yang dilakukan di dalam proses penelitian.

BAB II
DASAR TEORI

Bab ini membahas tinjauan pustaka tentang Aplikasi *Web Service* Pada *E-Commerce*, dan dasar teori yang digunakan untuk menunjang penulisan skripsi mengenai pengembangan Aplikasi *Web Service* Pada *E-Commerce*.

2.1. Tinjauan Pustaka

Web Service menyediakan standar komunikasi di antara berbagai aplikasi *software* yang berbeda – beda. Dimana dapat berjalan di berbagai *platform* maupun *framework* yang berbeda.

Pada *web service* terdapat istilah *providers* dan *requesters*. Dimana *providers entity* adalah seseorang atau organisasi yang menyediakan agen untuk layanan fungsi tertentu. Sedangkan *requesters entity* adalah seseorang atau organisasi yang menggunakan layanan yang disediakan oleh *providers entity*. Agar dapat berkomunikasi dengan lancar, kedua entitas ini harus sepakat menggunakan semantik dan mekanisme yang sama dalam pertukaran pesan.

Mekanisme pertukaran pesan didokumentasikan di *Web Service Description* (WSD). WSD adalah spesifikasi mesin yang dapat berproses pada *interface web service*. Dimana WSD ini dapat mendefinisikan format pesan, tipe data, protokol transpor, dan format serialisasi yang digunakan antara agen *requester* dan agen di *providers*. Selain itu, WSD juga menentukan satu atau lebih *network* tempat agen provider akan dipanggil. Secara umum, proses *web service* dapat digambarkan sebagai berikut [SIS-04] :

1. Entitas peminta (*requester*) dan penyedia (*provider*) saling mengetahui satu sama lain. Atau, setidaknya salah satu mengetahui yang lain.
2. Entitas peminta dan penyedia sama – sama setuju deskripsi layanan dan semantik yang akan mengatur interaksi antar agen peminta dan penyedia.
3. Deskripsi layanan semantik berdasarkan agen peminta dan penyedia.
4. Agen peminta dan penyedia bertukar pesan.

Teknologi *Web Service* ini menjadi solusi bagi perkembangan internet. Dimana pertukaran data dan pertukaran *service* antar *website* menjadi marak

dalam integrasi antar aplikasi. Khususnya pada *E-Commerce* yang semakin banyak muncul di internet dewasa ini. Dimana beberapa proses dan aplikasi pada satu *E-Commerce* dapat digunakan dan diintegrasikan dengan *E-Commerce* lainnya.

Saat ini *E*-Commerce menjadi solusi bagi beberapa perusahaan yang bergerak di bidang perdagangan. Karena beberapa alasan dan banyaknya keuntungan yang didapat dengan menggunakan *E-Commerce*, banyak pula perusahaan/individu yang menggunkannya sebagai media bisnis dan perdagangan. Beberapa proses yang ada dalam *E-Commerce* adalah diantaranya sebagai berikut [AND-08] :

- Presentasi elektronik (pembuatan *website*) untuk produk dan layanan.
- Pemesanan secara langsung dan tersedianya tagihan.
- Otomasi *account* pelanggan secara aman (baik nomor rekening maupun nomor kartu kredit)
- Pembayaran yang dilakukan secara langsung (*online*) dan penanganan transaksi

Beberapa situs *E-Commerce* di luar negeri menggunakan sistem pembayaran *online*, seperti *PayPal*, *eBay*, dan sebagainya. Sebagian besar *costumer* yang sering melakukan belanja *online* menggunakan *payment gateway* tersebut. Akan tetapi, terdapat beberapa kendala dalam menggunakan pola *payment gateway* dalam proses pembayaran *online* di Indonesia.

Misalnya bagi pengguna *PayPal* di Indonesia baru dapat menggunakan fasilitas ini untuk pembayaran dan mengirim dana saja belum dapat menerima dana. Selain itu, para pengguna *PayPal* di Indonesia masih harus menggunakan dananya dari kartu kredit dalam hitungan *US dollar* karena rupiah belum tersedia di *PayPal*.

Kendala lainnya adalah budaya kartu kredit di Indonesia bisa dikatakan masih sangat muda. Sehingga banyak masyrakat yang tidak menggunakan kartu kredit sebagai alat pembayaran. Padahal sebagian besar transaksi *online* menggunakan kartu kredit.

Dengan adanya kendala tersebut, beberapa metode yang dilakukan dalam melakukan transaksi di *E-Commerce*, khususnya di Indonesia, yakni :

1. Transaksi model-ATM, yang menyangkut hanya institusi finansial dan pemegang *account* yang akan melakukan pengambilan atau mendeposit uangnya dari *account* masing-masing.
2. Pembayaran dua pihak tanpa perantara, transaksi dilakukan langsung antara dua pihak tanpa perantara menggunakan uang nasional-nya.
3. Pembayaran dengan perantaraan pihak ke tiga, umumnya proses pembayaran yang menyangkut debit, kredit maupun *check* masuk dalam kategori ini.
4. *Micropayment*, dalam bahasa sederhananya adalah pembayaran untuk uang recehan yang kecil-kecil. Mekanisme *Micropayment* ini penting dikembangkan karena sangat diperlukan pembayaran receh yang kecil tanpa *overhead* transaksi yang tinggi.
5. *Anonymous digital cash*, uang elektronik yang di enkripsi, di dahului oleh David Chaum dengan *Digicash*-nya. Uang elektronik menjamin *privacy* dari *user cash* tetap terjamin sama seperti uang kertas maupun *coin* yang kita kenal.

Yang sudah umum di *webstore-webstore* di Indonesia adalah nomor 1. Transaksi model-ATM, mungkin dapat dikatakan yang paling mudah dan *flexibel*, dengan banyaknya jaringan ATM yang sekarang ini ditambah dengan internet/*sms banking* sangat memberikan kenyamanan bagi kedua belah pihak. Uang yang masuk ke rekening mudah di cek, sehingga setelah itu barang dapat di *delivery* [RAH-08].

2.2. Dasar Teori

Teori dasar yang digunakan yaitu, teori dasar *web service*, *e-commerce*, basis data, pemrograman *web*, dan rekayasa perangkat lunak. Teori dasar basis data meliputi pengertian basis data, bahasa basis data SQL (*Structured Query Language*), basis data MySQL. Teori dasar pemrograman *web* meliputi HTML (*Hyper Text Markup Language*) dan PHP (*PHP Hypertext Preprocessor*). Teori dasar rekayasa perangkat lunak meliputi teori analisis kebutuhan, analisis berorientasi obyek, dan desain berorientasi obyek dengan menggunakan bahasa pemodelan UML (Unified Modelling Languange), dan pengujian berorientasi

obyek.

2.2.1. *Web Service*

Web Service adalah sebuah sistem *software* yang dirancang untuk mendukung interoperabilitas interaksi antara mesin ke mesin pada sebuah *network*. *Interface* dideskirpsikan pada format mesin seperti WSDL. Sistem lain yang berinteraksi dengan *Web Service* dilakukan melalui antar muka menggunakan pesan seperti pada SOAP. Pada umumnya pesan ini memanfaatkan XML dan HTTP yang merupakan salah satu standar *web*.

Perangkat lunak aplikasi yang ditulis dalam berbagai bahasa pemrograman dan berjalan pada berbagai *platform* dapat menggunakan *web service* untuk pertukaran data pada jaringan komputer seperti internet dalam cara yang serupa dengan komunikasi inter-*proccess* pada komputer tunggal. Interoperabilitas ini (sebagai contoh, antara Java dan Pyton, atau Microsoft Windows dan Aplikasi Linux) adalah dalam kaitan dengan penggunaan dari open standar.

Sehingga *web service* tersedia pada "intranets, ekstranet, dan internet" dimana *web service* tidak hanya menjadi publik, mereka dapat ada pada satu jaringan internal untuk aplikasi internal. *Web service* bisa digunakan antar mitra organisasi dalam solusi B2B yang kecil. Berikut ini beberapa keuntungan Web Service:

- *Web Service* menyediakan interoperabilitas antar berbagai aplikasi perangkat lunak yang *running* pada *platform* yang berbeda.
- *Web Service* menggunakan standar dan protokol yang *open*. Jika memungkinkan protokol dan format data adalah *text-based*, membuatnya mudah bagi pengembang untuk memahami.
- Dengan pemanfaatan HTTP, *Web Service* dapat bekerja melalui banyak pengukuran keamanan *firewall* yang umum tanpa menuntut perubahan bagi aturan *firewall filtering*.
- *Web Service* mengijinkan perangkat lunak dan *service* dari perusahaan dan lokasi yang berbeda untuk dikombinasikan dengan mudah untuk menyediakan suatu *service* yang terintegrasi.
- *Web Service* mengijinkan penggunaan kembali *service* dan komponen di

dalam suatu infrastruktur

- *Web Service* dapat secara bebas digabungkan(*loosely coupled*) dengan demikian memudahkan suatu pendekatan terdistribusi ke pengintegrasian aplikasi.

Sedangkan Kekurangan Web Service adalah:

- Karakteristik standar *Web Service* saat ini masih dalam tahap perkembangan awal dibandingkan *open standard* komputer terdistribusi yang lebih matang seperti CORBA. Ini nampaknya akan merupakan suatu kerugian yang temporer ketika kebanyakan vendor sudah merasa terikat dengan standar OASIS untuk menerapkan mutu dari aspek *service* dari produk mereka.
- *Web Service* dapat saja memiliki *performance*/kinerja yang lemah dibandingkan dengan pendekatan komputasi terdistribusi lain seperti RMI, CORBA, atau DCOM. Ini merupakan suatu *trade-off* yang umum ketika memilih format yang *text-based*. XML dengan tegas tidak menghitung antar tujuan disain-nya baik singkatan dari penyandian maupun efisiensi dari uraian. Ini bisa berubah dengan standar XML *Infoset*, yang menguraikan bahasa yang XML-*based* dalam kaitan dengan hal – hal yang abstrak (unsur-unsur, atribut, logika bersarang). Penyajian *angle-bracket* (< >) secara tradisional kini dilihat sebagai suatu serialisasi ASCII (atau Unicode) dari XML, bukan XML itu sendiri. Pada model ini, serialisasi biner adalah suatu alternatif yang sama yang sah. Penyajian biner seperti SOAP MTOM menjanjikan untuk meningkatkan efisiensi *wire* dari XML messaging [WUW-08].

2.2.1.1. Arsitektur *Web Service*

Gambar 2. 1 : Lapisan Dasar *Web Service*
Sumber : [WUW-08]

Gambar diatas memberi suatu pandangan layer menyangkut definisi *web service* yang dinyatakan sebagai *layer*/lapisan. Bersandar pada pondasi bagi XML untuk teknologi dari *web service*, dan HTTP sebagai dasar protokol baku untuk mencapai kemampuan dari akses, deskripsi, dan penemuan/*discovery*. SOAP (*Simple Object Access Protocol*) adalah standar untuk bertukar pesan – pesan berbasis XML melalui jaringan komputer atau sebuah jalan untuk program yang berjalan pada suatu sistem operasi (OS) untuk berkomunikasi dengan program pada OS yang sama maupun berbeda dengan menggunakan HTTP dan XML sebagai mekanisme untuk pertukaran data.

Adapun standard uraian *Service* untuk *Web Service* adalah misalnya terdapat suatu jaringan yang umum untuk berkomunikasi dan suatu satuan format dan interpretasi message yang disetujui secara umum, maka apa persyaratan yang berikutnya untuk memudahkan komunikasi antara penyedia service (*provider*) dan pemohon *service* (*requester*)? Mereka harus mempunyai suatu pemahaman semantik yang umum tentang isi dari *message* mengenai apa yang mereka maksud untuk memenuhi transaksi mereka pada jaringan tersebut.

Suatu pemohon yang potensial harus mengetahui service apa yang tersedia dari penyedia *service*, format *message* apa yang diperlukan untuk membuat permohonan, biaya – biaya apa yang dilibatkan, dan lain-lain. Seorang pedagang

yang ingin menggunakan penyedia service untuk menjual barang – barangnya harus mampu menguraikannya sedemikian sehingga penyedia *service* dapat memahami uraian mengenai barang-barang tersebut dan menyampaikannya ke para pembeli yang potensial.

Standarisasi dari uraian *service* untuk mendukung *web service* dicapai melalui WSDL. Bahasa ini menggambarkan *interface* yang diperlukan untuk interaksi antara pemohon dan penyedia *service* dan juga menentukan penempatan/lokasi dari penyedia *service* tersebut.

Penyedia service menerbitkan suatu *service* dengan membuat dokumen uraian WSDL-nya tersedia untuk pemohon yang potensial. Ini bisa dilakukan dalam berbagai cara, tetapi satu cara yang standar adalah bagi penyedia *service* untuk mendaftarkan service dengan suatu *registry* (pencatatan) dan bagi pemohon *service* untuk menemukan service dengan pencarian *registry* tersebut. Spesifikasi yang digunakan untuk pencatatan adalah spesfikasi UDDI [WUW-08].

2.2.1.2. Teknologi yang Digunakan pada *Web Service*

Beberapa teknologi yang menjadi standar dalam membangun *web service*, ialah XML, SOAP, dan WSDL. Sekilas tentang teknologi *web service* tersebut akan dibahas dibawah ini.

2.2.1.2.1. XML (*eXtensible Markup Language*)

XML singkatan dari *eXtensible Markup Language*. Dimana XML ini sudah menjadi bagian yang sangat penting bagi *programmer* yang ingin mengembangkan kemampuan melakukan transfer data antarplatform. XML juga memiliki kemampuan integrasi data disamping pertukaran data antarplatform. XML memiliki kelebihan dibanding bahasa pemrograman lainnya, yakni [SIS-04] :

- *Intelligence*. XML dapat menangani berbagai level kompleksitas dan Markup bertingkat – tingkat.
- *Adaptation*. Tag dapat diberi nama atau inisial sesuai dengan keinginan dan kebutuhan pembuat.

- *Maintenance*. Mudah dalam pemeliharaan karena hanya berisi data dan markup. *Stylesheet* dan *link* terpisah dari dokumen.
- *Simplicity*. XML memiliki struktur pemrograman yang sangat sederhana dibanding bahasa pemrograman lainnya.
- *Portability*. XML mempunyai sifat portabilitas yang bagus karena memisahkan antara data dan pengaturan tampilan.

Contoh *file* XML dapat dilihat sebagai berikut :

```
<?xml version="1.0" encoding="utf-8" ?>
<DataKampus>
      <Mahasiswa>
            <Nama>Firman Faizal Rahman</Nama>
            <Nim>0310630050</Nim>
      </Mahasiswa>
      <Mahasiswa>
            <Nama>Tjahya Indrawati</Nama>
            <Nim>0310630105</Nim>
      </Mahasiswa>
</DataKampus>
```

2.2.1.2.2. SOAP (*Simple Object Access Protocol*)

XML, SOAP , dan pemrograman multitier adalah teknologi yang luar biasa ynag dapat berkolaborasi dengan *programming* lain dalam membuat aplikasi yang besar dan komplek. Teknologi ini memungkinkan untuk membangun aplikasi dengan memisahkan antara lapisan data dari lapisan presentasi, mudah mengorganisasikan, serta aplikasi tersebut dapat dikembangkan hingga komplek dengan jumlah *user* yang banyak.

SOAP singkatan dari *Simple Object Access Protocol*. SOAP adalah protokol untuk pertukaran informasi dengan desentralisasi dan terdistribusi. Dapat dikatakan bahwa [SIS-04] :

- SOAP adalah protokol komunikasi.
- SOAP untuk berkomunikasi antar aplikasi.
- SOAP adalah sebuah format untuk mengirim pesan.
- SOAP didesain untuk berkomunikasi melalui internet.
- SOAP adalah platform yang independen.
- SOAP adalah bahasa yang independen.
- SOAP berbasis XML.

- SOAP sederhana dan dapat dikembangkan.
- SOAP akan dibangun sebagai W3C Standar.

Sebuah pesan SOAP adalah dokumen XML yang berisi elemen - elemen berikut [SIS-04] :

a. *Envelope element* yang mengidentifikasikan dokumen XML sebagai sebuah pesan SOAP. Elemen ini bersifat wajib.
b. Elemen *header* yang berisi informasi *header*. Elemen ini opsional.
c. Elemen *body* yang berisi panggilan dan merespon informasi. Elemen ini bersifat wajib.
d. *Fault element* yang berisi pesan kesalahan yang terjadi pada waktu proses. Elemen ini opsional.

Contoh pesan SOAP sebagai berikut :

SOAP *request* :

```
POST /InStock HTTP/1.1
Host: www.example.org
Content-Type: application/soap+xml; charset=utf-8
Content-Length: nnn

<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
  <soap:Body xmlns:m="http://www.example.org/stock">
    <m:GetStockPrice>
      <m:StockName>IBM</m:StockName>
    </m:GetStockPrice>
  </soap:Body>
</soap:Envelope>
```

SOAP *response* :

```
HTTP/1.1 200 OK
Content-Type: application/soap+xml; charset=utf-8
Content-Length: nnn
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
  <soap:Body xmlns:m="http://www.example.org/stock">
    <m:GetStockPriceResponse>
      <m:Price>34.5</m:Price>
```

```
    </m:GetStockPriceResponse>
  </soap:Body>
</soap:Envelope>
```

2.2.1.2.3. WSDL (*Web Service Describe Language*)

Web Service Describe Language adalah format XML untuk menjelaskan tentang *web service*. Secara singkat WSDL dapat dijabarkan sebagai berikut [SIS-04]:

- WSDL ditulis dalam bahasa XML.
- WSDL adalah dokumen XML.
- WSDL digunakan untuk mendiskripsikan *web service*.

WSDL dapat dikatakan representasi kontrak antara *requestor* dan *provider*. Secara lebih teknis adalah representasi kontrak antara kode klien dan kode di *server*. Dengan menggunakan WSDL, klien dapat memanfaatkan fungsi – fungsi publik yang disediakan oleh *server*.

WSDL adalah dokumen XML yang *machine-readable* bukan *human-readable*. Oleh karena itu, dokumen tersebut dapat dimanfaatkan untuk melakukan otomatisasi proses pengintegrasian layanan ke aplikasi *requestor*. Dimana permintaan dari *requestor* akan membangkitkan *provider* untuk menghasilkan dokumen WSDL yang akan dimanfaatkan secara langsung oleh aplikasi *requestor*. Struktur utama dari dokumen WSDL adalah sebagai berikut [SIS-04] :

- **Elemen definisi** (*definition element*) harus merupakan root dari dokumen WSDL. Dimana mendefinisikan nama dari *web service*, mendeklarasikan *namespace*, dan semua deskripsi elemen.
- **Elemen tipe** (*types element*) mendeskripsikan semua tipe data yang digunakan antara klien dan server. WSDL tidak terikat secara eksklusif terhadap suatu sistem penulisan, tetapi ia menggunakan spesifikasi skema W3C XML sebagai defaultnya. Jika hanya menggunakan tipe sederhana, seperti string dan integer, elemen type tidak digunakan.
- **Elemen pesan** (*messages element*) menjelaskan pesan satu arah. Dimana mendefinisikan nama dari *message* dan berisi kosong atau lebih elemen bagian pesan yang dapat merujuk ke parameter pesan atau nilai pesan

kembali.

- **WSDL Port** menjelaskan *interface* (operasi legal) yang diekspos oleh *web service*. Elemen <portType> merupakan elemen yang paling penting, karena mendefinisikan *web service*, operasi yang dapat dijalankan dan pesan yang dapat terlibat.

2.2.2. *E-Commerce*

E-Commerce memiliki kepanjangan *Electronic Commerce* dimana merupakan perniagaan elektronik yang menjadi bagian dari *Electronic*. Secara umum definisi *E-Commerce* adalah suatu cara berbelanja atau berdagang secara *online* atau *direct selling* yang memanfaatkan fasilitas internet dimana terdapat *website* yang dapat menyediakan layanan "*get and deliver*". *E-Commerce* akan merubah semua kegiatan *marketing* dan juga memangkas biaya – biaya operasional untuk kegiatan *trading* (perdagangan) [AND-08].

Keuntungan yang diperoleh dengan menggunakan transaksi melalui *E-Commerce* bagi suatu perusahaan adalah sebagai berikut [AND-08] :

- Meningkatkan pendapatan dengan menggunakan *online channel* yang biayanya lebih murah.
- Mengurangi biaya – biaya yang berhubungan dengan kertas, seperti biaya pos surat, pencetakan, *report*, dan sebagainya.
- Mengurangi keterlambatan dengan menggunakan *transfer* elektronik/pembayaran yang tepat waktu dan dapat langsung dicek.
- Mempercepat pelayanan ke pelanggan dan pelayanan lebih responsif.

2.2.2.1. Karakteristik *E-Commerce*

Berbeda dengan transaksi perdagangan biasa, transaksi *E-Commerce* memiliki beberapa karakteristik khusus, yaitu [AND-08] :

a. Transaksi tanpa batas

Sebelum era internet, batas – batas geografi menjadi penghalang suatu perusahaan atau individu yang ingin *go-international*. Sehingga, hanya perusahaan atau individu yang bemodal besar dapat memasarkan produknya ke luar negeri. Dewasa ini dengan internet pengusaha kecil atau menengah dapat

memasarkan produknya secara internasional cukup dengan membuat *website* atau memasang iklan – iklan di internet tanpa batas waktu (24 jam), dan tentu saja pelanggan dari seluruh dunia dapat mengakses situs tersebut dan melakukan transaksi *online*.

b. Transaksi anonim

Para penjual dan pembeli dalam transaksi melalui internet tidak harus bertemu muka satu sama lainnya. Penjual tidak memerlukan nama pembeli sepanjang mengenai pembayarannya telah diotorisasi oleh penyedia sistem pembayaran yang ditentukan, yang biasanya dengan kartu kredit.

c. Produk digital dan non digital

Produk – produk digital seperti *software* komputer, musik, dan produk lain yang bersifat digital dapat dipasarkan melalui internet dengan cara men-*download* secara elektronik. Dalam perkembangannya objek yang ditawarkan melalui internet juga meliputi barang – barang kebutuhan hidup lainnya.

d. Produk barang tak berwujud

Banyak perusahaan yang bergerak di bidang *E-Commerce* dengan menawarkan barang tak berwujud seperti data, *software* dan ide – ide yang dijual melalui internet.

2.2.2.2.Sistem Pembayaran

Beberapa situs *E-Commerce* di luar negeri menggunakan sistem pembayaran *online*, seperti *PayPal*, *eBay*, dan sebagainya. Sebagian besar *costumer* yang sering melakukan belanja *online* menggunakan *payment gateway* tersebut. Akan tetapi, terdapat beberapa kendala dalam menggunakan pola *payment gateway* dalam proses pembayaran *online* di Indonesia.

Misalnya budaya kartu kredit di Indonesia bisa dikatakan masih sangat muda. Sehingga banyak masyrakat yang tidak menggunakan kartu kredit sebagai alat pembayaran. Padahal sebagian besar transaksi *online* menggunakan kartu kredit.

Dengan adanya kendala tersebut, beberapa metode yang dilakukan dalam melakukan transaksi di *E-Commerce*, khususnya di Indonesia, yakni :

1. Transaksi model-ATM, yang menyangkut hanya institusi finansial dan

pemegang *account* yang akan melakukan pengambilan atau mendeposit uangnya dari *account* masing-masing.

2. Pembayaran dua pihak tanpa perantara, transaksi dilakukan langsung antara dua pihak tanpa perantara menggunakan uang nasional-nya.
3. Pembayaran dengan perantaraan pihak ke tiga, umumnya proses pembayaran yang menyangkut debit, kredit maupun *check* masuk dalam kategori ini.
4. *Micropayment*, dalam bahasa sederhananya adalah pembayaran untuk uang recehan yang kecil-kecil. Mekanisme *Micropayment* ini penting dikembangkan karena sangat diperlukan pembayaran receh yang kecil tanpa *overhead* transaksi yang tinggi.
5. *Anonymous digital cash*, uang elektronik yang di enkripsi, di dahului oleh David Chaum dengan *Digicash*-nya. Uang elektronik menjamin *privacy* dari *user cash* tetap terjamin sama seperti uang kertas maupun *coin* yang kita kenal.

Yang sudah umum di *webstore* - *webstore* di Indonesia adalah nomor 1. Transaksi model-ATM, mungkin dapat dikatakan yang paling mudah dan *flexibel*, dengan banyaknya jaringan ATM yang sekarang ini ditambah dengan internet/*sms banking* sangat memberikan kenyamanan bagi kedua belah pihak. Uang yang masuk ke rekening mudah di cek, sehingga setelah itu barang dapat di *delivery* [RAH-08].

2.2.3. HTML dan XHTML

HTML (*Hyper Text Markup Language*) adalah dokumen teks khusus yang digunakan oleh *web browser* untuk merepresentasikan teks dan grafik. HTML mendefinisikan *syntax* dan penempatan sasaran khusus yang ada didalamnya yang tidak ditampilkan oleh *browser*. HTML mengatur bagaimana menampilkan isi dokumen, termasuk teks, gambar dan media pendukung lainnya. Bahasa ini juga membuat dokumen menjadi interaktif melalui link *hypertext* khusus, yang menghubungkan suatu dokumen dengan dokumen lainnya pada suatu komputer atau komputer lainnya seperti juga pada sumber-sumber pada internet. HTML berbasiskan pada SGML, *the Standard Generalized Markup Language* [KEN-06].

Dokumen HTML tersusun dari teks yang mendefinisikan isi dari dokumen dan *tag* yang mendefinisikan struktur dan kenampakan dokumen. Struktur dari sebuah dokumen HTML adalah sederhana.

```
<html>
<head>
<title>Title of page</title>
</head>
<body>
This is my first homepage
</body>
</html>
```

Gambar 2. 2 : Contoh dokumen HTML sederhana
Sumber : [KEN-06]

XHTML (*Extensible Hyper Text Markup Language*) adalah reformulasi dari HTML yang digambarkan sebagai sebuah aplikasi XML (*Extensible Markup Language*). XHTML hampir identik dengan HTML versi 4.01 tetapi lebih kaku dan jelas. HTML memiliki beberapa kelemahan dimana kadang kala suatu kode HTML yang tidak ditulis sesuai aturan HTML tetap dapat dijalankan dengan baik di *browser*. XHTML merupakan kombinasi dari HTML dan XML dimana XML adalah sebuah bahasa *mark up* dimana segala sesuatunya harus dituliskan dengan benar sehingga menghasilkan dokumen yang tersusun baik. Kombinasi kedua bahasa ini akan menghasilkan dokumen yang tersusun dengan baik yang bekerja pada semua *browser*.

Perbedaan yang paling penting dari HTML dan XHTML adalah semua elemen pada XHTML harus disarangkan (*nested*) dengan tepat, setiap elemen harus ditutup dan menggunakan huruf kecil dan setiap dokumen harus memiliki satu elemen dasar [KEN-06].

Tag adalah suatu penanda atau perintah yang dimasukkan dalam dokumen yang menyatakan bagaimana format dari dokumen tersebut. Pada HTML, *tag* tidak perlu ditutup dan tidak *case sensitive* tetapi pada XHTML *tag* harus ditutup dan harus menggunakan huruf kecil.

Beberapa contoh *tag* yang digunakan pada HTML maupun XHTML adalah sebagai berikut :

Tabel 2. 1 : Contoh tag HTML dan XHTML

Tag	Keterangan
<html>	Mendefinisikan dokumen HTML
<head>	Mendefinisikan tentang informasi pada dokumen
<title>	Mendefinisikan judul dokumen
<body>	Mendefinisikan elemen pada *body* dokumen
<b>	Teks cetak tebal
<i>	Teks cetak miring
	Mendefinisikan pembagian dokumen
<form>	Mendefinisikan *form*
<table>	Mendefinisikan tabel

Sumber : [KEN-06]

2.2.4. PHP (PHP *Hypertext Preprocessor*)

PHP (akronim rekursif untuk PHP: *Hypertext Preprocessor*) adalah bahasa scripting *open source* yang digunakan secara luas untuk berbagai kebutuhan pemrograman. Secara khusus lebih cocok digunakan untuk pembangunan web dan dapat dipasangkan dalam HTML [ACH-07].

Sebagian besar sintaks mirip dengan bahasa C, Java dan Perl, ditambah beberapa fungsi PHP yang spesifik. Contoh penggunaan PHP:

```
<html>
    <head>
        <title>Example</title>
    </head>
    <body>

        <?php
        echo "Hi, I'm a PHP script!";
        ?>

    </body>
</html>
```

Gambar 2. 3 : Contoh program PHP
Sumber : [ACH-07]

Seluruh aplikasi berbasis *web* dapat dibuat dengan PHP. Namun kekuatan yang paling utama PHP adalah pada konektivitasnya dengan sistem basis data di dalam *web*. Sistem basis data yang dapat didukung oleh PHP diantaranya adalah [ACH-07] :

1. dBase
2. IBM DB2
3. Informix
4. Interbase
5. MySQL
6. Oracle
7. PostgreSQL
8. Sybase

PHP merupakan *software* yang *open source* yang dapat di*download* secara gratis dari situs resminya yaitu http://www.php.net. PHP dapat berjalan di berbagai sistem operasi seperti Windows 98/NT, UNIX/LINUX, Solaris maupun Macintosh. *Software* ini juga dapat berjalan pada *web server* seperti PWS (*Personal Web Server*), Apache, IIS, AOLServer, fhttpd, phttpd dan sebagainya. PHP juga merupakan bahasa pemrograman yang dapat kita kembangkan sendiri seperti menambah fungsi-fungsi baru. Keunggulan lainnya dari PHP adalah PHP juga mendukung komunikasi dengan layanan seperti protocol IMAP, SNMP, NNTP, POP3 bahkan HTTP.

PHP dapat diinstal sebagai bagian atau modul dari Apache Web Server atau sebagai CGI *script* yang mandiri. Banyak keuntungan yang dapat diperoleh jika menggunakan PHP sebagai modul dari Apache di antaranya adalah :

1. Tingkat keamanan yang cukup tinggi.
2. Waktu eksekusi yang lebih cepat dibandingkan dengan bahasa pemograman *web* lainnya yang berorientasi pada *server-side scripting*.
3. Akses ke sistem basis data yang lebih fleksibel seperti MySQL.

2.2.4.1. *Syntax* PHP

Saat PHP melakukan *parsing* suatu file, PHP akan mencari pembuka dan penutup *tag* yang akan memberitahukan kapan PHP akan mulai atau berhenti

menerjemahkan kode yang ada didalamnya. Ada empat pasangan *tag* pembuka dan penutup berbeda yang dapat digunakan pada PHP, seperti pada contoh berikut ini yang tercetak tebal [ACH-07] :

1. <?php echo 'if you want to serve XHTML documents, do like this'; ?>
2. <script language="php">
   echo 'some editors (like FrontPage) don\'t like processing instructions';
   </script>
3. <? echo 'this is the simple SGML processing instruction'; ?>
4. <% echo 'You may optionally use ASP-style tags'; %>

Tag pada contoh nomor satu dan dua selalu tersedia, contoh nomor satu paling umum digunakan dan direkomendasikan dibandingkan contoh kedua. Contoh nomor tiga dan empat adalah *short tag* dan ASP *style*, keduanya umumnya tidak direkomendasikan.

Seperti pada bahasa C dan Perl, PHP membutuhkan instruksi untuk mengakhiri berupa titik koma pada akhir dari setiap pernyataan. *Tag* penutup dari sebuah blok kode PHP secara otomatis menyiratkan titik koma sehingga tidak diperlukan titik koma untuk mengakhiri baris terakhir pada blok PHP.

PHP mendukung delapan tipe data primitif yaitu *boolean*, *integer*, *float*, *string*, *array*, *object*, *resource*, dan *NULL*.

2.2.4.2. Cookies

PHP secara langsung mendukung HTTP *cookies*. *Cookies* adalah mekanisme untuk menyimpan data dalam *browser* yang dikendalikan dan kemudian melacak atau mengidentifikasi pemakai asalnya. *Cookies* adalah bagian dari HTTP *header* sehingga fungsinya harus dipanggil sebelum ada keluaran apapun yang dikirim ke *browser*.Untuk mengatur *cookies* digunakan fungsi ***setcookie***() atau ***setrawcookie***().

```
<?php
$value = 'something from somewhere';

setcookie("TestCookie", $value);
setcookie("TestCookie", $value, time()+3600);
?>


<?php
// Print an individual cookie
echo $_COOKIE["TestCookie"];
echo $HTTP_COOKIE_VARS["TestCookie"];

// Another way to debug/test is to view all cookies
print_r($_COOKIE);
?>
```

Gambar 2. 4 : Contoh penggunaan cookies
Sumber : [ACH-07]

Dalam HTTP, *cookie* adalah bagian dari teks yang tidak lebih besar dari 4 *kilobytes* atau 4096 *byte*. Meskipun semua *server* dapat mencoba untuk mengirimkan *cookie* ke *browser* klien, tidak pernah ada jaminan bahwa *browser* pada klien akan menerima *cookie* tersebut. Lebih lanjut, *browser* hanya akan mengirim kembali *cookie* ke domain yang membuatnya [COG-04].

Tabel 2. 2 : Parameter *cookies*

Parameter	Keterangan
NAME=VALUE	Berupa *string* dan merupakan sederetan karakter, tidak termasuk titik koma, koma dan spasi.
expires=DATE	Menyatakan masa berlaku dari *cookie*, ketika mencapai kadaluarsa, *cookie* tidak akan disimpan atau diberikan lagi. Format yang digunakan DD-MM-YYYY HH:MM:SS
domain=DOMAIN_NAME	Menyatakan *domain server* yang mendefinisikan *cookies*.
path=PATH	Digunakan untuk menentukan *subset* dari URL dalam *domain* dimana *cookies* tersebut berlaku. Jika *cookie* telah melewati pencocokan *domain*, kemudian nama *path* dari URL dibandingkan dengan atribut ini dan jika cocok maka *cookie* dinyatakan *valid* dan dikirim bersama permintaan URL.

Sumber : [ACH-07]

2.2.4.3. Session

Salah satu penggunaan *cookies* yang paling bermanfaat adalah kemampuan untuk membuat *session*. Penggunakan *session* dalam PHP akan memberikan kemampuan untuk menyimpan variabel termasuk kelas dan *array* diantara eksekusi *script* dan memanggilnya kemudian. Agar sistem ini berfungsi,

web server harus dapat mengidentifikasi suatu *web browser* dari lainnya dan disinilah *cookies* memainkan perannya [COG-04].

Dukungan terhadap *session* pada PHP meliputi cara untuk menyajikan data melalui akses yang berikutnya. Ketika ada pengunjung yang mengakses sebuah situs *web*, PHP akan memeriksa secara otomatis atau berdasarkan permintaan atau secara tidak langsung apakah suatu *session identifier* yang spesifik telah dikirim bersamaan dengan permintaan [ACH-07].

Session akan dimulai ketika klien mulai masuk sebuah situs dan akan berakhir begitu klien tersebut menutup halaman situs yang telah dibukanya. Klien akan mendapat variabel yang terus ada selama ia melakukan kunjungannya tersebut. Setiap kali suatu *session* dibentuk, maka akan terdapat referensi yang menunjuk ke *session* yang bersangkutan. Referensi ini dikenal dengan *session identifer*. *Session identifier* ini akan disimpan sebagai *cookie* apabila fasilitas *cookie* dalam keadaan dihidupkan pada *browser*, sedangkan data *session* akan disimpan di *server*.

Modul *session* tidak dapat menjamin bahwa informasi yang tersimpan dalam suatu *session* hanya dilihat oleh pemakai yang membuat *session* tersebut. Berdasarkan hal tersebut diperlukan perhitungan tambahan untuk secara aktif melindungi integritas dari *session*, bergantung pada nilai yang berhubungan dengannya.

2.2.5. Basis Data

Basis data adalah kumpulan dari data, secara khusus menjelaskan aktifitas dari satu atau lebih organisasi yang berhubungan. Basis data dapat dianalogikan dengan lemari arsip atau lebih tepatnya sederetan lemari arsip dimana basis data adalah tempat penyimpanan data yang terstruktur. Tujuan keseluruhan dari tempat penyimpanan tersebut adalah untuk mengelola data untuk suatu tujuan keorganisasian [BEY-04].

Basis data biasanya juga berhubungan dengan sederetan properti yang saling berhubungan, yaitu :

1. *Data sharing*, data yang tersimpan di basis data tidak hanya diakses oleh satu orang saja, basis data diakses oleh lebih dari satu orang dan mungkin

dalam waktu yang bersamaan.

2. *Data integration*, salah satu tanggung jawab utama dari penggunaan basis data adalah untuk memastikan bahwa data tetap terintegrasi. Basis data seharusnya adalah koleksi data dimana idealnya tidak ada redundansi data.
3. *Data integrity*, hal ini berarti jika suatu relasi antar obyek eksis di dunia nyata, dimana obyek tersebut direpresentasikan oleh data dalam sebuah basis data maka perubahan yang dibuat pada satu sisi pada relasi tersebut seharusnya secara tepat dicerminkan dalam perubahan yang dibuat pada sisi lain dalam relasi tersebut.
4. *Data security*, salah satu cara utama untuk memastikan integritas dari suatu basis data adalah dengan membatasi akses, dengan kata lain mengamankan basis data.
5. *Data abstraction*, basis data dapat dilihat sebagai model dari realitas. Informasi yang disimpan dalam basis data biasanya adalah representasi dari properti dari suatu obyek dalam dunia nyata karena itulah basis data adalah abstraksi dari dunia nyata.
6. *Data independence*, jika suatu perubahan dibuat pada basis data dari suatu aplikasi, program pada aplikasi yang menggunakan data yang terpengaruh perubahan tersebut tidak perlu dirubah atau sebaliknya.

2.2.5.1. Sistem Manajemen Basis Data (SMBD)

Sistem Manajemen Basis Data (SMBD) adalah perangkat lunak yang didesain untuk membantu pengelolaan dan pengolahan kumpulan data yang besar. SMBD memiliki serangkaian fasilitas yang terorganisasi untuk mengakses dan mengelola satu atau lebih basis data [BEY-04].

Fungsi dari Sistem Manajemen Basis Data adalah untuk mendefinisikan, mengelola, mengambil dan mengontrol data. Beberapa keuntungan menggunakan SMBD adalah sebagai berikut :

1. Independensi data

   Program aplikasi harus sebisa mungkin berdiri sendiri dari detil representasi data dan penyimpanan data. SMBD dapat menyediakan tampilan implisit dari data untuk mengisolasi kode-kode aplikasi dari

detil-detil tersebut.

2. Akses data yang efisien

   SMBD menggunakan berbagai teknik yang canggih untuk menyimpan dan menerima data secara efisien. Fitur ini penting terutama jika data disimpan pada media penyimpanan eksternal.

3. Integrasi dan keamanan data

   Jika data selalu diakses melalui SMBD, SMBD dapat memperkuat batasan kesatuan pada data. Selain itu SMBD dapat memperkuat kontrol akses yang mengatur data apa saja yang dapat terlihat pada kelas pemakai yang berbeda.

4. Administrasi data

   Ketika beberapa pemakai berbagi data, pemusatan administrasi data dapat menawarkan peningkatan yang signifikan.

5. Pengaturan akses serentak dan pengendalian kerusakan

   SMBD menjadwal akses yang terjadi secara serentak pada data dan melindungi para pemakainya dari efek kegagalan sistem.

6. Pengurangan waktu pembangunan aplikasi

   SMBD mendukung berbagai fungsi yang umum untuk aplikasi-aplikasi dalam mengakses data yang tersimpan dalam basis data. Dalam hubungannya dengan antarmuka tingkat tinggi pada data, memfasilitasi pembangunan aplikasi secara cepat.

Tujuan utama dari sistem basis data adalah untuk memberikan tampilan abstrak dari data. Sistem menyembunyikan sebagian detil dari bagaimana data disimpan dan dikelola. Pengembang basis data menyembunyikan kompleksitas dari pemakai melalui beberapa level abstraksi untuk mempermudah interaksi pemakai dengan sistem. Level-level abstraksi tersebut adalah [SIK-05]:

1. Tingkat fisik (*physical level*)

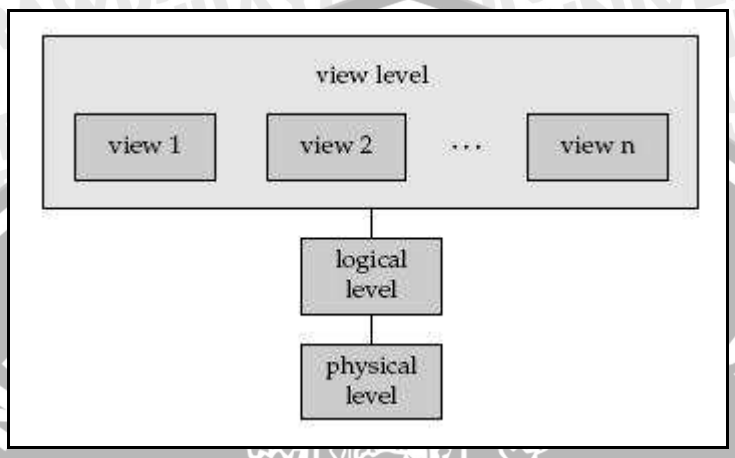
   Tingkat abstraksi terendah yang menjelaskan bagaimana *record* disimpan.

2. Tingkat logikal (*logical level*)

   Tingkat abstraksi lebih tinggi berikutnya yang menjelaskan data-data apa yang disimpan dalam basis data dan relasi apa yang ada diantara data-data tersebut.

3. Tingkat penampakan (*view level*)

   Tingkat abstraksi tertinggi yang menjelaskan hanya sebagian dari keseluruhan basis data. Pada tingkat ini detil dari tipe data dan informasi disembunyikan untuk tujuan keamanan.



Gambar 2. 5 : Level Abstraksi
Sumber : [SIK-05]

2.2.5.2. Sistem Manajemen Basis Data (SMBD)

Basis data terdiri dari beberapa komponen untuk membentuk struktur basis data yang diperlukan. Tingkatan dari komponen basis data terdiri dari :

1. Karakter (*Characters*)

   Karakter merupakan bagian data terkecil, dapat berupa karakter numerik, huruf, ataupun karakter-karakter khusus yang membentuk suatu *field*.

2. *Field*

   Field merepresentasikan suatu atribut dari *record* yang menunjukan suatu item dari data, seperti nama, alamat, dan lain sebagainya. Kumpulan dari *field* akan membentuk suatu *record*.

3. *Record*

   Record adalah sekumpulan *field* atau item yang saling berhubungan dan merepresentasikan suatu elemen data, dimana kumpulan dari *record* akan membentuk suatu *file* atau tabel.

4. *File*

   File adalah kumpulan *record-record* yang menggambarkan kesatuan data yang sama.

5. Basis Data

Basis data adalah kumpulan dari *file* atau tabel.

2.2.5.3. *Model Entity Relationship*

Model data atau *data model* adalah kumpulan *tool* konseptual dalam menggambarkan data, relasi antar data, semantik data dan batasan data. Ada beberapa model data diantaranya adalah *entity-relationship model*.

Model *entity-relationship* memberikan kemungkinan untuk menjelaskan data yang di dunia nyata dalam bentuk obyek dan hubungannya diantaranya. Model data ini menggunakan tiga notasi dasar yaitu entitas, relasi dan atribut [SIK-05].

2.2.5.3.1. Entitas, Relasi dan Atribut

1. Entitas

Entitas adalah ”sesuatu” atau ”objek” di dunia nyata yang dapat dibedakan dari objek lain. Setiap entitas memiliki sekumpulan properti atau karakteristik dan nilai untuk beberapa kumpulan karateristik tersebut mungkin saja secara unik mengidentifikasi entitas tersebut. Entitas dapat berupa obyek yang nyata atau konkrit, misalnya kursi atau pelanggan dan dapat juga berupa obyek yang abstrak, misalnya jadwal atau pinjaman. Setiap entitas memiliki atribut tertentu.

Himpunan entitas (*entity set*) adalah sekumpulan entitas yang sejenis dan berada dalam lingkup yang sama, misalnya semua pelanggan, semua orang yang meminjam di perpustakaan. Entitas menunjuk pada individu suatu obyek sedangkan himpunan entitas menunjuk pada rumpun dari individu tersebut.

Atribut

Setiap entitas memiliki atribut yang mendeskripsikan karakteristik atau properti dari entitas tersebut dimana setiap atribut akan memiliki nilai (*values*). Batas-batas nilai yang diperbolehkan bagi suatu atribut disebut domain.

Tipe-tipe dari atribut adalah sebagai berikut :

1. Atribut *simple* dan *composite*

Atribut *simple* adalah atribut sederhana yang tidak dapat dibagi lagi dalam bagian-bagian sedangkan atribut *composite* adalah atribut yang dapat

dibagi lagi dalam beberapa bagian, misalnya nama, dapat dibagi lagi menjadi nama depan, nama inisial dan nama belakang.

2. Atribut *single-valued* dan *multivalued*

Atribut *single-valued* adalah atribut yang memiliki paling banyak satu nilai untuk setiap baris data sedangkan atribut *multivalued* adalah atribut yang dapat diisi lebih dari satu nilai tetapi dengan jenis sama, misalnya nomor telepon, alamat.

3. Atribut *derived* atau turunan

Atribut turunan adalah atribut yang diperoleh dari pengolahan atribut lain yang berhubungan, misalnya umur. Nilai dari atribut turunan tidak disimpan tetapi dihitung ketika dibutuhkan.

Sebuah atribut memiliki nilai *null* ketika suatu entitas tidak mempunyai nilai untuk atribut tersebut. *Null* dapat juga ditujukan untuk nilai dari suatu atribut yang tidak diketahui.

2. Relasi

Relasi adalah hubungan antara beberapa entitas. Himpunan relasi adalah kumpulan semua relasi yang merupakan hubungan matematis antara $n \geq 2$ entitas dari himpunan entitas-entitas yang ada.

$\{ (e1, e2, ..., en) \mid e1 \in E1, e2 \in E2, ..., en \in En \}$

dimana : (E1, E2, ..., E*n*) adalah entitas

i. (e1, e2, ..., e*n*) adalah relasi

Banyaknya himpunan entitas yang saling berelasi ditunjukkan oleh derajat relasi. Himpunan relasi yang melibatkan dua himpunan entitas disebut binari atau berderajat dua. Secara umum himpunan relasi dalam sistem basis data adalah binari.

Pemetaan kardinalitas atau derajat kardinalitas menunjukkan jumlah maksimum entitas yang dapat berelasi dengan entitas pada himpunan entitas yang lain. Kardinalitas relasi yang terjadi antara dua himpunan entitas dapat berupa :

1. Satu ke satu (*one to one*)

Kardinalitas relasi ini berarti setiap entitas pada himpunan entitas yang pertama dapat berhubungan paling banyak dengan satu entitas pada

himpunan entitas yang kedua dan sebaliknya.

2. Satu ke banyak (*one to many*)

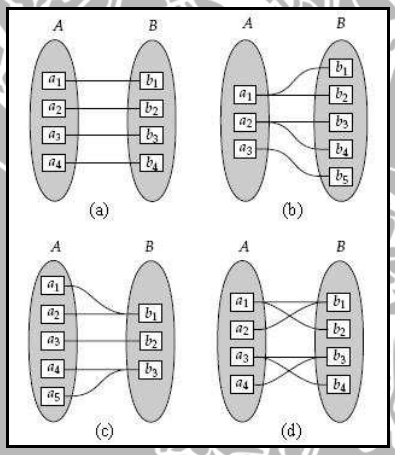
   Kardinalitas relasi ini berarti setiap entitas pada himpunan entitas yang pertama dapat berhubungan dengan banyak entitas pada himpunan entitas yang kedua tetapi tidak sebaliknya.

3. Banyak ke satu (*many to one*)

   Kardinalitas relasi ini berarti setiap entitas pada himpunan entitas yang pertama dapat berhubungan paling banyak dengan satu entitas pada himpunan entitas yang kedua sedangkan pada himpunan entitas yang kedua, setiap entitasnya dapat berhubungan dengan banyak entitas pada himpunan entitas yang pertama.

4. Banyak ke banyak (*many to many*)

   Kardinalitas relasi ini berarti setiap entitas pada himpunan entitas yang pertama dapat berhubungan dengan banyak entitas pada himpunan entitas yang kedua dan sebaliknya.



Gambar 2. 6 : Pemetaan kardinalitas (a) satu ke satu (b) satu ke banyak (c) banyak ke satu (d) banyak ke banyak
Sumber : [SIK-05]

2.2.5.3.2. *Key*

Penggunaan *key* merupakan cara untuk membedakan suatu entitas dalam himpunan entitas dengan entitas yang lain. *Key* adalah satu atau gabungan dari beberapa atribut yang dapat membedakan semua baris dalam tabel secara unik. Ada tiga macam *key* yang dapat diterapkan pada suatu tabel :

1. *Super key*

merupakan satu atau lebih atribut yang dapat membedakan setiap baris data dalam sebuah tabel secara unik.

2. *Candidate key*

   merupakan kumpulan atribut minimal yang dapat membedakan setiap baris data dalam sebuah tabel secara unik.

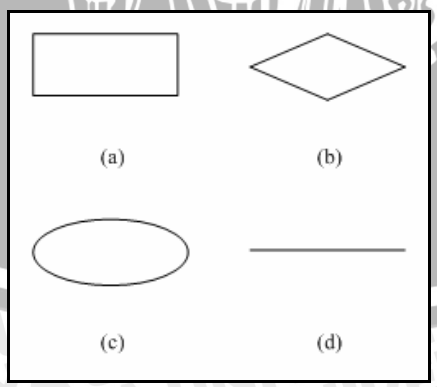
3. *Primary key*

   merupakan salah satu dari *candidate key* yang terpilih, pemilihan *primary key* dari sejumlah *candidate key* umumnya didasari pertimbangan bahwa *key* tersebut lebih sering untuk dijadikan acuan, lebih ringkas dan jaminan keunikannya lebih baik.

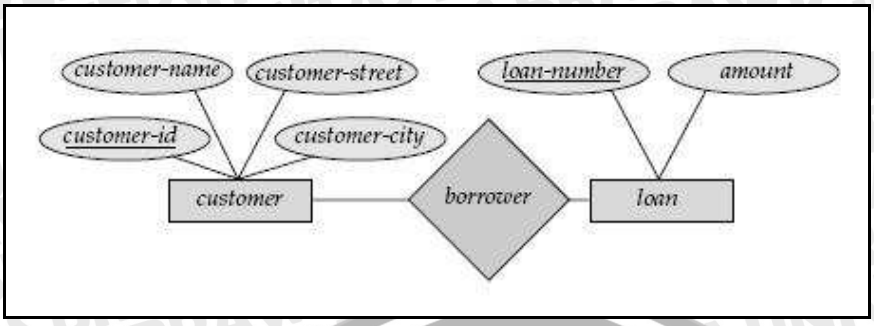
2.2.5.3.3. *Entity Relationship Diagram*

Diagram *Entity Relationship* atau diagram E-R merupakan model konseptual untuk menggambarkan struktur logikal dari basis data berbasis grafik. Komponen-komponen utama dalam diagram E-R adalah sebagai berikut :

1. Persegi panjang, menyatakan himpunan entitas.
2. Elips, menyatakan atribut, atribut yang merupakan *key* digarisbawahi.
3. Belah ketupat, menyatakan himpunan relasi.
4. Garis, menghubungkan himpunan entitas dan himpunan relasi, atribut dan himpunan entitas.
5. Kardinalitas relasi dapat dinyatakan dengan banyaknya garis cabang atau dengan pemakaian angka.



Gambar 2. 7 : Komponen Diagram E-R (a) himpunan entitas (b) himpunan relasi (c) atribut (d) penghubung
Sumber : [SIK-05]



Gambar 2. 8 : Diagram E-R
Sumber : [SIK-05]

2.2.6. MySQL

MySQL adalah *open source SQL Database Management System* yang paling populer, dikembangkan, didistribusikan dan didukung oleh MySQL AB. MySQL AB adalah perusahaan komersial yang didirikan oleh developer MySQL. Beberapa keunggulan dari MySQL antara lain :

1. MySQL adalah *Database Management System*.

   Untuk menambah, mengakses dan memproses data yang tersimpan dalam basis data di komputer diperlukan *database management system* seperti MySQL Server. Karena komputer sangat baik dalam penanganan data dalam jumlah besar, *database management system* berperan utama dalam pemrosesan sebagai alat yang berdiri sendiri maupun sebagai bagian dari aplikasi lainnya.

2. MySQL adalah *Relational Database Management System*.

   Basis data relasional menyimpan data dalam tabel yang terpisah dibanding meletakkan data dalam satu ruang penyimpanan yang besar. Hal ini dapat menambah kecepatan dan fleksibilitas. SQL atau Structured Query Language adalah bahasa standar paling umum yang digunakan untuk mengakses database dan didefinisikan oleh ANSI/ISO SQL Standard.

3. MySQL adalah *open source*.

   Open source berarti memungkinkan untuk setiap orang untuk menggunakan dan memodifikasi perangkat lunak tersebut. Setiap orang dapat melakukan *download* perangkat lunak MySQL dari internet dan menggunakannya tanpa harus membayar apapun. Perangkat lunak MySQL menggunakan GPL (GNU General Public License) untuk menjelaskan apa

yang boleh dan tidak boleh dilakukan dengan perangkat lunak tersebut pada situasi-situasi yang berbeda.

4. MySQL Database Server sangat cepat, handal dan mudah untuk digunakan.

   MySQL Server pada dasarnya dibuat untuk menangani database yang besar dengan lebih cepat daripada solusi yang telah ada dan telah berhasil digunakan dalam lingkungan produksi dengan permintaan tinggi selama beberapa tahun. Meskipun dalam pengembangan yang terus menerus, MySQL Server saat ini menawarkan kumpulan fungsi yang lebih lengkap dan berguna. Konektifitas, kecepatan dan keamanan membuat MySQL Server sangat cocok untuk mengakses basis data di internet.

5. MySQL *Server* berjalan pada aplikasi *client/server* maupun *embedded systems*.

   The MySQL Database Software adalah sistem *client/server* yang terdiri *dari multi-threaded SQL server* yang mendukung *backends* yang berbeda-beda, beberapa program klien yang berbeda dan berbagai *library, administrative tools,* dan berbagai *application programming interfaces* (*APIs*). MySQL dapat juga berperan sebagai *embedded multi-threaded library* yang dapat dihubungkan pada aplikasi untuk mendapatkan produk *standalone* yang lebih kecil, cepat, dan mudah untuk diatur.

6. Source MySQL dapat diperoleh dengan mudah.

2.2.6.1. Tipe Data

MySQL mendukung berbagai tipe data dalam beberapa kategori yaitu numerik, tanggal dan waktu, karakter atau *string*.

Tabel 2. 3 : Tipe Data Numerik

Tipe	Ukuran (Byte)	Nilai Minimum (*signed*/*unsigned*)	Nilai Maksimum (*signed*/*unsigned*)
TINYINT	1	-128	127
		0	255
SMALLINT	2	-32768	32767
		0	65535
MEDIUMINT	3	-8388608	8388607
		0	16777215

INT	4	-2147483648	2147483647
		0	4294967295
BIGINT	8	-9223372036854775808	9223372036854775807
		0	18446744073709551615

Sumber : [ANO-08]

Tabel 2. 4 : Tipe Data Tanggal dan Waktu

Tipe	Format
DATETIME	'YYYY-MM-DD HH:MM:SS'
DATE	'YYYY-MM-DD'
TIMESTAMP (versi 4.1 keatas)	'YYYY-MM-DD HH:MM:SS'
TIMESTAMP (sebelum versi 4.1)	'YYYYMMDDHHMMSS'
TIME	'HH:MM:SS'
YEAR	'YYYY'

Sumber : [ANO-08]

Tabel 2. 5 : Tipe Data Karakter/String

Tipe	Ukuran	Keterangan
CHAR (M)	M byte	Lebar data tetap
VARCHAR (M)	M byte	Lebar data bervariasi

Sumber : [ANO-08]

2.2.7. Rekayasa Perangkat Lunak

IEEE (*Institute of Electrical and Electronics Engineers*) telah mengembangkan definisi yang komprehensif mengenai rekayasa perangkat lunak yang menyatakan bahwa rekayasa perangkat lunak adalah aplikasi dari pendekatan yang sistematik, teratur, dan terukur pada pengembangan, pengoperasian dan perawatan perangkat lunak [PRE-05].

Rekayasa perangkat lunak adalah teknologi yang terdiri dari beberapa lapisan. Lapisan-lapisan tersebut meliputi proses, sekumpulan metode dan sederetan alat bantu untuk pengembangan perangkat lunak.

Dasar dari rekayasa perangkat lunak adalah lapisan proses. Proses rekayasa perangkat lunak adalah perekat yang menyatukan lapisan-lapisan teknologi dan memungkinkan pengembangan perangkat lunak komputer yang rasional dan tepat pada waktunya. Proses rekayasa perangkat lunak menjadi basis kendali manajemen pengembangan perangkat lunak dan membuat konteks untuk

penerapan metode teknis, penciptaan produk kerja (model, dokumen, data, laporan, formulir-formulir, dan sebagainya), penentuan *milestone*, terjaminnya kualitas dan adanya manajemen perubahan.

Metode rekayasa perangkat lunak menyediakan tata cara teknis untuk membangun perangkat lunak sedangkan alat bantu menyediakan dukungan otomatis atau semiotomatis untuk lapisan proses dan metode.

Pengembangan perangkat lunak memerlukan strategi dalam penetapan proses, metode dan alat bantu yang akan digunakan. Strategi ini sering disebut paradigma rekayasa perangkat lunak. Paradigma mengacu kepada cara pandang spesifik dalam usaha pencarian solusi, yaitu menyangkut konseptualisasi permasalahan dan solusi. Beberapa paradigma yang dapat diterapkan pada Rekayasa Perangkat Lunak adalah paradigma terstruktur, paradigma berorientasi objek, paradigma metode formal, paradigma berbasis komponen.

2.2.7.1. *Unified Modelling Language* (UML)

Unified Modelling Language (UML) adalah sebuah bahasa grafis untuk visualisasi, spesifikasi, konstruksi, dan dokumentasi artifak perangkat lunak. UML menyediakan cara standar untuk membuat cetak biru sistem, melingkupi hal-hal yang konseptual seperti proses bisnis dan fungsional sistem, dan juga hal-hal yang nyata seperti kelas-kelas yang ditulis dalam bahasa pemrograman yang spesifik, skema basis data, komponen perangkat lunak yang dapat digunakan ulang [BOR-05].

Saat ini UML meliputi 13 jenis diagram yaitu *class diagram*, *object diagram*, *component diagram*, *composite structure diagram*, *use case diagram*, *sequence diagram*, *communication diagram*, *state diagram*, *activity diagram*, *deployment diagram*, *package diagram*, *timing diagram*, dan *interaction view diagram*.

Dalam skripsi ini hanya akan dipakai tiga diagram yaitu *use case diagram* dalam proses analisis dan *class diagram* dan *sequence diagram* dalam proses perancangan.

2.2.7.2. Analisis Berorientasi Obyek

Sebelum membangun suatu sistem berorientasi obyek, kita perlu mendefinisikan semua kelas yang merepresentasikan masalah yang harus dipecahkan, cara dimana suatu kelas berhubungan dan berinteraksi dengan satu dan lainnya serta mekanisme komunikasi yang membuat semuanya dapat bekerja bersama. Untuk melakukan analisis berorientasi obyek, hal-hal yang harus dilakukan adalah [PRE-05]:

1. Kebutuhan dasar dari *user* harus dikomunikasikan antara konsumen dan *sofware engineer*.
2. Kelas-kelas harus diidentifikasi.
3. Hirarki dari kelas didefinisikan.
4. Relasi dari obyek ke obyek harus direpresentasikan.
5. Perilaku dari obyek harus dimodelkan.
6. Langkah-langkah dari nomor 1 sampai dengan 5 diaplikasikan secara berulang sampai model terbentuk secara utuh.

Tujuan analisis berorientasi obyek adalah mengembangkan suatu model yang menjelaskan perangkat lunak komputer untuk memenuhi kebutuhan yang didefinisikan oleh konsumen.

2.2.7.3. Proses Analisis Berorientasi Obyek

Proses analisis beorientasi obyek tidak dimulai dengan hal-hal mengenai obyek. Proses ini dimulai lebih pada pemahaman tentang bagaimana sistem yang dibuat akan dipakai, misalnya jika sistem adalah *human interaktif*, dijalankan oleh mesin, terkait dengan kontrol proses atau program lain, atau jika sistem mengkoordinasikan dan mengontrol aplikasi. Saat skenario penggunaan sistem telah didefinisikan maka proses pemodelan perangkat lunak dimulai.

2.2.7.3.1. Analisis Kebutuhan

Rekayasa kebutuhan menyediakan mekanisme yang dibutuhkan untuk memahami keinginan konsumen, menganalisis kebutuhan, menilai kelayakan, menegosiasikan solusi yang dapat diterima, menentukan solusi yang tidak ambigu, menentukan spesifikasi dan memanajemen kebutuhan untuk

ditransformasikan ke dalam sistem operasional.

Rekayasa kebutuhan terdiri dari beberapa proses untuk menganalisis kebutuhan konsumen dan menentukan solusinya, proses tersebut adalah [PRE-05]:

1. Insepsi atau permulaan

   Proses ini bertujuan untuk mewujudkan pemahaman dasar dari permasalahan, konsumen yang membutuhkan solusi, solusi yang diinginkan dan kolaborasi dan komunikasi awal yang efektif antara konsumen dan pengembang.

2. Elisitasi atau penimbulan

   Proses ini bertujuan untuk menentukan tujuan dari sistem yang dibuat, apa saja yang harus dicapai, bagaimana sistem memenuhi kebutuhan bisnis dan bagaimana sistem digunakan untuk sehari-hari.

3. Elaborasi atau pengembangan

   Informasi yang didapatkan dari konsumen pada proses insepsi dan elisitasi dikembangkan dan disempurnakan pada proses ini. Proses ini memfokuskan pada pengembangan model teknis yang disempurnakan dari fungsi, fitur dan batasan perangkat lunak.

4. Negosiasi

   Konsumen, pemakai dan pemegang keputusan lainnya memiliki pandangan yang berbeda tentang kebutuhan mereka. Pada proses ini perbedaan tersebut harus diselesaikan dengan menyusun kebutuhan sesuai tingkatannya dan kemudian mendiskusikan perbedaan yang timbul sesuai prioritas. Dengan menggunakan pendekatan iteratif, kebutuhan dielininasi, dikombinasikan, dan atau dimodifikasi sehingga tiap pihak mendapatkan kepuasan atas keputusan tersebut.

5. Spesifikasi

   Spesifikasi adalah hasil akhir yang berfungsi sebagai dasar untuk aktifitas rekayasa perangkat lunak selanjutnya.

6. Validasi

   Validasi kebutuhan menilai spesifikasi yang telah dibuat untuk memastikan bahwa semua kebutuhan perangkat lunak telah dinyatakan

secara jelas dan tidak ambigu, semua ketidakkonsistenan, hal-hal yang terlewat, dan kesalahan-kesalahan yang ada telah terdeteksi dan dikoreksi dan hasil kerja telah memenuhi standar yang dibutuhkan.

Hasil dari analisis kebutuhan dalam spesifikasi dari karateristik operasional perangkat lunak, mengindikasikan antarmuka perangkat lunak dengan elemen-elemen sistem lainnya dan menentukan batasan-batasan yang harus dipenuhi perangkat lunak. Analisis kebutuhan mengembangkan kebutuhan dasar yang dibuat pada proses rekayasa kebutuhan di awal dan membangun model-model yang menggambarkan skenario dari *user*, aktifitas fungsional, permasalahan kelas dan relasi diantaranya, *behaviour* sistem dan kelas, dan aliran data.

Analisis kebutuhan merepresentasikan informasi, fungsi dan *behaviour* yang dapat diterjemahkan pada arsitektur, antarmuka dan perancangan pada level komponen.

2.2.7.3.2. *Use Case Diagram*

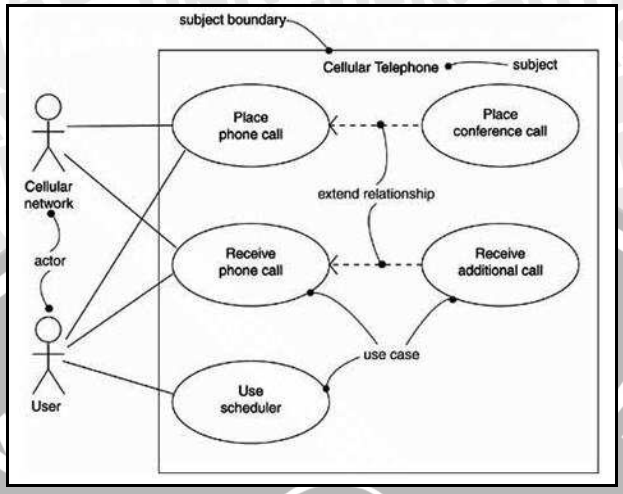
Secara esensi, *use case* menceritakan tentang bagaimana seorang *end user* berinteraksi dengan sistem dibawah kumpulan keadaaan yang spesifik. *Use case* dapat berupa teks naratif, ringkasan tugas atau interaksi, deskripsi berbasis tampilan atau representasi dalam bentuk diagram. Apapun bentuknya, *use case* menggambarkan perangkat lunak atau sistem dari pandangan *end user*.

Use case harus mencapai tujuan-tujuan berikut :

1. mendefinisikan kebutuhan operasional dan fungsional dari sistem dengan mendefinisikan skenario pemakaian yang disetujui oleh pemakai dan pengembang.
2. menyediakan deskripsi yang jelas dan tidak ambigu dari bagaimana *end-user* dan sistem berinteraksi satu dengan lainnya.
3. menyediakan dasar untuk pengujian validasi.

Langkah pertama dalam penulisan *use case* adalah mendefinisikan kumpulan aktor yang terlibat. Aktor adalah orang atau benda yang menggunakan sistem dalam konteks fungsi dan tingkah laku yang harus dijelaskan. Setelah aktor diidentifikasi maka *use case* dan relasi diantaranya dapat dibuat.

Representasi dalam bentuk diagram dari *use case* dengan menggunakan notasi UML adalah *use case diagram*. *Use case diagram* adalah salah satu diagram dalam UML untuk memodelkan kebutuhan dari sistem.



Gambar 2. 9 : Contoh *Use Case Diagram*
Sumber : [BOR-05]

Beberapa bentuk relasi yang digunakan dalam *use case diagram* adalah sebagai berikut :

Tabel 2. 6 : Relasi dalam *Use Case Diagram*

Relasi	Simbol	Keterangan
Extend	<<extend>>	menunjukkan bahwa *use case* yang menjadi target, mengembangkan *behaviour* dari *use case* asalnya
Include	<<include>>	menunjukkan bahwa *use case* asal secara eksplisit menyertakan *behaviour* dari *use case* yang lain pada lokasi yang ditentukan oleh *use case* asalnya
Generalisasi		relasi antara *use case* atau aktor yang lebih umum dengan *use case* atau aktor yang lebih khusus atau spesifik
Realisasi		menunjukkan relasi antara *use case* dan kolaborasi yang merealisasikan *use case* tersebut
Asosiasi		menunjukkan bahwa aktor dan *use case* berkomunikasi antara satu dengan lainnya, salah satunya mungkin menerima atau mengirim pesan

Sumber : [BOR-05]

2.2.7.4. Perancangan Berorientasi Obyek

Perancangan berorientasi obyek terbagi dalam dua aktivitas utama: perancangan sistem dan perancangan obyek. Perancangan sistem mendefinisikan

arsitektur produk (fungsi sistem dan kelas yang dienkapsulasi dalam subsistem). Perancangan sistem difokuskan pada spesifikasi dari tiga komponen: *user interface*, fungsi manajemen data, dan fasilitas manajemen kerja. Perancangan obyek difokuskan pada detail internal dari kelas individu dan skema penyampaian pesan. Proyek perancangan beorientasi obyek harus diperiksa kembali untuk menjamin kualitas [PRE-05].

Pada tahap perancangan ini, digunakan pemodelan dengan menggunakan dua macam diagram, yaitu *class diagram dan sequence diagram*.

2.2.7.4.1. Diagram Kelas (*Class Diagram*)

Kelas adalah kumpulan dari obyek yang berbagi atribut, operasi relasi dan semantik yang sama. Setiap kelas harus memiliki nama, dimana nama tersebut adalah unik sehingga dapat membedakannya dengan kelas lainnya. Selain itu kelas juga memiliki fitur atribut dan operasi dan fitur atau properti lainnya yang bersifat *optional*. Penamaan kelas dapat dilakukan dengan dua cara yaitu [BOR-05]:

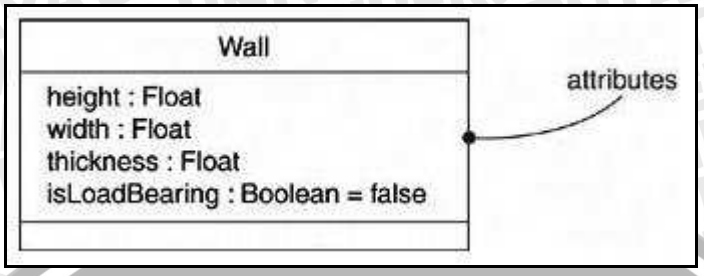
1. *Simple name*, nama kelas yang sederhana tanpa diawali dengan nama dari *package* dimana kelas tersebut berada.
2. *Qualified name*, nama kelas diawali dengan nama dari *package* dimana kelas tersebut berada.

Kelas dinotasikan dengan gambar kotak dengan nama kelas diletakkan di tengah *compartment* yang pertama. Gambar berikut menggambarkan penotasian kelas.

Gambar 2. 10 : Notasi kelas
Sumber : [BOR-05]

Kelas dapat memiliki beberapa *compartment*. *Compartment* adalah daerah dalam notasi kelas yang dipisahkan oleh garis horisontal. Notasi kelas yang umum memiliki tiga buah *compartment*, dimana *compartment* pertama berisi nama, *compartment* berisi daftar atribut dan *compartment* ketiga berisi daftar operasi.

Atribut adalah properti dari kelas yang menjelaskan suatu rentang nilai yang mungkin dimiliki oleh *instance* dari properti tersebut. Suatu kelas mungkin memiliki beberapa atribut ataupun tidak sama sekali.



Gambar 2. 11 : Atribut kelas
Sumber : [BOR-05]

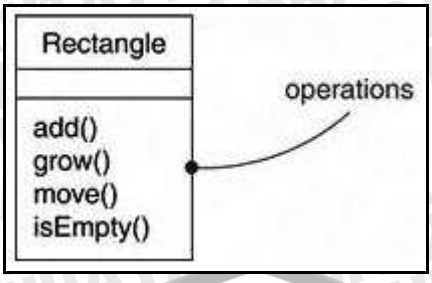
Atribut dapat dijelaskan dengan lebih lanjut dengan menyatakan nama dari atribut tersebut dengan disertai dengan nilai *default* yang mungkin. Detil desain lainnya yang dapat dijelaskan untuk atribut adalah *visibility*. *Visibility* dari suatu fitur menjelaskan apakah fitur tersebut dapat digunakan oleh *classifier-classifier* lainnya.

Tabel 2. 7 : Daftar *visibility*

Visibility	Simbol	Keterangan
public	+	Semua *classifier* dapat menggunakan fitur tersebut.
protected	#	*Classfier* itu sendiri dan semua turunannya dapat menggunakan fitur tersebut.
private	-	Hanya *classfier* itu sendiri yang dapat menggunakan fitur tersebut.
package	~	Hanya *classfier* yang dideklarasikan dalam *package* yang sama yang dapat menggunakan fitur tersebut.

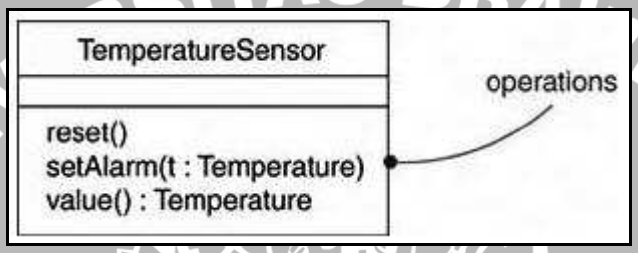
Sumber : [BOR-05]

Operasi adalah implementasi dari layanan yang dapat diminta dari semua obyek dari kelas untuk mempengaruhi *behaviour*. Dengan kata lain, operasi adalah abstraksi dari sesuatu yang dapat dilakukan pada suatu obyek yang dibagi dengan semua obyek pada kelas tersebut. Sebuah kelas dapat memiliki beberapa operasi atau tidak sama sekali. Secara grafis, operasi diletakkan dalam *compartment* dibawah atribut kelas.



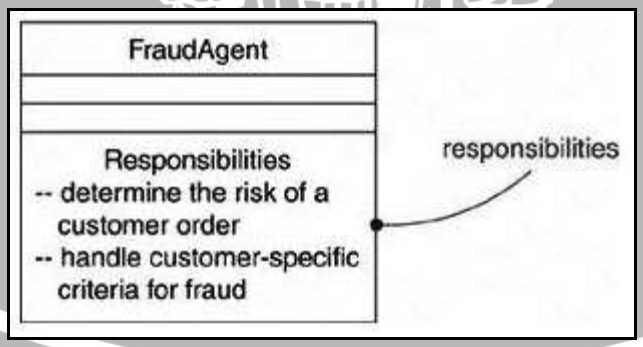
Gambar 2. 12 : Operasi kelas umum
Sumber : [BOR-05]

Operasi dapat dijelaskan lebih lanjut dengan menyatakan *signature* dari operasi tersebut, meliputi nama, tipe, dan nilai *default* dari semua parameter dan tipe pengembalian seperti yang tampak pada gambar berikut.



Gambar 2. 13 : Operasi kelas spesifik
Sumber : [BOR-05]

Responbility adalah suatu kontrak atau obligasi dari suatu kelas. Ketika suatu kelas dibuat maka terdapat pernyataan yang menyatakan bahwa semua obyek dari kelas tersebut memiliki pernyataan dan *behaviour* yang sejenis. Pada tingkatan yang lebih abstrak, atribut dan operasi yang berhubungan hanyalah suatu fitur dengan mana tanggung jawab dari suatu kelas dilaksanakan.



Gambar 2. 14 : Responbility kelas
Sumber : [BOR-05]

Representasi dalam bentuk diagram dari kumpulan kelas, antarmuka, dan kolaborasi dan relasinya adalah *class diagram*. *Class diagram* secara umum mengandung hal-hal berikut :

1. Kelas
2. Antarmuka
3. Relasi *dependency*, *generalization*, dan *association*

Seperti diagram lainnya, *class diagram* mungkin juga memiliki catatan dan batasan. *Class diagram* mungkin juga memiliki *packages* atau subsistem, keduanya digunakan untuk mengumpulkan elemen dari model menjadi kumpulan yang lebih besar.

Class diagram digunakan untuk memodelkan *design view* statis dari suatu sistem. *Class diagram* dapat digunakan untuk hal-hal berikut :

1. Untuk memodelkan perbendaharaan sistem

   Pemodelan perbendaharaan dari sistem meliputi pembuatan keputusan mengenai abstraksi-abstraksi mana yang menjadi bagian dari sistem dalam pertimbangan dan mana saja yang keluar dari ikatan tersebut. *Class diagram* digunakan untuk menjelaskan abstraksi-abstraksi ini dan tanggung jawabnya.
2. Untuk memodelkan kolaborasi sederhana

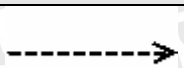
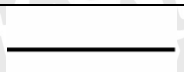
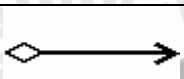
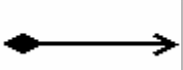
   Kolaborasi adalah perkumpulan dari kelas, antarmuka dan elemen-elemen lainnya yang bekerja bersama untuk membentuk suatu kerja sama *behaviour* yang lebih besar dari keseluruhan elemen.
3. Untuk memodelkan skema logik basis data

   Skema dari basis data dapat dimodelkan menggunakan *class diagram*.

Relasi antara satu kelas dengan kelas lainnya dapat direpresentasikan dengan beberapa cara, yaitu :

1. *Dependency*
2. *Association*
3. *Aggregation*
4. *Compotition*
5. *Generalization*

Tabel 2. 8 : Relasi dalam *class diagram*

Relasi	Simbol	Keterangan
Dependency		Relasi yang menyatakan bahwa suatu kelas menggunakan informasi dan layanan dari kelas lain.
Association		Relasi yang menyatakan bahwa suatu obyek dari suatu kelas terhubung dengan obyek dari kelas lain.
Aggregation		Relasi yang menyatakan bahwa suatu kelas mengandung kelas lainnya.
Compotition		Relasi yang menyatakan hubungan keseluruhan bagian. *Compotition* adalah jenis khusus dari *aggregation.*
Generalization		Relasi yang menyatakan hubungan antara suatu kelas yang lebih umum dengan kelas yang lebih khusus.

Sumber : [BOR-05]

2.2.7.4.2. Diagram Sekuen (*Sequence Diagram*)

Sequence diagram adalah diagram interaksi yang menitikberatkan pada urutan waktu dari suatu *messages*. Diagram interaksi pada umumnya berisi [BOR-05]:

1. Obyek atau tugas
2. Komunikasi atau *link*
3. *Message*

Sequence diagram memiliki dua fitur, yaitu :

1. *Lifeline*

   Lifeline obyek adalah garis putus-putus vertikal yang merepresentasikan eksistensi obyek pada suatu periode waktu.

2. *Focus of control*

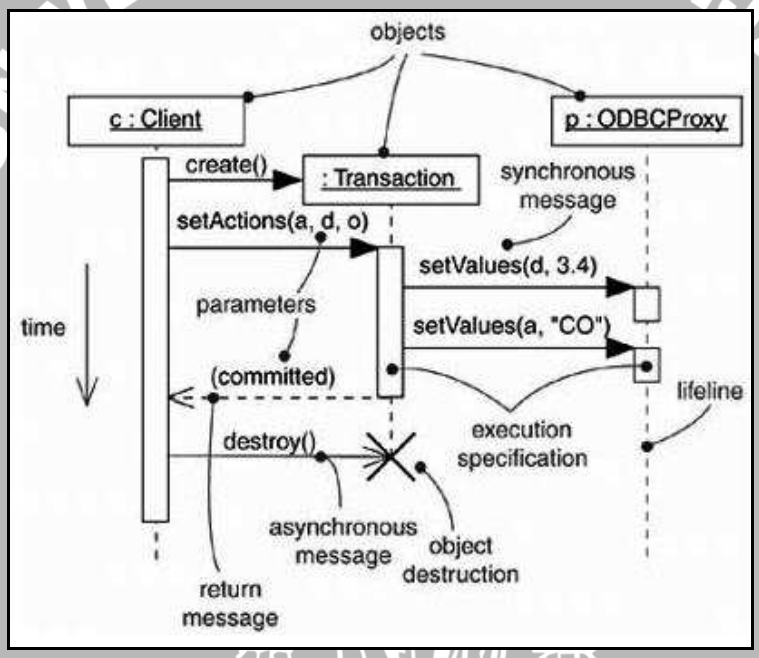
   Focus of control adalah kotak panjang dan tipis yang menunjukkan lamanya periode waktu dimana suatu obyek melakukan suatu aksi, baik langsung maupun melalui prosedur subordinatnya. Bagian atas dari kotak tersebut sebaris dengan mulainya suatu aksi, bagian bawahnya sebaris dengan selesainya suatu aksi dan bisa ditandai dengan *return message*.

Kandungan utama dari suatu s*equence diagram* adalah *message*. *Message* menyatakan suatu komunikasi yang terjadi antara *lifeline* dalam suatu interaksi. Suatu *message* ditunjukkan oleh suatu anak panah dari satu *lifeline* ke *lifeline* lainnya. Kepala anak panah menunjuk ke penerima. Jenis-jenis *message* adalah

sebagai berikut :

1. *Asynchronous message*, dengan simbol anak panah dengan kepala panah berbentuk batang dan terbuka.
2. *Synchronous message*, dengan simbol anak panah dengan kepala anak panah berbentuk segitiga, terisi dan tertutup.

Reply dari suatu *message* atau *return message* yang merupakan nilai kembalian dari suatu panggilan ditunjukkan dengan anak panah putus-putus dengan kepala anak panah berbentuk batang dan terbuka. Kepala anak panah menuju ke *lifeline* yang melakukan panggilan. Dalam *sequence diagram* dapat juga terjadi *self-call* dimana suatu obyek memanggil metode yang ada dalam dirinya sendiri.



Gambar 2. 15 : *Sequence Diagram*
Sumber : [BOR-05]

2.2.7.5. Pengujian Berorientasi Obyek

Saat proses perancangan perangkat lunak telah selesai maka perangkat lunak harus diuji untuk mengetahui dan memperbaiki sebanyak mungkin kesalahan. Pengujian adalah elemen yang sangat penting untuk menjamin kualitas perangkat lunak. Tujuan dari pengujian perangkat lunak adalah untuk menemukan kesalahan dan pengujian yang baik adalah satu pengujian yang memiliki kemungkinan paling besar untuk menemukan kesalahan dengan usaha dan waktu

paling minimum [PRE-05].

Proses pengujian dilakukan untuk menguji logika internal dan antarmuka dari tiap komponen perangkat lunak serta menguji domain masukan dan keluaran dari program untuk mencari kesalahan dalam fungsi, *behaviour* dan performansi program.

2.2.7.5.1. Strategi Pengujian

Strategi untuk melakukan pengujian perangkat lunak, dimulai dari dengan "pengujian kecil" bergerak menuju ke "pengujian besar". Demikian juga dalam konteks pengujian berorientasi objek, dalam hal ini pengujian dimulai dari pengujian unit, bergerak menuju pengujian integrasi dan berakhir pada proses validasi.

1. Pengujian Unit

Pada saat perangkat lunak berorientasi objek diperhatikan, konsep mengenai unit menjadi berubah. Enkapsulasi mengendalikan definisi kelas dan objek. Ini berarti bahwa masing-masing kelas dan contoh suatu kelas (objek) mengemas (data) dan operasi yang memanipulasi data-data tersebut. Selain modul individual, unit terkecil yang dapat diuji merupakan data atau objek enkapsulasi. Kelas dapat berisi sejumlah operasi yang berbeda, dan operasi khusus dapat muncul sebagai bagian dari kelas-kelas yang berbeda.

Kelas pengujian untuk menguji perangkat lunak beorientasi obyek ekivalen dengan pengujian unit untuk perangkat lunak konvesional. Tidak seperti pengujian unit perangkat lunak konvesional, yang cenderung berfokus pada detail algoritma dari suatu modul dan data yang mengalir pada interface modul, pengujian kelas untuk perangkat lunak berorientasi obyek dikendalikan oleh operasi yang dienkapsulasi oleh kelas dan keadaan tingkah laku dari kelas tersebut.

2. Pengujian Integrasi

Ada dua strategi yang berbeda untuk pengujian integrasi dari sistem berorientasi objek, pertama, pengujian *thread-based* yang mengintegrasikan

himpunan kelas yang dibutuhkan untuk merespon ke satu masukan atau bahkan untuk sistem. Masing-masing *thread* diintegrasikan dan diuji secara individual. Pengujian regresi diaplikasikan untuk memastikan bahwa tidak terjadi efek samping. Pendekatan integrasi kedua, pengujian *use-based*, memulai konstruksi sistem dengan menguji kelas-kelas tersebut (disebut kelas independen) yang menggunakan sangat sedikit (atau kalau ada) kelas-kelas server. Setelah kelas-kelas independen diuji, lapisan kelas selanjutnya diuji, yaitu kelas dependen yang menggunakan kelas independen. Urutan lapisan pengujian kelas dependen ini berlanjut sampai keseluruhan sistem dibangun.

Cluster testing adalah salah satu langkah di dalam pengujian integrasi perangkat lunak beorientasi obyek. Disini *cluster* kelas yang berkolaborasi (ditentukan dengan menguji CRC dan model hubungan objek) digunakan untuk mendesain kasus uji yang digunakan untuk mengungkap kesalahan di dalam kolaborasi.

3. Pengujian Validasi

Pada tingkat sistem atau validasi, detail sambungan kelas hilang. Seperti validasi konvesional, validasi perangkat lunak beorientasi obyek difokuskan pada aksi yang dapat dilihat oleh pemakai dan keluaran yang dapat dikenali oleh pemakai sistem tersebut. Untuk membantu derifasi pengujian validasi, penguji harus menggunakan *use case*. Yang merupakan bagian dari model analisis. *Use case* menyediakan skenario yang kemungkinan besar mengungkap kesalahan di dalam persyaratan interaksi pemakai. Metode pengujian *black box* konvesional dapat digunakan untuk mengendalikan pengujian validasi.

BAB III
METODOLOGI PENELITIAN

Pada bab ini dijelaskan mengenai langkah-langkah yang akan dilakukan dalam perancangan, implementasi dan pengujian dari aplikasi perangkat lunak yang akan dibuat. Kesimpulan dan saran disertakan sebagai catatan atas aplikasi dan bagaimana aplikasi dapat dikembangkan lebih lanjut.

3.1. Studi Literatur

Studi literatur menjelaskan dasar teori yang digunakan untuk menunjang penulisan skripsi. Teori-teori tersebut meliputi:

1. Pengertian dan konsep dasar *Web Service* sebagai sebagai dasar sistem yang akan diterapkan pada aplikasi *Web Service E-Commerce*.
2. Pengertian dan konsep dasar *E-Commerce* sebagai sebagai dasar sistem yang akan diterapkan pada aplikasi *Web Service E-Commerce*.
3. Pengertian dan konsep dasar pemrograman *web* yang meliputi struktur dan pemrograman PHP dan HTML.
4. Pengertian dan konsep dasar pemrograman *web service* yang meliputi bahasa pemrograman XML, SOAP dan WSDL.
5. Pengertian dan konsep dasar basis data sebagai dasar pembangunan basis data untuk aplikasi *Web Service E-Commerce*.

3.2. Studi Lapangan

Studi Lapangan bertujuan untuk memperoleh data dengan jalan mengadakan tanya jawab atau pengamatan secara langsung. Data yang didapat pada studi lapangan akan digunakan dalam pembuatan aplikasi. Data yang diperoleh meliputi :

Data alur dan proses – proses transaksi *E-Commerce*.

Data mengenai barang – barang yang dijual.

3.3. Perancangan

Perancangan didasarkan pada teori-teori yang ada sehingga dapat diaplikasikan pada perangkat lunak yang dibuat yaitu Aplikasi *Web Service E-Commerce*.

Perancangan aplikasi dilakukan untuk mempermudah implementasi, analisis algoritma dan pengujian. Perancangan aplikasi berdasarkan *Object Oriented Analysis* dan *Object Oriented Design* yaitu menggunakan pemodelan UML (*Unified Modeling Language*). Pada tahap ini dibuat suatu diagram pemodelan sistem aplikasi secara keseluruhan yang sesuai dengan analisis kebutuhan sistem yang digambarkan dalam *use case diagram*. Perancangan dilakukan dengan membuat *class diagram* dan *sequence diagram* yang memperlihatkan interaksi pengguna dengan objek-objek dalam aplikasi dan diagram kelas untuk melihat hubungan antar kelas, objek dan *interface* yang ada di dalam aplikasi.

3.4. Implementasi

Implementasi aplikasi dilakukan dengan mengacu kepada perancangan aplikasi. Implementasi perangkat lunak dilakukan dengan menggunakan bahasa pemrograman berorientasi objek yaitu menggunakan bahasa pemrograman PHP 5, SOAP (*Simple Object Access Protocol*), dan WSDL (*Web Service Describe Language*).

3.5. Pengujian dan Analisis

Pengujian aplikasi yaitu untuk mengetahui kesesuaian analisa kebutuhan yang dibuat dengan implementasi aplikasi. Analisis sistem dilakukan dengan membandingkan sistem aplikasi yang dibuat dengan teori yang ada sehingga didapatkan suatu kesimpulan.

Proses pengujian dilakukan tahapan - tahapan yaitu pengujian unit, pengujian integrasi, pengujian validasi, pengujian perancangan basis data, dan pengujian performansi koneksi *web service*. Proses analisis dilakukan dengan membandingkan sistem aplikasi dengan teori yang ada sehingga didapatkan suatu kesimpulan.

3.6. Pengambilan Kesimpulan dan Saran

Pengambilan kesimpulan dilakukan setelah semua tahapan perancangan, implementasi dan pengujian sistem aplikasi telah selesai dilakukan dan didasarkan pada kesesuaian antara teori dan praktek. Kesimpulan diambil untuk menjawab rumusan masalah yang telah ditetapkan sebelumnya. Tahap terakhir dari penulisan adalah saran yang dimaksudkan untuk memperbaiki kesalahan-kesalahan yang terjadi dan menyempurnakan penulisan serta untuk memberikan pertimbangan atas pengembangan aplikasi selanjutnya.

BAB IV
PERANCANGAN

Bab ini menjelaskan tentang perancangan aplikasi *Web Service* pada *E-Commerce* dengan menggunakan pendekatan berorientasi obyek. Perancangan yang dilakukan meliputi dua tahap yaitu analisis berorientasi obyek dan desain berorientasi obyek. Pada tahap analisis berorientasi obyek, analisa yang dilakukan berupa analisis terhadap sistem eksisting dan analisis kebutuhan sistem yang akan dirancang.

Tahap desain berorientasi obyek terdiri dari perancangan umum dan perancangan detail. Perancangan umum menggambarkan arsitektur sistem dan blok diagram. Perancangan detail terdiri dari perancangan *class diagram*, *sequence diagram* sebagai pemodelan perangkat lunak, perancangan basis data, dan perancangan *user interface*.

4.1. Analisis Kebutuhan

Analisis kebutuhan perangkat lunak adalah aktifitas rekayasa perangkat lunak yang menjembatani antara kebutuhan ditingkat sistem yang sudah ada (eksisting) dengan yang dirancang pada perancangan perangkat lunak. Analisis kebutuhan perangkat lunak dapat juga diartikan proses yang digunakan untuk mendapatkan, menganalisis, dan memvalidasi kebutuhan-kebutuhan sistem.

Analisis kebutuhan perangkat lunak menggunakan bahasa pemodelan UML. Tahap analisis kebutuhan menggunakan pemodelan *use case diagram* beserta *use case specification*-nya.

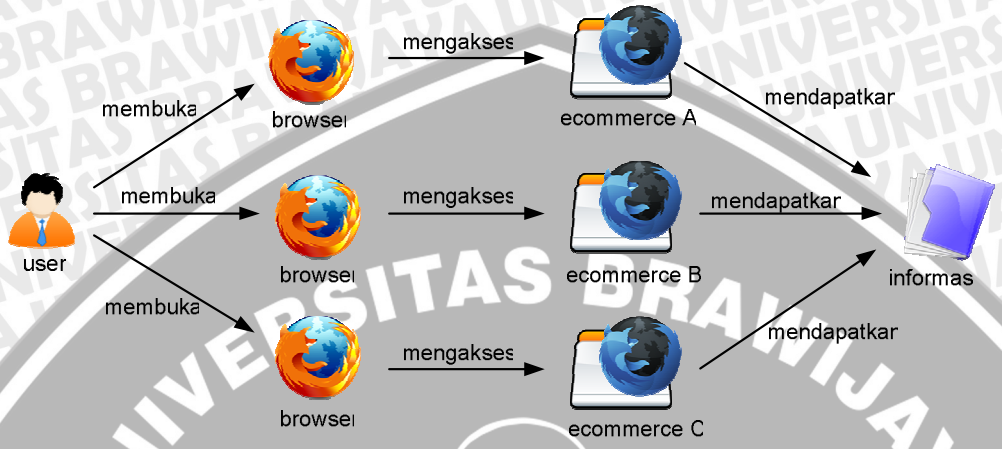
4.1.1. Analisis Sistem Eksisting

Analisa sistem eksisting merupakan analisa kebutuhan tentang sistem yang telah ada. Analisa sistem eksisting pada skripsi ini didasarkan pada 2 hal, yakni dari segi kebutuhan pengguna dan dari segi sistemnya.

4.1.1.1. Analisa Sistem Eksisting berdasarkan Penggunanya

Apabila masyarakat (konsumen) ingin melakukan transaksi secara *online*, proses yang dilakukan adalah mencari informasi tersebut dari beberapa *E-*

Commerce yang ada dengan membuka *browser* (IE, mozilla, opera, dan sebagainya) kemudian mengakses masing - masing situs tersebut. Sehingga konsumen harus melompat dari satu situs *E-Commerce* yang satu ke situs *E-Commerce* yang lain. Proses tersebut dapat dilihat pada gambar 4.1 dibawah ini.

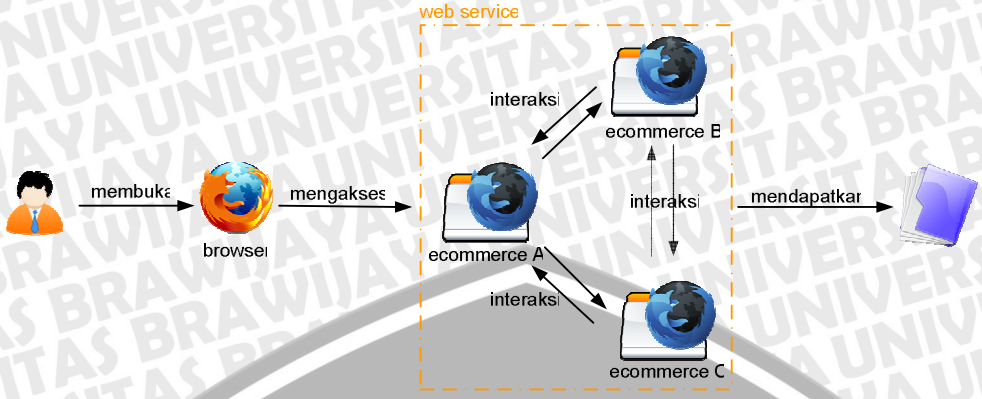


Gambar 4. 1 : Proses pencarian informasi pada *E-Commerce*
Sumber: [Analisis]

Proses pada gambar 4.1 diatas memiliki kelebihan dan kekurangan. Bagi masyarakat yang baru mengenal internet, hal tersebut merupakan suatu hal yang menantang karena dapat menjelah dari satu situs ke situs yang lain. Dimana eksplorasi internet membawa banyak pengalaman baru. Dan istilah banjir informasi mungkin tidak ada.

Namun, lain halnya bagi masyarakat yang sudah sering memanfaatkan internet untuk mendapatkan informasi. Banjir informasi dapat menjadi sesuatu yang merepotkan karena harus memilah – milah informasi. Situasi ini semakin merepotkan karena harus berlompatan dari situs satu ke situs lainnya yang menjadikan semakin tidak efisien dan praktis dalam medapatkan informasi.

Proses pencarian informasi setelah memanfaatkan *web service* akan lebih cepat dan efisien, karena apabila masyarakat (konsumen) ingin mendapatkan informasi yang dibutuhkan, misalnya barang/jasa yang akan dibeli secara *online*, maka cukup membuka satu situs yang memuat potongan – potongan informasi dari berbagai situs *E-Commerce* lainnya yang sejenis. Proses tersebut dapat dilihat pada gambar 4.3 dibawah ini.



Gambar 4. 2 : Proses pencarian informasi setelah memanfaatkan *Web Service*
Sumber: [Analisis]

Dari segi perusahaan pemilik *E-Commerce* yang memiliki banyak *partner*, hal tersebut memberikan keuntungan karena teknologi *web service* memungkinkan untuk integrasi aplikasi yang berbeda *platform* sekalipun. Sehingga si *partner* tidak perlu membangun aplikasi - aplikasi sejenis, akan tetapi cukup menggunakan modul – modul yang telah disediakan oleh *Web Service E-Commerce* yang kemudian diletakkan pada situs *partner*.

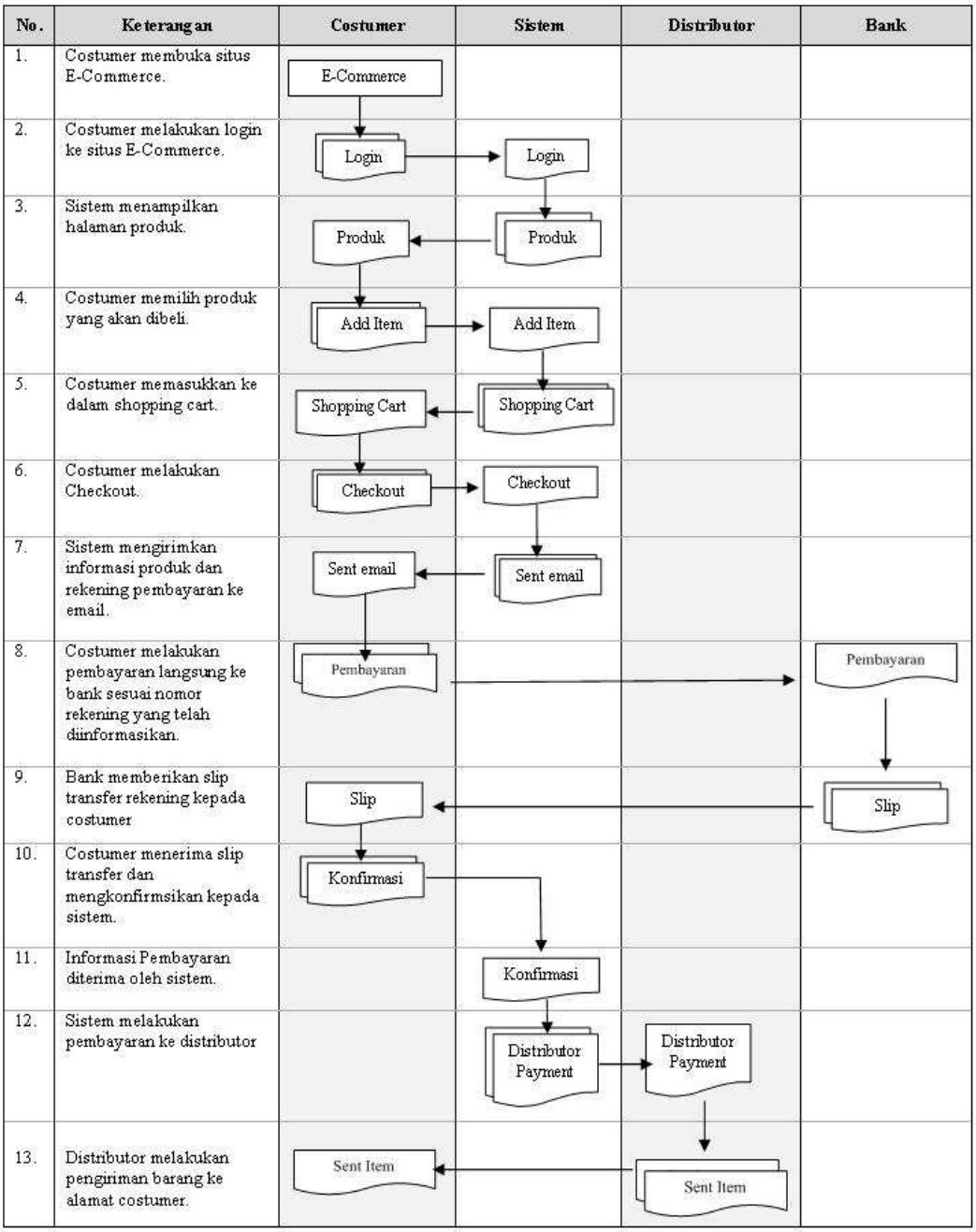
4.1.1.2. Analisa Sistem Eksisting berdasarkan Sistem

Saat ini proses transaksi *online* yang digunakan pada aplikasi *E-Commerce* di Indonesia masih bersifat konvensional. Dimana proses pembayarannya masih melalui transfer antar rekening dan secara tidak langsung masih bermodalkan kepercayaan antara pembeli dan penjual. Beberapa kendala pembayaran *online* di Indonesia adalah :

1. Beberapa *payment gateway* belum *support* terhadap sebagian besar bank yang ada di Indonesia. Hanya beberapa bank dan kartu kredit tertentu yang dapat digunakan untuk pembayaran *online*.
2. Beberapa bank yang sudah mendukung pembayaran *online* masih memiliki kendala dalam hal *withdraw* (penarikan uang). Sehingga hanya dapat melakukan transaksi sepihak saja, yakni pembelian produk.
3. Banyaknya kriminalitas internet khususnya di Indonesia, seperti *card froud* (pencurian akses kartu kredit). Hal ini membuat konsumen merasa tidak nyaman menggunakan kartu kredit.
4. Budaya orang Indonesia tentang kartu kredit belum begitu populer. Hanya

masyarakat golongan ekonomi menengah keatas saja yang menggunakan fasilitas kartu kredit.

Proses transaksi yang terjadi pada *E-Commerce* yang bersifat konvensional tersebut digambarkan gambar 4.3.



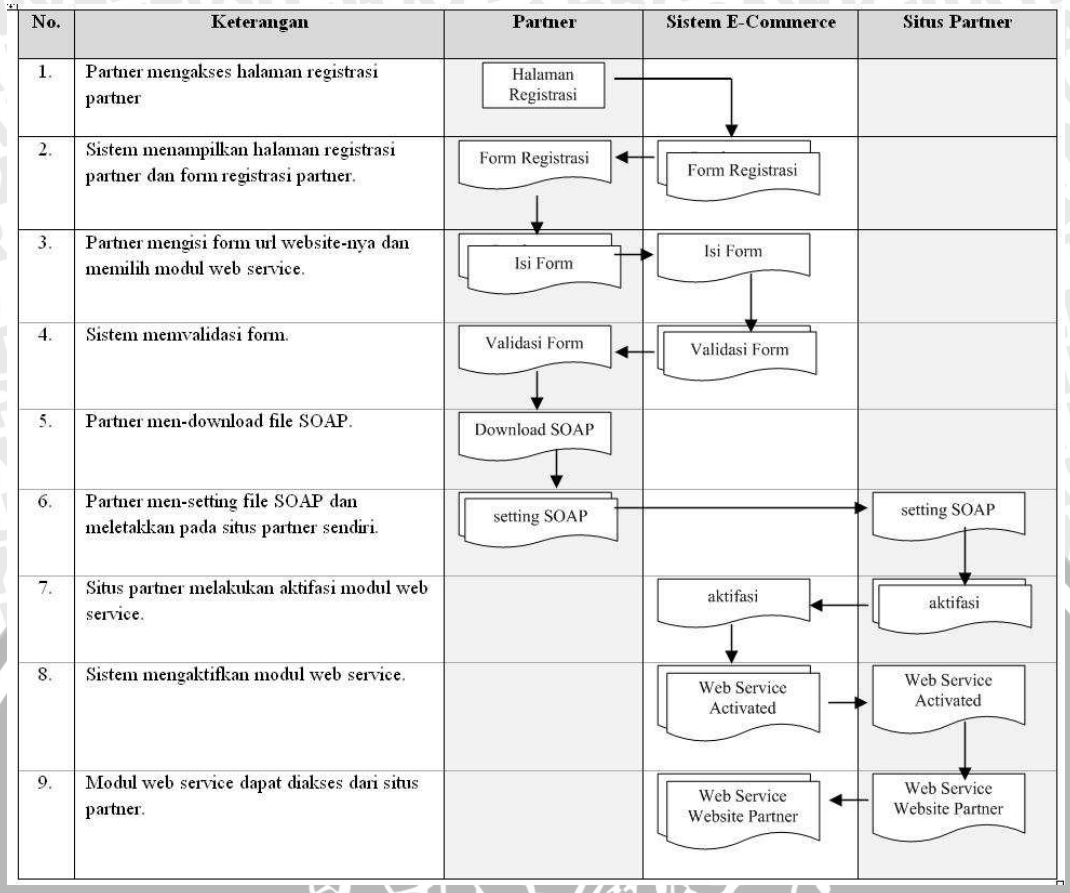
Gambar 4. 3 : *Standar Operating Procedure E-Commerce*
Sumber : [Analisis]

Proses pada gambar 4.3 adalah sebagai berikut :

1. *Costumer* membuka situs *E-Commerce*.
2. *Costumer* melakukan login ke situs *E-Commerce*.
3. Sistem menampilkan halaman produk.
4. *Costumer* memilih produk yang akan dibeli.
5. *Costumer* memasukkan ke dalam *shopping cart*.
6. *Costumer* melakukan *Checkout*.
7. Sistem mengirimkan informasi produk dan rekening pembayaran ke *email*.
8. *Costumer* melakukan pembayaran langsung ke bank sesuai nomor rekening yang telah diinformasikan.
9. Bank memberikan slip transfer rekening kepada *costumer*.
10. *Costumer* menerima slip transfer dan mengkonfirmsikan kepada sistem.
11. Informasi pembayaran diterima oleh sistem.
12. Sistem melakukan pembayaran ke distributor.
13. Distributor melakukan pengiriman barang ke alamat costumer.

4.1.2. Analisis Sistem yang akan Dirancang

Analisa sistem yang akan dirancang merupakan proses kelanjutan setelah melakukan analisa sistem eksisting. Berdasarkan analisa sistem eksisting tersebut sistem *E-Commerce* dibuat dengan mengacu pada SOP *E-Commerce* pada gambar 4.3. Sedangkan sistem *Web Service* yang akan digambarkan pada SOP dalam gambar 4.4.



Gambar 4. 4 : *Standar Operating Procedure Web Service*
Sumber : [Analisis]

Tahap analisis kebutuhan dilakukan dengan memodelkan kebutuhan ke dalam *use case diagram* berdasakan SOP *E-Commerce* pada gambar 4.3 dan SOP *Web Service* pada gambar 4.4. *Use case diagram* bertujuan untuk menggambarkan kebutuhan-kebutuhan fungsional yang harus disediakan oleh *Aplikasi Web Service E-Commerce* agar dapat memecahkan permasalahan yang dihadapi oleh *user*.

4.1.2.1. Daftar Kebutuhan

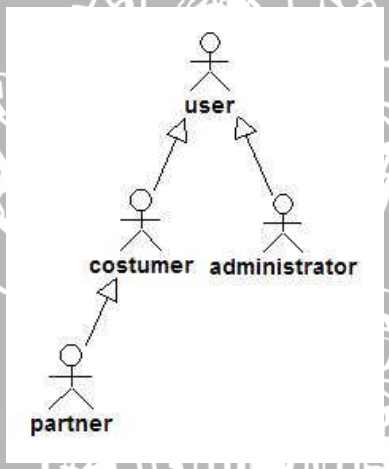
Daftar kebutuhan merupakan daftar yang menguraikan kebutuhan-kebutuhan *user* yang harus disediakan oleh perangkat lunak baik kebutuhan fungsional maupun non fungsional. Proses yang dilakukan sebelum menentukan daftar kebutuhan adalah mengidentifikasi aktor yang menggunakan sistem Aplikasi *Web Service E-Commerce*. Tabel 4.1 memperlihatkan daftar aktor beserta penjelasannya yang merupakan hasil dari proses identifikasi aktor.

Tabel 4. 1 : Deskripsi Aktor

No.	Aktor	Keterangan
1.	User	Aktor yang mengakses aplikasi dan belum melakukan login.
2.	Administrator	Aktor yang mengelolah sistem termasuk manajemen data dan manajemen user.
3.	Customer	Aktor yang melakukan proses pembelian barang.
4.	Partner	Aktor yang menggunakan fasilitas *web service ecommerce*.

Sumber : [Analisis]

Aktor *customer* dan *partner* merupakan aktor yang mendaftarkan pada Aplikasi *Web Service E-Commerce*. Sedangkan aktor *administrator* merupakan aktor yang melakukan segala proses administrasi pada aplikasi ini. Generalisasi dari para aktor Aplikasi *Web Service E-Commerce* ini ditunjukkan dengan gambar 4.5.



Gambar 4. 5 : Generalisasi Aktor - Aktor Aplikasi *Web Service E-Commerce*
Sumber : [Analisis]

Penyusunan daftar kebutuhan fungsional dari aplikasi ini dilakukan setelah identifikasi aktor. Daftar kebutuhan fungsional disertai dengan nama *use case* yang mempresentasikan fungsionalitas dari kebutuhan tersebut. Daftar kebutuhan fungsional tersebut ditunjukkan pada Tabel 4.2.

Tabel 4. 2 : Daftar Kebutuhan Fungsional Aplikasi *Web Service E-Commerce*

No.	Kebutuhan	*Use Case*
1.	Sistem harus menyediakan fasilitas registrasi, sehingga setiap pengguna yang akan berinteraksi dengan aplikasi telah terdaftar sebagai anggota.	Registrasi
2.	Sistem harus menyediakan fasilitas *login*, sehingga hanya pengguna yang terdaftar sebagai anggota saja yang dapat mengakses fasilitas tertentu. Pengguna yang tidak melakukan *login* dianggap sebagai aktor *user*.	*Login*
3.	Sistem harus menyediakan fasilitas *logout* bagi pengguna, sehingga pengguna yang telah *login* dapat keluar dari aplikasi.	*Logout*
4.	Sistem harus menyediakan fasilitas untuk mencari informasi pada katalog barang yang terdapat pada sistem.	Mencari Data Barang
5.	Sistem harus menyediakan fasilitas untuk melihat data katalog barang. Data tersebut berupa daftar – daftar nama barang, harga barang, dan sekilas deskripsi tiap – tiap barang.	Melihat Data Katalog Barang
6.	Sistem harus menyediakan fasilitas untuk melihat detail barang. Data tersebut berupa nama barang, harga barang, status barang/stok barang, dan deskripsi barang.	Melihat Data Detail Barang
7.	Sistem harus menyediakan fasilitas untuk menambah barang. Data yang harus disimpan dari proses penambahan data ini adalah nama barang, kategori barang, harga barang, berat barang, gambar barang, dan deskripksi barang. Fasilitas ini hanya dapat digunakan oleh *administrator*.	Menambah Barang
8.	Sistem harus menyediakan fasilitas untuk mengedit stok barang. Dimana fasilitas ini akan menambah/mengurangi jumlah stok barang yang telah ada pada sistem. Fasilitas ini hanya dapat diakses oleh *administrator*,	Mengedit Stok Barang
9.	Sistem harus menyediakan fasilitas untuk mengedit data barang. Data yang dapat dirubah dari proses ini adalah nama barang, kategori barang, harga barang, berat barang, dan deskripsi barang. Fasilitas ini hanya dapat digunakan oleh *administrator*.	Mengedit Data Barang
10.	Sistem harus menyediakan fasilitas untuk mengedit gambar barang. Fasilitas ini hanya dapat digunakan oleh *administrator*.	Mengedit Gambar Barang
11.	Sistem harus menyediakan fasilitas untuk menghapus data barang. Fasilitas ini hanya dapat digunakan oleh *administrator*.	Menghapus Data Barang
12.	Sistem harus dapat menyediakan fasilitas untuk melihat *shopping cart*. Dimana *customer* dapat melihat *item* yang dibeli dan disimpan di *shopping cart* (keranjang belanja) berupa nama barang, jumlah barang, harga barang per-*item*, dan harga total barang.	Melihat *Shopping Cart*
13.	Sistem harus menyediakan fasilitas untuk menambah *item shopping cart*. Data yang dapat disimpan di *shopping cart* ini berupa nama barang, jumlah barang yang dibeli, harga barang per-*item*, dan harga total barang.	Menambah *Item Shopping Cart*

14.	Sistem harus menyediakan fasilitas utuk edit *Item Shopping Cart*. Dimana data yang dapat dirubah adalah jumlah *item* yang dipesan.	Edit *Item Shopping Cart*
15.	Sistem harus menyediakan fasilitas untuk menghapus item *shopping cart*.	Menghapus *Item Shopping Cart*
16.	Sistem harus menyediakan fasilitas untuk memilih pembayaran untuk pembelian barang yang telah dipesan oleh *customer*. Dimana *customer* dapat mnggunakan metode pembayaran melalui kartu kredit atau melalui rekening bank.	Memilih Pembayaran
17.	Sistem harus menyediakan fasilitas untuk mengisi kartu kredit untuk *customer* yang menggunakan jasa kartu kredit dalam pembelian barang. Dimana *customer* dapat mengisi nomor kartu kredit dan waktu expired kartu kredit.	Mengisi Data Kartu Kredit
18.	Sistem harus menyediakan fasilitas untuk memilih jasa kirim untuk pengiriman barang yang telah dipesan oleh *customer*. Dimana *customer* dapat melihat informasi pengiriman barang berupa nama jasa kirim dan harga jasa kirim.	Memilih Jasa Kirim
19.	Sistem harus menyediakan fasilitas untuk *check out shopping cart*. Dimana fasilitas ini untuk menyimpan data yang telah dipesan oleh *customer* kedalam sistem berupa nama barang, jumlah barang, harga barang per-*item*, harga total barang, jenis pengiriman barang, biaya pengiriman barang, dan nomor rekening untuk proses pembayaran, serta batas waktu pembayaran.	*Checkout Shopping Cart*
20.	Sistem harus menyediakan fasilitas untuk konfirmasi *email*. Dimana informasi tentang barang yang dipesan oleh *customer* dikirimkan ke alamat *email customer* tersebut.	Konfirmasi *Email*
21.	Sistem harus menyediakan fasilitas untuk melihat daftar *order* barang. Daftar *order* barang yang dapat dilihat berupa status pembayaran, tanggal pemesanan, nama pemesan, dan total pembelian barang. Fasilitas ini hanya dapat diakses oleh *administrator*.	Melihat Daftar *Order* Barang
22.	Sistem harus menyediakan fasilitas untuk melihat detail *order* barang. Data tersebut berupa status pembayaran, tanggal pemesanan, nama pemesan, nama barang yang dipesan, jenis jasa kirim, harga kirim, dan total pembelian. Fasilitas ini hanya dapat diakses oleh *administrator*.	Melihat Detail *Order* Barang
23.	Sistem harus menyediakan fasilitas untuk mengedit status *order* barang. Dimana fasilitas ini dapat diakses oleh *administrator*.	Mengedit Status *Order* Barang
24.	Sistem harus menyediakan fasilitas untuk menghapus *order* barang. Fasilitas ini hanya dapat diakses oleh *administrator*.	Menghapus *Order* Barang
25.	Sistem harus menyediakan fasilitas untuk melihat daftar berita aplikasi *e-commerce*. Data yang ditampilkan berupa judul berita, tanggal berita, dan sekilas isi berita.	Melihat Daftar Berita
26.	Sistem harus menyediakan fasilitas untuk melihat detail berita. Data yang ditampilkan berupa judul berita, tanggal berita, isi detail berita, dan penulis berita.	Melihat Detail Berita

27.	Sistem harus menyediakan fasilitas untuk menambah berita. Data yang disimpan pada proses ini berupa judul berita, tanggal berita, dan isi berita. Fasilitas ini hanya dapat diakses oleh *administrator*.	Menambah Berita
28.	Sistem harus menyediakan fasilitas untuk mengedit berita seputar musik. Data yang dirubah pada proses ini berupa judul berita, dan isi berita. Fasilitas ini hanya dapat diakses oleh *administrator*.	Mengedit Berita
29.	Sistem harus menyediakan fasilitas untuk menghapus berita. Fasilitas ini hanya dapat diakses oleh *administrator*.	Menghapus Berita
30.	Sistem harus menyediakan fasilitas untuk melihat daftar konten *website*. Fasilitas ini hanya dapat diakses oleh *administrator*. Dimana konten *website* ini berupa *site map*, *site credit*, kontak, dan cara *order*.	Melihat Daftar Konten *Website*
31.	Sistem harus menyediakan fasilitas untuk melihat detail konten *website*. Dimana fasilitas ini dapat diakses oleh *user* di halaman *website*, dan diakses oleh *administrator* di halaman administrasi website.	Melihat Detail Konten *Website*
32.	Sistem harus menyediakan fasilitas untuk mengedit konten *website*. Fasilitas ini hanya dapat digunakan oleh *administrator*.	Mengedit Konten *Website*
33.	Sistem harus menyediakan fasilitas untuk melihat daftar *user* yang terdaftar pada sistem. Data yang dapat dilihat berupa username, kategori *user*, dan *email user*. Proses ini hanya dapat dilakukan oleh *administrator*.	Melihat Daftar *User*
34.	Sistem harus menyediakan fasilitas untuk melihat detail *user*. Dimana data *user* yang dapat dilihat berupa *username*, kategori *user*, nama lengkap, alamat *user*, email *user*, propinsi, kota, kode pos, dan nomor telepon *user*. Untuk *partner* terdapat tambahan tentang informasi *website partner* dan kode aktifasi *web service*.	Melihat Data Profil *User*
35.	Sistem harus menyediakan fasilitas untuk mengedit data profil *user*. Proses ini hanya dapat dilakukan oleh *user* itu sendiri. Data yang dapat dirubah berupa *username*, nama lengkap, alamat *user*, email *user*, propinsi, kota, kode pos, dan nomor telepon *user*.	Mengedit Data Profil *User*
36.	Sistem harus menyediakan fasilitas untuk mengedit *password user*. Proses ini hanya dapat dilakukan oleh *user* itu sendiri.	Mengedit *Password User*
37.	Sistem harus menyediakan fasilitas untuk mengapus *user*. Fasilitas ini hanya dapat dilakukan oleh *administrator*.	Menghapus *User*
38.	Sistem harus menyediakan fasilitas untuk registrasi *website*. Dimana yang didaftarkan oleh *partner* merupakan URL *website* milik *partner* dan *web service* yang akan digunakan pada situsnya.	Registrasi *Website Partner*
39.	Sistem harus menyediakan fasilitas untuk *download* SOAP *file*. Dimana *file* tersebut merupakan SOAP *client* untuk mengakses *web service*.	*Download* SOAP *file*
40.	Sistem harus menyediakan fasilitas untuk melihat daftar *parter web service*. Dimana data tersebut berupa *website partner* dan *username partner*. Fasilitas ini hanya dapat diakses oleh *administrator*.	Melihat Daftar *Partner Web Service*

41.	Sistem harus menyediakan fasilitas untuk melihat detail *partner web service*. Dimana data yang dapat dilihat berupa *website partner, username partner*, tanggal pendaftaran *partner*, dan jenis *web service*. Fasilitas ini hanya dapat diakses oleh *administrator*.	Melihat Detail *Partner Web Service*
42.	Sistem harus menyediakan fasilitas menghapus *website partner*. Fasilitas ini hanya dapat diakses oleh *administrator*.	Menghapus *Partner Web Service*
43.	Sistem harus menyediakan fasilitas untuk melihat daftar kategori *Web Service*. Dimana data yang dilihat berupa nama *web service* dan sekilas deskripsi *web service*.	Melihat Daftar Kategori *Web Service*
44.	Sistem harus menyediakan fasiltas untuk melihat detail kategori *web service*. Dimana data tersebut berupa nama *web service* dan detail deskripsi *web service*.	Melihat Detail Katagori *Web Service*
45.	Sistem harus menyediakan fasilitas untuk mengedit kategori *web service*. Data yang dapat dirubah berupa nama *web service* dan deskripsi *web service*. Fasilitas ini hanya dapat diakses oleh *administrator*.	Mengedit Kategori *Web Service*

Sumber : [Analisis]

Keseluruhan kebutuhan fungsionalitas di atas dibagi menjadi 4 modul untuk lebih mempermudah pemahaman dan pendesainan sistem. Kelima modul tersebut adalah sebagai berikut :

1. Modul Pendukung Sistem.
2. Modul Katalog Barang.
3. Modul Data *Website*.
4. Modul *Web Service*

Tabel 4.3 berikut menyatakan pembagian modul-modul dan *use case* yang termasuk dalam modul tersebut :

Tabel 4. 3 : Modul dan Kebutuhan Fungsionalitas

No.	Modul	Kebutuhan Fungsionalitas
1.	Modul Pendukung Sistem	1. Registrasi 2. *Login* 3. *Logout* 4. Melihat Daftar *User* 5. Melihat Data Profil *User* 6. Mengedit Data Profil *User* 7. Mengedit *Password User* 8. Menghapus *User*
2.	Modul Katalog Barang	1. Mencari Data Barang 2. Melihat Data Katalog Barang 3. Melihat Data Detail Barang 4. Menambah Barang 5. Mengedit Stok Barang 6. Mengedit Data Barang 7. Mengedit Gambar Barang

		8. Menghapus Data Barang 9. Melihat *Shopping Cart* 10. Menambah *Item Shopping Cart* 11. Edit *Item Shopping Cart* 12. Menghapus *Item Shopping Cart* 13. Memilih Pembayaran 14. Memasukkan Data Kartu Kredit 15. Memilih Jasa Kirim 16. *Checkout Shopping Cart* 17. Konfirmasi *Email* 18. Melihat Daftar *Order* Barang 19. Melihat Detail *Order* Barang 20. Mengedit Status *Order* Barang 21. Menghapus *Order* Barang
3.	Modul Data *Website*	1. Melihat Daftar Berita 2. Melihat Detail Berita 3. Menambah Berita 4. Mengedit Berita 5. Menghapus Berita 6. Melihat Daftar Konten *Website* 7. Melihat Detail Konten *Website* 8. Mengedit Konten *Website*
4.	Modul *Web Service*	1. Registrasi *Website Partner* 2. Download SOAP *file* 3. Melihat Daftar *Partner Web Service* 4. Melihat Detail *Partner Web Service* 5. Menghapus *Partner Web Service* 6. Melihat Daftar Kategori *Web Service* 7. Melihat Detail Kategori *Web Service* 8. Mengedit Kategori *Web Service*

Sumber : [Analisis]

Daftar kebutuhan non fungsional sistem informasi Aplikasi *Web Service E-Commerce* ditunjukkan pada Tabel 4.4.

Tabel 4. 4 : Daftar Kebutuhan Non Fungsional Aplikasi *Web Service E-Commerce*

No.	Kebutuhan Fungsionalitas
1.	Aplikasi dikembangkan dengan menggunakan bahasa pemrograman XML *Web Service* (SOAP dan WSDL), PHP dan basis data MySQL.
2.	Aplikasi harus dapat diakses melalui *web browser*.

Sumber: [Analisis]

4.1.2.2. *Use Case Diagram*

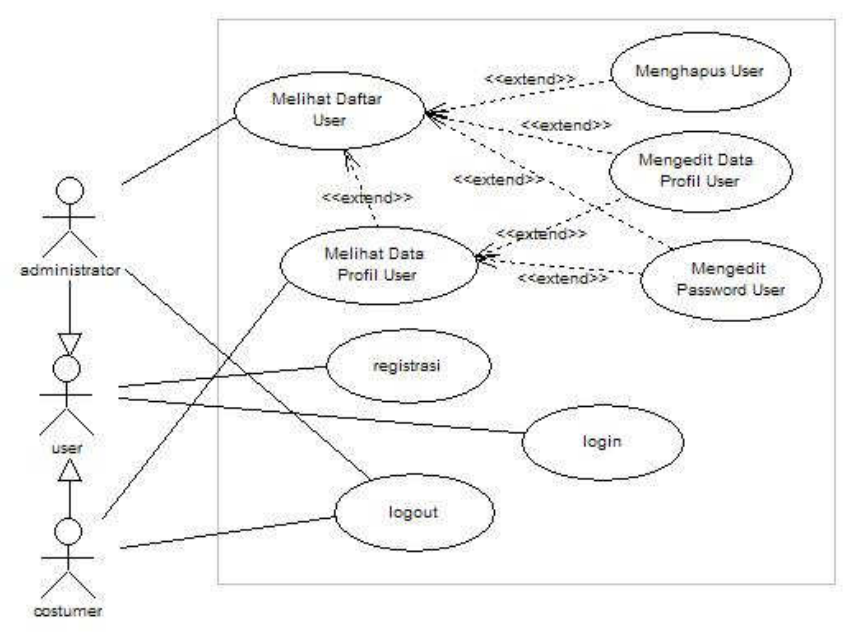
Use case diagram merupakan salah satu diagram untuk memodelkan aspek perilaku sistem. Masing-masing *use case diagram* menunjukkan sekumpulan *use case*, aktor dan hubungannya.. *Use case* merupakan fungsionalitas dari sistem yang diinisiasi oleh aktor. Aktor adalah sebuah entitas-entitas luar yang berkomunikasi dengan sistem.

Pemodelan dalam *use case diagram* yang menggambarkan fungsionalitas yang disediakan oleh aplikasi *Web Service E-Commerce* dibagi menjadi enam buah diagram yang bersesuaian dengan modul dalam aplikasi *Web Service E-Commerce*.

4.1.2.2.1. *Use Case Diagram* untuk Modul Pendukung Sistem

Use case diagram untuk modul pendukung sistem melibatkan 3 buah aktor dan 8 buah *use case*. Aktor – aktor yang terlibat pada modul pendukung sistem aplikasi *Web Service E-commerce* adalah :

1. *User*
2. *Administrator*
3. *Customer*



Gambar 4. 6 : *Use Case Diagram* untuk Modul Pendukung Sistem
Sumber : [Analisis]

1. *Use Case Spesification* registrasi

Tabel 4. 5 : *Use case specification* registrasi

Nama *use case*	Registrasi
Aktor	*User*
Deskripsi	Sistem harus menyediakan fasilitas registrasi, sehingga setiap pengguna yang akan berinteraksi dengan aplikasi telah terdaftar sebagai anggota.

Pra-kondisi	*User* mengakses halaman utama *website*.
Pasca-kondisi	*User* telah terdaftar pada sistem sebagai *customer*.

Aliran Utama

Aksi dari Aktor	**Tanggapan dari Sistem**
1. *User* menekan tombol registrasi pada halaman *website*	2. Sistem menampilkan *form* masukan berupa *field username*, *password*, konfirmasi *password*, nama lengkap, *email*, alamat, propinsi, kota, kode pos, nomor telepon, dan kode keamanan.
3. *User* memasukkan data *username*, *password*, *email*, nama lengkap, alamat, nomor telepon, dan kode keamanan.	4. Sistem memvalidasi bahwa data *username*, *password*, nama lengkap, *email*, alamat, propinsi, kota, kode pos, atau nomor telepon tidak harus diisi, *username*, *password*, nama lengkap, *email*, alamat, kode pos, atau nomor telepon memiliki karakter yang diijinkan oleh sistem, *username* atau nama lengkap belum ada dalam *database* sistem, serta kode keamanan diisi dengan benar.
	5. Sistem menampilkan pesan bahwa *user* telah berhasil melakukan proses registrasi.

Aliran Alternatif 1: *username, password,* konfirmasi *password,* nama lengkap, *email,* alamat, propinsi, kota, kode pos. nomor telepon, atau kode keamanan kosong.

(*User* pada langkah nomer 3 aliran utama tidak mengisi *username*, *password*, konfirmasi *password*, nama lengkap, *email*, alamat, propinsi, kota, kode pos, nomor telepon, atau kode keamanan.)

Aksi dari Aktor	**Tanggapan dari Sistem**
	1. Menampilkan pesan bahwa *username*, *password*, konfirmasi *password*, nama lengkap, *email*, alamat, propinsi, kota, kode pos, nomor telepon, atau kode keamanan harus diisi.
	2. Kembali pada langkah nomer 2 aliran utama.

Aliran Alternatif 2: *username, password, email,* nama lengkap, alamat, kode pos, atau nomor telepon memiliki karakter yang tidak diijinkan oleh sistem.

(*User* pada langkah nomer 3 aliran utama memasukkan karakter yang tidak diijinkan oleh sistem *username*, *password*, *email*, nama lengkap, alamat, kode pos, dan nomor telepon.)

Aksi dari Aktor	Tanggapan dari Sistem
	1. Menampilkan pesan bahwa karakter *username*, *password*, *email*, nama lengkap, alamat, kode pos, atau nomor telepon tidak diperbolehkan.
	2. Kembali pada langkah nomer 2 aliran utama.
Aliran Alternatif 3: *username* atau nama lengkap telah ada dalam sistem. (*User* pada langkah nomer 3 aliran utama memasukkan *username* atau nama lengkap yang telah ada dalam sistem.)	
Aksi dari Aktor	**Tanggapan dari Sistem**
	1. Menampilkan pesan bahwa *username*, atau nama lengkap telah terdaftar.
	2. Kembali pada langkah nomer 2 aliran utama.
Aliran Alternatif 3: kode keamanan tidak sesuai. (*User* pada langkah nomer 3 aliran utama memasukkan kode keamanan tidak sesuai.)	
	1. Menampilkan pesan bahwa kode keamanan tidak sesuai.
	2. Kembali pada langkah nomer 2 aliran utama.

Sumber : [Analisis]

2. *Use Case Spesification Login*

Tabel 4. 6 : *Use case specification login*

Nama *use case*	Login
Aktor	*User*
Deskripsi	Sistem harus menyediakan fasilitas *login*, sehingga hanya pengguna yang terdaftar sebagai anggota saja yang dapat mengakses fasilitas tertentu. Pengguna yang tidak melakukan *login* dianggap sebagai aktor *user*.
Pra-kondisi	*User* mengakses halaman utama *website*.
Pasca-kondisi	*User* telah login pada sistem sebagai *administrator* atau *customer*.
Aliran Utama	
Aksi dari Aktor	**Tanggapan dari Sistem**
1. *User* memasukkan *username* dan *password* pada form *login* yang tersedia di halaman *website*.	2. Sistem memvalidasi bahwa *username* dan *password* tidak kosong dan telah terdaftar pada sistem.

	3. Sistem menampilkan status *login* pada halaman *website*.
Aliran Alternatif 1: *username* atau *password* kosong. (User pada langkah nomer 1 aliran utama tidak memasukkan *username* atau *password*.)	
Aksi dari Aktor	**Tanggapan dari Sistem**
	1. Menampilkan pesan bahwa *username/password* harus diisi.
Aliran Alternatif 2: *username* atau *password* belum terdaftar pada sistem. (*User* pada langkah nomer 1 aliran utama memasukkan *username* dan *password* tidak sesuai.)	
Aksi dari Aktor	**Tanggapan dari Sistem**
	1. Menampilkan pesan bahwa *username*/*password* salah.

Sumber : [Analisis]

3. *Use Case Spesification Logout*

Tabel 4. 7 : *Use case specification logout*

Nama *use case*	*Logout*
Aktor	*Administrator* atau *customer*.
Deskripsi	Sistem harus menyediakan fasilitas *logout* bagi pengguna, sehingga pengguna yang telah *login* dapat keluar dari aplikasi.
Pra-kondisi	*User* telah melakukan *use case login*
Pasca-kondisi	Aktor telah *logout* dari sistem.
Aliran Utama	
Aksi dari Aktor	**Tanggapan dari Sistem**
1. Aktor menekan tombol ”logout” pada bagian status *login* di halaman *website*.	2. Sistem melakukakan proses *logout* untuk aktor tersebut dan menampilkan form *login* kembali.

Sumber : [Analisis]

4. *Use Case Spesification* Melihat Daftar User

Tabel 4. 8 : *Use case specification* melihat daftar *user*

Nama *use case*	Melihat daftar *user*
Aktor	*Administrator*
Deskripsi	Sistem harus menyediakan fasilitas untuk melihat daftar *user* yang terdaftar pada sistem. Proses ini hanya dapat dilakukan oleh *administrator*.
Pra-kondisi	*User* telah melakukan *use case login* sebagai *administrator* dan mengakses halaman administrasi *website*.
Pasca-kondisi	Sistem menampilkan daftar *customer*.

Aliran Utama	
Aksi dari Aktor	**Tanggapan dari Sistem**
1. Administrator memilih menu "Manajemen User".	2. Sistem menampilkan daftar *user* yang telah terdaftar pada sistem. Daftar yang dapat dilihat berupa *username*, *email*, dan kategori *user*.

Sumber : [Analisis]

5. *Use Case Spesification* Melihat Data Profil User

Tabel 4. 9 : *Use case specification* melihat data profil *user*

Nama *use case*	Melihat data profil *user*
Aktor	*Administrator* atau *customer*
Deskripsi	Sistem harus menyediakan fasilitas untuk melihat data profil *user*. Dimana data *user* yang dapat dilihat berupa *username*, kategori *user*, nama lengkap, alamat *user*, *email*, propinsi, kota, kode pos, dan nomor telepon *user*. Untuk *partner* terdapat tambahan tentang informasi *website partner* dan kode aktifasi *web service*.
Pra-kondisi	*Administrator* telah melakukan *use case* melihat daftar *user*, atau *customer* telah melakukan *use case login*.
Pasca-kondisi	Sistem menampilkan detail profil *user*.
Aliran Utama	
Aksi dari Aktor	**Tanggapan dari Sistem**
1. *Administrator* menekan tombol "detail" pada salah satu baris dari daftar *user* setelah melakukan *use case* melihat daftar *user*. Atau *customer* menekan tombol "profil" pada bagian status *login* di halaman *website*.	2. Sistem menampilkan *username*, kategori *user*, nama lengkap, alamat *user*, *email*, propinsi, kota, kode pos, dan nomor telepon *user*. Untuk *user partner* terdapat tambahan tentang informasi *website partner* dan kode aktifasi *web service*.

Sumber : [Analisis]

6. *Use Case Spesification* Mengedit Data Profil User

Tabel 4. 10 : *Use case specification* mengedit data profil *user*

Nama *use case*	Mengedit data profil *user*
Aktor	*Administrator* dan *customer*
Deskripsi	Sistem harus menyediakan fasilitas untuk mengedit data profil *user*. Proses ini hanya dapat dilakukan oleh *user* itu sendiri. Data yang dapat dirubah berupa *username*, nama lengkap, alamat *user*, *email*, propinsi, kota, kode pos, dan nomor telepon *user*.
Pra-kondisi	*Administrator* atau *customer* telah melakukan *use case* melihat data

	profil *user*. Atau *Administrator* telah melakukan *use case* melihat daftar *user* di halaman administrasi *website*.
Pasca-kondisi	Data *user* yang telah diedit dan disimpan dalam sistem.

Aliran Utama

Aksi dari Aktor	**Tanggapan dari Sistem**
1. *Administrator* atau *customer* menekan tombol "edit profil" pada bagian status *login* di halaman *website*. Atau *Administrator* menekan tombol "edit profil" di bagian data *administrator* yang merupakan hasil melakukan *use case* melihat daftar *user*.	2. Sistem menampilkan *form* berisi *username*, nama lengkap, alamat *user*, *email*, propinsi, kota, kode pos, dan nomor telepon *user* yang bersangkutan.
3. *Administrator* atau *customer* mengedit data *username*, nama lengkap, alamat *user*, *email*, propinsi, kota, kode pos, dan nomor telepon.	4. Sistem memvalidasi *username*, nama lengkap, alamat *user*, *email*, propinsi, kota, kode pos, atau nomor telepon tidak boleh kosong, *username*, nama lengkap, alamat *user*, *email*, kode pos, atau nomor telepon tidak memiliki karakter yang tidak diijinkan oleh sistem, dan *username* atau nama lengkap yang telah diedit belum terdaftar di sistem.
	5. Sistem menampilkan pesan "Data anda telah berhasil diproses".

Aliran Alternatif 1: *username*, nama lengkap, alamat *user*, *email*, propinsi, kota, kode pos, atau nomor telepon kosong.

(*Administrator* atau *customer* pada langkah nomer 3 aliran utama memasukkan data kosong pada *username*, nama lengkap, alamat *user*, *email*, propinsi, kota, kode pos, atau nomor telepon.)

Aksi dari Aktor	**Tanggapan dari Sistem**
	1. Menampilkan pesan bahwa *username*, nama lengkap, alamat *user*, *email*, propinsi, kota, kode pos, atau nomor telepon harus diisi.
	2. Kembali pada langkah nomer 2 aliran utama.

Aliran Alternatif 2: *username*, nama lengkap, *email*, alamat, kode pos, atau nomor telepon memiliki karakter yang tidak diijinkan oleh sistem.

(*Administrator* atau *customer* pada langkah nomer 3 aliran utama memasukkan *username*, nama lengkap, *email*, alamat, kode pos, atau nomor telepon dengan karakter yang tidak diperbolehkan oleh sistem.)

Aksi dari Aktor	Tanggapan dari Sistem
	1. Menampilkan pesan bahwa karakter *username*, nama lengkap, *email*, alamat, kode pos, atau nomor telepon tidak diperbolehkan.
	2. Kembali pada langkah nomer 2 aliran utama.

Aliran Alternatif 3: *username*, atau nama lengkap telah terdaftar.

(*Administrator* atau *customer* pada langkah nomer 3 aliran utama memasukkan *username* atau nama lengkap telah terdaftar dalam sistem.)

Aksi dari Aktor	Tanggapan dari Sistem
	1. Menampilkan pesan bahwa *username* atau .nama lengkap telah terdaftar.
	2. Kembali pada langkah nomer 2 aliran utama.

Sumber : [Analisis]

7. *Use Case Spesification* Mengedit Password User

Tabel 4. 11 : *Use case specification* mengedit *password user*

Nama *use case*	Mengedit *password user*
Aktor	*Administrator* atau *customer*
Deskripsi	Sistem harus menyediakan fasilitas untuk mengedit *password user*. Proses ini hanya dapat dilakukan oleh *user* itu sendiri.
Pra-kondisi	*Administrator* atau *customer* telah melakukan *use case* melihat data profil *user*. Atau *administrator* telah melakukan *use case* melihat daftar *user* di halaman administrasi *website*.
Pasca-kondisi	Sistem menampilkan detail profil *user*.

Aliran Utama

Aksi dari Aktor	Tanggapan dari Sistem
1. *Administrator* atau *customer* menekan tombol ”edit password” pada bagian status *login* di halaman *website*. Atau *Administrator* menekan tombol ”edit password” di bagian data *administrator* sendiri yang merupakan hasil melakukan *use case* melihat daftar *user*.	2. Sistem menampilkan form berupa *password* lama, *password* baru, dan *re-type password* baru.
3. *Administrator* atau *customer* memasukkan data pada *form* yang tersedia.	4. Sistem memvalidasi data masukkan dimana *password* lama harus benar.
	5. Sistem menampilkan pesan ”Data anda

	berhasil diproses”
Aliran Alternatif 1: *password* lama tidak sesuai. (*Administrator* atau *customer* pada langkah nomer 1 aliran utama memasukkan *password* lama yang salah.)	
Aksi dari Aktor	**Tanggapan dari Sistem**
	1. Menampilkan pesan bahwa *password* lama salah.
	2. Kembali pada langkah nomer 2 aliran utama.

Sumber : [Analisis]

8. *Use Case Spesification* Menghapus User

Tabel 4. 12 : *Use case specification* menghapus *user*

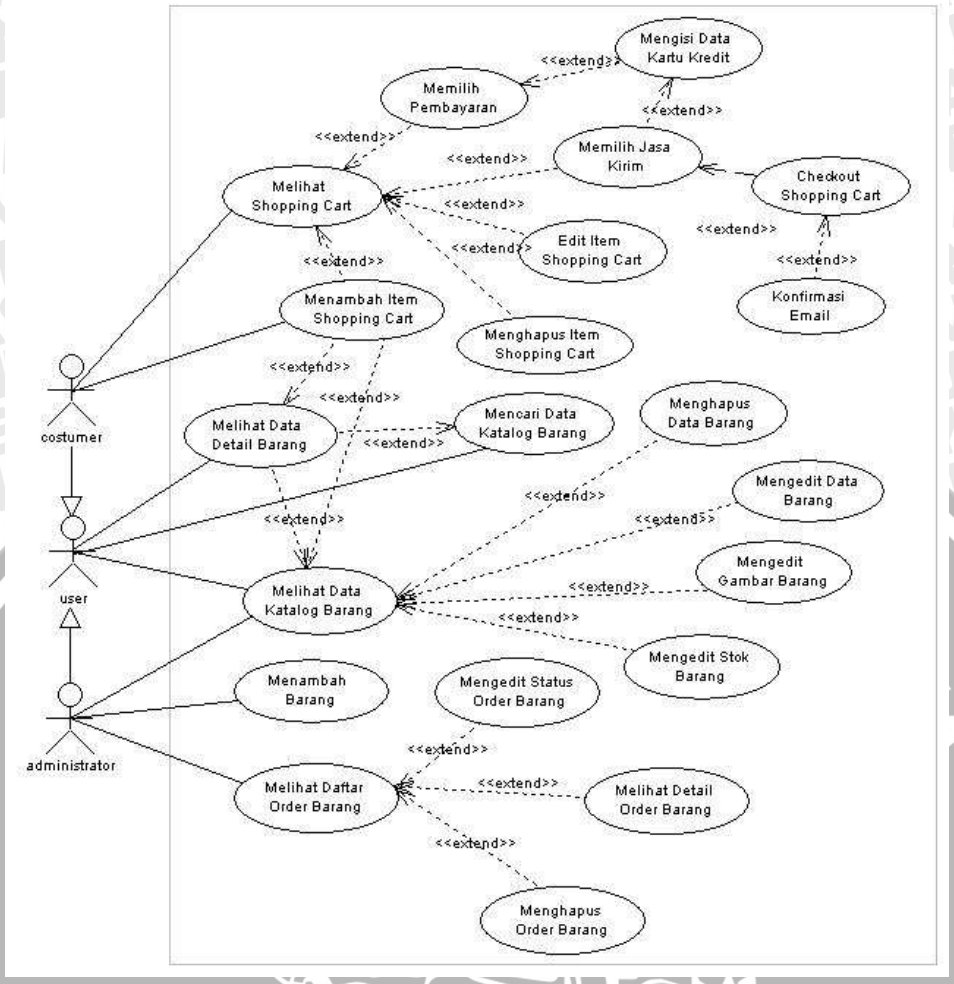
Nama *use case*	Menghapus *user*
Aktor	*Administrator*
Deskripsi	Sistem harus menyediakan fasilitas untuk menghapus *user*. Fasilitas ini hanya dapat dilakukan oleh *administrator*.
Pra-kondisi	*Administrator* telah melakukan *use case* melihat daftar *user*.
Pasca-kondisi	Data *user* dihapus dari dalam sistem.
Aliran Utama	
Aksi dari Aktor	**Tanggapan dari Sistem**
1. *Administrator* menekan tombol ”hapus” pada salah satu data *user* yang merupakan hasil melakukan *use case* melihat daftar *user*.	2. Sistem menampilkan pesan bahwa data *user* berhasil dihapus.

Sumber : [Analisis]

4.1.2.2.2. *Use Case Diagram* untuk Modul Katalog Barang

Use case diagram untuk modul katalog barang melibatkan 3 buah aktor dan 19 buah *use case*. Aktor – aktor yang terlibat pada modul katalog barang aplikasi *Web Service E-commerce* adalah :

1. *User*
2. *Administrator*
3. *Customer*



Gambar 4. 7 : *Use Case Diagram* untuk Modul Katalog Barang
Sumber : [Analisis]

1. *Use Case Spesification* Mencari Data Barang

Tabel 4. 13 : *Use case specification* mencari data barang

Nama *use case*	Mencari data barang
Aktor	*User*
Deskripsi	Sistem harus menyediakan fasilitas untuk mencari informasi pada katalog barang yang terdapat pada sistem.
Pra-kondisi	*User* mengakses halaman utama *website*.
Pasca-kondisi	Sistem menampilkan data barang yang dicari oleh *user*.
Aliran Utama	
Aksi dari Aktor	**Tanggapan dari Sistem**
1. *User* memasukkan "keyword" pada *form* pencarian data pada halaman *website*.	2. Sistem menampilkan daftar data barang yang dicari oleh *user*.
Aliran Alternatif 1: Masukan *keyword* kosong.	
(*User* pada langkah nomer 1 aliran utama memasukkan data kosong.)	

Aksi dari Aktor	Tanggapan dari Sistem
	1. Menampilkan pesan bahwa data tidak ditemukan.

Sumber : [Analisis]

2. *Use Case Spesification* Melihat Data Katalog Barang

Tabel 4. 14 : *Use case specification* melihat data katalog barang

Nama *use case*	Melihat data katalog barang
Aktor	*User*
Deskripsi	Sistem harus menyediakan fasilitas untuk melihat data katalog barang. Data tersebut berupa nama barang, harga barang, dan sekilas deskripsi barang.
Pra-kondisi	*User* mengakses halaman utama *website* atau *administrator* mengakses halaman administrasi *website*.
Pasca-kondisi	Sistem menampilkan daftar barang.
Aliran Utama	
Aksi dari Aktor	**Tanggapan dari Sistem**
1. *User* memilih menu ”produk” pada halaman *website*. Atau *administrator* memilih menu ”produk” pada halaman administrasi *website*.	2. Sistem menampilkan daftar katalog barang berupa nama barang, harga barang, dan sekilas deskripsi barang.

Sumber : [Analisis]

3. *Use Case Spesification* Melihat Data Detail Barang

Tabel 4. 15 : *Use case specification* melihat data detail barang

Nama *use case*	Melihat data detail barang
Aktor	*User*
Deskripsi	Sistem harus menyediakan fasilitas untuk melihat data detail barang. Data tersebut berupa nama barang, harga barang, status barang/stok barang, dan deskripsi barang.
Pra-kondisi	*User* telah melakukan *use case* melihat data katalog barang atau telah melakukan *use case* mencari data barang.
Pasca-kondisi	Sistem menampilkan detail barang.
Aliran Utama	
Aksi dari Aktor	**Tanggapan dari Sistem**
1. *User* menekan tombol ”selengkapnya” pada salah satu barang yang merupakan hasil melakukan *use case* melihat data katalog	2. Sistem menampilkan data detail barang berupa nama barang, harga barang, status barang/stok barang, dan deskripsi barang,.

barang. Atau menekan tombol "selengkapnya" pada barang yang merupakan hasil melakukan *use case* mencari data barang.	

Sumber : [Analisis]

4. *Use Case Spesification* Menambah Barang

Tabel 4. 16 : *Use case specification* menambah barang

Nama *use case*	Menambah barang
Aktor	*Administrator*
Deskripsi	Sistem harus menyediakan fasilitas untuk menambah barang. Data yang harus disimpan dari proses penambahan data ini adalah nama barang, kategori barang, harga barang, berat barang, gambar barang, dan deskripksi barang. Fasilitas ini hanya dapat digunakan oleh *administrator*.
Pra-kondisi	*Administrator* mengakses halaman administrasi *website*.
Pasca-kondisi	Data barang berhasil ditambahkan dan disimpan kedalam sistem.

Aliran Utama

Aksi dari Aktor	**Tanggapan dari Sistem**
1. *Administrator* memilih menu "tambah produk" pada halaman administrasi *website*.	2. Sistem menampilkan *form* tambah katalog barang. Dimana terdapat *field* nama barang, kategori barang, harga, berat barang, jumlah barang, gambar, dan deskripsi barang.
3. *Administrator* memasukkan data nama barang, kategori barang, harga, berat barang, jumlah barang, gambar, dan deskripsi barang..	4. Sistem menampilkan pesan bahwa data telah berhasil diproses.

Aliran Alternatif 1: Masukan nama barang, kategori barang, harga, berat barang, jumlah barang, gambar, atau deskripsi barang kosong.

(*Administrator* pada langkah nomer 3 aliran utama memasukkan nama barang, kategori barang, harga, berat barang, jumlah barang, gambar, atau deskripsi barang kosong.)

Aksi dari Aktor	**Tanggapan dari Sistem**
	1. Menampilkan pesan bahwa nama barang, kategori barang, harga, berat barang, jumlah barang, gambar, atau deskripsi barang harus diisi.
	2. Kembali pada langkah nomer 2 aliran utama.

Sumber : [Analisis]

5. *Use Case Spesification* Mengedit Stok Barang

Tabel 4. 17 : *Use case specification* mengedit stok barang

Nama *use case*	Mengedit stok barang
Aktor	*Administrator*
Deskripsi	Sistem harus menyediakan fasilitas untuk mengedit stok barang. Dimana fasilitas ini akan menambah/mengurangi jumlah stok barang yang telah ada pada sistem. Fasilitas ini hanya dapat diakses oleh *administrator*.
Pra-kondisi	*Administrator* telah melakukan *use case* melihat data katalog barang atau *use case* melihat data detail barang.
Pasca-kondisi	Jumlah stok barang telah ditambahkan kedalam sistem.

Aliran Utama

Aksi dari Aktor	**Tanggapan dari Sistem**
1. *Administrator* menekan tombol "edit stok" pada salah satu barang yang merupakan hasil melakukan *use case* melihat data katalog barang. Atau *administrator* menekan tombol "edit stok" hasil melakukan *use case* melihat data detail barang.	2. Sistem menampilkan *form* jumlah barang.
3. *Administrator* memasukkan data jumlah barang.	4. Sistem menampilkan pesan bahwa data barang berhasil disimpan.

Aliran Alternatif 1: Masukan pilihan jumlah barang kosong.

(*Administrator* pada langkah nomer 3 aliran utama memasukkan jumlah barang kosong.)

Aksi dari Aktor	**Tanggapan dari Sistem**
	3. Menampilkan pesan bahwa jumlah barang harus diisi.
	4. Kembali pada langkah nomer 2 aliran utama.

Sumber : [Analisis]

6. *Use Case Spesification* Mengedit Data Barang

Tabel 4. 18 : *Use case specification* mengedit data barang

Nama *use case*	Mengedit data barang
Aktor	*Administrator*
Deskripsi	Sistem harus menyediakan fasilitas untuk mengedit data barang. Data yang dapat dirubah dari proses ini adalah nama barang, kategori barang, harga barang, berat barang, dan deskripsi barang. Fasilitas ini hanya

	dapat digunakan oleh *administrator*.
Pra-kondisi	*Administrator* telah melakukan *use case* melihat data katalog barang atau *use case* melihat data detail barang.
Pasca-kondisi	Data barang telah diedit dan disimpan kedalam sistem.
Aliran Utama	
Aksi dari Aktor	**Tanggapan dari Sistem**
1. *Administrator* menekan tombol "edit data" pada salah satu data barang yang merupakan hasil melakukan *use case* melihat data katalog barang. Atau *administrator* menekan tombol "edit data" hasil melakukan *use case* melihat data detail barang.	2. Sistem menampilkan nama barang, kategori barang, harga barang, berat barang, dan deskripsi barang.
3. *Administrator* memasukkan data nama barang, kategori barang, harga barang, berat barang, dan deskripsi barang.	4. Sistem memvalidasi masukan nama barang, kategori barang, harga barang, berat barang, atau deskripsi barang tidak boleh kosong.
	5. Sistem menampilkan pesan bahwa data barang berhasil disimpan.
Aliran Alternatif 1: Masukan nama barang, kategori barang, harga barang, berat barang, dan deskripsi barang kosong. (*Administrator* pada langkah nomer 3 aliran utama memasukkan nama barang, kategori barang, harga barang, berat barang, dan deskripsi barang kosong.)	
Aksi dari Aktor	**Tanggapan dari Sistem**
	5. Sistem menampilkan pesan bahwa nama barang, kategori barang, harga barang, berat barang, dan deskripsi barang harus diisi.
	6. Kembali pada langkah nomer 2 aliran utama.

Sumber : [Analisis]

7. *Use Case Spesification* Mengedit Gambar Barang

Tabel 4. 19 : *Use case specification* mengedit gambar barang

Nama *use case*	Mengedit gambar barang
Aktor	*Administrator*
Deskripsi	Sistem harus menyediakan fasilitas untuk mengedit gambar barang. Fasilitas ini hanya dapat digunakan oleh *administrator*.
Pra-kondisi	*Administrator* telah melakukan *use case* melihat data katalog barang

	atau *use case* melihat data detail barang.
Pasca-kondisi	Gambar barang telah diedit dan disimpan kedalam sistem.
Aliran Utama	
Aksi dari Aktor	**Tanggapan dari Sistem**
1. *Administrator* menekan tombol "edit gambar" salah satu baris dari daftar kategori barang pada *use case* melihat data katalog barang atau tombol "edit gambar" saat melakukan *use case* melihat data detail barang.	2. Sistem menampilkan gambar barang dan *form upload* gambar baru.
3. *Administrator* memasukkan gambar baru.	4. Sistem memvalidasi bahwa gambar barang baru tidak boleh kosong dan telah ada dalam sistem.
	5. Sistem menampilkan pesan bahwa gambar barang telah berhasil diedit dan disimpan kedalam sistem.
Aliran Alternatif 1: Masukan gambar barang kosong. (*Administrator* pada langkah nomer 3 aliran utama memasukkan data kosong.)	
Aksi dari Aktor	**Tanggapan dari Sistem**
	1. Sistem menampilkan pesan bahwa gambar barang tidak boleh kosong.
	2. Kembali pada langkah nomer 2 aliran utama.

Sumber : [Analisis]

8. *Use Case Spesification* Menghapus Data Barang

Tabel 4. 20 : *Use case specification* menghapus data barang

Nama *use case*	Menghapus data barang
Aktor	*Administrator*
Deskripsi	Sistem harus menyediakan fasilitas untuk menghapus data barang. Fasilitas ini hanya dapat digunakan oleh *administrator*.
Pra-kondisi	*Administrator* telah melakukan *use case* melihat data katalog barang atau *use case* melihat data detail barang.
Pasca-kondisi	Data barang telah dihapus dari sistem.
Aliran Utama	
Aksi dari Aktor	**Tanggapan dari Sistem**
1. *Administrator* menekan tombol "hapus" salah satu baris dari daftar katalog barang yang	2. Sistem menampilkan pesan bahwa data barang telah dihapus dari sistem.

ditampilkan *use case* melihat data katalog barang atau menekan tombol "hapus" saat menjalankan *use case* melihat data detail barang.	

Sumber : [Analisis]

9. *Use Case Spesification* Melihat Daftar *Order* Barang

Tabel 4. 21 : *Use case specification* melihat daftar *order* barang

Nama *use case*	Melihat daftar *order* barang
Aktor	*Administrator*
Deskripsi	Sistem harus menyediakan fasilitas untuk melihat daftar *order* barang. Daftar *order* barang yang dapat dilihat berupa status pembayaran, tanggal pemesanan, nama pemesan, dan total pembelian barang. Fasilitas ini hanya dapat diakses oleh *administrator*.
Pra-kondisi	*Administrator* mengakses halaman administrasi *website*.
Pasca-kondisi	Sistem menampilkan daftar *order* barang.
Aliran Utama	
Aksi dari Aktor	**Tanggapan dari Sistem**
1. *Administrator* memilih menu "order" pada halaman administrasi *website*.	2. Sistem menampilkan daftar *order* barang berupa status pembayaran, tanggal pemesanan, nama pemesan, dan total pembelian barang.

Sumber : [Analisis]

10. *Use Case Spesification* Melihat Detail *Order* Barang

Tabel 4. 22 : *Use case specification* melihat detail *order* barang

Nama *use case*	Melihat detail *order* barang
Aktor	*Administrator*
Deskripsi	Sistem harus menyediakan fasilitas untuk melihat detail *order* barang. Data tersebut berupa status pembayaran, tanggal pemesanan, nama pemesan, nama barang yang dipesan, jenis jasa kirim, harga kirim, dan total pembelian. Fasilitas ini hanya dapat diakses oleh *administrator*.
Pra-kondisi	*Administrator* telah melakukan *use case* melihat daftar *order* barang.
Pasca-kondisi	Sistem menampilkan data detail *order* barang.
Aliran Utama	
Aksi dari Aktor	**Tanggapan dari Sistem**
1. *Administrator* menekan tombol "detail" pada salah satu data *order* barang yang merupakan	2. Sistem menampilkan data detail *order* barang berupa status pembayaran, tanggal

hasil melakukan *use case* melihat daftar barang.	pemesanan, nama pemesan, nama barang yang dipesan, jenis jasa kirim, harga kirim, dan total pembelian.

Sumber : [Analisis]

11. *Use Case Spesification* Mengedit Status *Order* Barang

Tabel 4. 23 : *Use case specification* mengedit status *order* barang

Nama *use case*	Mengedit status *order* barang
Aktor	*Administrator*
Deskripsi	Sistem harus menyediakan fasilitas untuk mengedit status *order* barang. Dimana fasilitas ini dapat diakses oleh *administrator*.
Pra-kondisi	*Administrator* telah melakukan *use case* melihat detail *order* barang.
Pasca-kondisi	Sistem menyimpan status hasil *editing* ke dalam sistem.
Aliran Utama	
Aksi dari Aktor	**Tanggapan dari Sistem**
1. *Administrator* menekan tombol "edit status" pada data *order* barang yang merupakan hasil melakukan *use case* melihat detail *order* barang. Dimana status *order* barang yang dapat diedit adalah status 'order'.	2. Sistem merubah status 'order' menjadi status 'complete' dan kemudian menyimpannya ke dalam sistem.

Sumber : [Analisis]

12. *Use Case Spesification* Menghapus *Order* Barang

Tabel 4. 24 : *Use case specification* menghapus *order* barang

Nama *use case*	Menghapus *order* barang
Aktor	*Administrator*
Deskripsi	Sistem harus menyediakan fasilitas untuk menghapus *order* barang. Fasilitas ini hanya dapat diakses oleh *administrator*.
Pra-kondisi	*Administrator* telah melakukan *use case* melihat daftar *order* barang atau telah melakukan *use case* melihat detail *order* barang.
Pasca-kondisi	Sistem menghapus data *order* barang.
Aliran Utama	
Aksi dari Aktor	**Tanggapan dari Sistem**
1. *Administrator* menekan tombol "hapus" pada salah satu data yang merupakan hasil melakukan *use case* melihat daftar *order* barang. Atau *Administrator* menekan tombol	2. Sistem menghapus data *order* barang dan menampikan pesan bahwa *order* barang telah berhasil dihapus.

"hapus" pada data yang merupakan hasil melakukan *use case* melihat detail *order* barang.	

Sumber : [Analisis]

13. *Use Case Spesification* Melihat *Shopping Cart*

Tabel 4. 25 : *Use case specification* melihat *shopping cart*

Nama *use case*	Melihat *shopping cart*
Aktor	*Customer*
Deskripsi	Sistem harus dapat menyediakan fasilitas untuk melihat item *shopping cart*. Dimana *customer* dapat melihat *item* yang dibeli dan disimpan di *shopping cart* (keranjang belanja) berupa nama barang, jumlah barang yang dibeli, harga barang per-item, dan harga total barang.
Pra-kondisi	*Customer* telah melakukan *use case login* atau telah melakukan *use case* menambah *item shopping cart.*
Pasca-kondisi	Sistem menampilkan data barang yang dipesan oleh *customer*.
Aliran Utama	
Aksi dari Aktor	**Tanggapan dari Sistem**
1. *Customer* menekan tombol "lihat cart" pada halaman *website* atau telah melakukan *use case* menambah *item shopping cart*.	2. Sistem menampilkan data barang yang dipesan oleh *customer* berupa nama barang, jumlah barang yang dibeli, harga barang per-item, dan harga total barang.
Aliran Alternatif 1: *item shopping cart* kosong. (*Customer* belum melakukan *use case* menambah *item shopping cart.*)	
Aksi dari Aktor	**Tanggapan dari Sistem**
	7. Sistem menampilkan pesan "Shopping Cart anda masih kosong".

Sumber : [Analisis]

14. *Use Case Spesification* Menambah *Item Shopping Cart*

Tabel 4. 26 : *Use case specification* menambah *item shopping cart*

Nama *use case*	Menambah *item shopping cart*
Aktor	*Customer*
Deskripsi	Sistem harus menyediakan fasilitas untuk menambah item *shopping cart*. Data yang dapat disimpan di *shopping cart* ini berupa nama barang, jumlah barang yang dibeli, harga barang per-item, dan harga total barang.
Pra-kondisi	*Customer* mengakses halaman utama *website* atau telah melakukan *use*

	case melihat data katalog barang.
Pasca-kondisi	Sistem menyimpan data barang ke dalam *shopping cart*.
Aliran Utama	
Aksi dari Aktor	**Tanggapan dari Sistem**
1. *Customer* menekan tombol "masukkan cart" pada salah satu data barang yang merupakan hasil melakukan *use case* melihat data katalog barang. Atau menekan tombol "masukkan cart" saat melakukan *use case* melihat data detail barang.	2. Sistem menyimpan data barang ke dalam *shopping cart* dan menampilkannya berupa nama barang, jumlah barang yang dibeli, harga barang per-item, dan harga total barang.

Sumber : [Analisis]

15. *Use Case Spesification Edit Item Shopping Cart*

Tabel 4. 27 : *Use case specification* edit *item shopping cart*

Nama *use case*	Edit *item shopping cart*
Aktor	*Customer*
Deskripsi	Sistem harus menyediakan fasilitas utuk edit *Item Shopping Cart*. Dimana data yang dapat dirubah adalah jumlah *item* yang dipesan.
Pra-kondisi	*Customer* telah melakukan *use case* melihat *shopping cart*.
Pasca-kondisi	Sistem menampilkan perubahan jumlah barang pada *shopping cart*.
Aliran Utama	
Aksi dari Aktor	**Tanggapan dari Sistem**
	1. Sistem menampilkan *form* jumlah barang saat *customer* melakukan *use case* melihat *item shopping cart*.
2. *Customer* memasukkan data jumlah barang yang dipesan kemudian menekan tombol "update" yang tersedia di halaman *shopping cart*.	3. Sistem memvalidasi bahwa masukan data jumlah *item* tidak boleh kosong atau harus berisi angka.
	4. Sistem menampilkan perubahan jumlah barang yang telah diedit oleh *customer*.
Aliran Alternatif 1: Masukan jumlah *item* kosong.	
(*Customer* pada langkah nomer 2 aliran utama memasukkan data kosong.)	
	1. Sistem tidak menampilkan perubahan, dan jumlah barang tetap sebanyak data masukkan yang valid.
Aliran Alternatif 2: Masukan jumlah *item* selain angka.	

(*Customer* pada langkah nomer 2 aliran utama memasukkan data selain angka.)	
	1. Sistem tidak menampilkan perubahan, dan jumlah barang tetap sebanyak data masukkan yang valid.

Sumber : [Analisis]

16. *Use Case Spesification* Menghapus *Item Shopping Cart*

Tabel 4. 28 : *Use case specification* menghapus *item shopping cart*

Nama *use case*	Menghapus *item shopping cart*
Aktor	*Customer*
Deskripsi	Sistem harus menyediakan fasilitas untuk menghapus item *shopping cart*.
Pra-kondisi	*Customer* telah melakukan *use case* melihat *item shopping cart*.
Pasca-kondisi	Sistem menghapus data barang pada *shopping cart*.
Aliran Utama	
Aksi dari Aktor	**Tanggapan dari Sistem**
1. *Customer* menekan tombol "hapus" dari salah satu barang yang terdapat pada *shopping cart*.	2. Sistem menghapus data barang pada halaman *shopping cart*.

Sumber : [Analisis]

17. *Use Case Spesification* Memilih Pembayaran

Nama *use case*	Memilih Pembayaran
Aktor	*Customer*
Deskripsi	Sistem harus menyediakan fasilitas untuk memilih pembayaran untuk pembelian barang yang telah dipesan oleh *customer*. Dimana *customer* dapat mnggunakan metode pembayaran melalui kartu kredit atau melalui rekening bank.
Pra-kondisi	*Customer* telah melakukan *use case* melihat *item shopping cart*.
Pasca-kondisi	Sistem menampilkan informasi metode pembayaran.
Aliran Utama	
Aksi dari Aktor	**Tanggapan dari Sistem**
1. *Customer* menekan tombol "lanjut" pada halaman *shopping cart*.	2. Sistem menampilkan pilihan metode pembayaran terhadap pembelian barang.
3. *Customer* memilih salah satu metode pembayaran yang tersedia.	4.

Sumber : [Analisis]

18. *Use Case Spesification* Mengisi Data Kartu Kredit

Nama *use case*	Mengisi Data Kartu Kredit
Aktor	*Customer*
Deskripsi	Sistem harus menyediakan fasilitas untuk mengisi kartu kredit untuk *customer* yang menggunakan jasa kartu kredit dalam pembelian barang. Dimana *customer* dapat mengisi nomor kartu kredit dan waktu expired kartu kredit.
Pra-kondisi	*Customer* telah melakukan *use case* memilih pembayaran.
Pasca-kondisi	Sistem menampilkan informasi untuk memilih jasa kirim.

Aliran Utama	
Aksi dari Aktor	**Tanggapan dari Sistem**
1. *Customer* memilih metode pembayaran kartu kredit dari hasil melakukan *use case* memilih pembayaran.	2. Sistem menampilkan *form* kartu kredit berupa nomor kartu kredit dan waktu expired kartu kredit.
3. *Customer* memasukkan nomor kartu kredit dan waktu expired kartu kredit.	4. Sistem memvalidasi data masukan bahwa nomor kartu kredit dan waktu expired kartu kredit tidak boleh kosong dan valid.
	5. Sistem menampilkan informasi untuk memilih jasa kirim.
Aliran Alternatif 1: Masukan nomor kartu kredit dan waktu expired kartu kredit kosong. (User pada langkah nomer 3 aliran utama memasukkan nomor kartu kredit dan waktu expired kartu kredit data kosong.)	
	1. Sistem menampilkan pesan bahwa nomor kartu kredit dan waktu expired kartu kredit harus diisi.
	2. Kembali pada langkah nomer 2 aliran utama.
Aliran Alternatif 2: Masukan nomor kartu kredit dan waktu expired kartu kredit tidak *valid*. (User pada langkah nomer 3 aliran utama memasukkan nomor kartu kredit dan waktu expired kartu kredit data tidak valid.)	
	1. Sistem menampilkan pesan bahwa nomor kartu kredit dan waktu expired kartu kredit harus benar.
	2. Kembali pada langkah nomer 2 aliran utama.

Sumber : [Analisis]

19. *Use Case Spesification* Memilih Jasa Kirim

Tabel 4. 29 : *Use case specification* memilih jasa kirim

Nama *use case*	Memilih Jasa Kirim
Aktor	*Customer*
Deskripsi	Sistem harus menyediakan fasilitas untuk memilih jasa kirim untuk pengiriman barang yang telah dipesan oleh *customer*. Dimana *customer* dapat melihat informasi pengiriman barang berupa nama jasa kirim dan harga jasa kirim.
Pra-kondisi	*Customer* telah melakukan *use case* melihat *item shopping cart*.
Pasca-kondisi	Sistem menampilkan informasi pemesanan barang.
Aliran Utama	
Aksi dari Aktor	**Tanggapan dari Sistem**
5. *Customer* menekan tombol ”lanjut” pada halaman *shopping cart*.	6. Sistem menampilkan *form* pilihan jasa pengiriman barang, *form* alamat, kota, dan kode pos tujuan pengiriman barang.
7. *Customer* memasukkan data jasa pengiriman barang, alamat, kota dan kode pos tujuan pengiriaman barang.	8. Sistem memvalidasi data masukan bahwa jasa pengiriman barang, alamat, kota, dan kode pos tidak boleh kosong.
	9. Sistem menampilkan informasi pemesanan barang berupa tujuan pengiriman barang, barang yang dipesan, jenis pengiriman barang, harga pengiriman barang, dan total pembayarang, serta catatan untuk aturan pembayaran barang.
Aliran Alternatif 1: Masukan jasa pengiriman barang, alamat, kota, atau kode pos kosong. (User pada langkah nomer 3 aliran utama memasukkan jasa pengirirman barang, alamat, kota, kode pos data kosong.)	
	3. Sistem menampilkan pesan bahwa jasa pengiriman barang, alamat, kota, kode pos harus diisi.
	4. Kembali pada langkah nomer 2 aliran utama.

Sumber : [Analisis]

20. *Use Case Spesification Checkout Shoppping Cart*

Tabel 4. 30 : *Use case specification checkout shopping cart*

Nama *use case*	*Checkout shopping cart*

Aktor	*Customer*
Deskripsi	Sistem harus menyediakan fasilitas untuk *check out shopping cart*. Dimana fasilitas ini untuk menyimpan data yang telah dipesan oleh *customer* kedalam sistem berupa nama barang, jumlah barang, harga barang per-*item*, harga total barang, jenis pengiriman barang, biaya pengiriman barang, dan nomor rekening untuk proses pembayaran, serta batas waktu pembayaran.
Pra-kondisi	*Customer* telah melakukan *use case* memilih jasa kirim.
Pasca-kondisi	Sistem menyimpan data pesanan barang ke dalam sistem.
Aliran Utama	
Aksi dari Aktor	**Tanggapan dari Sistem**
1. *Customer* menekan tombol "checkout" pada halaman *shopping cart.*	2. Sistem menyimpan data pesanan barang ke dalam sistem.

Sumber : [Analisis]

21. *Use Case Spesification* Konfirmasi *Email*.

Tabel 4. 31 : *Use case specification* konfirmasi *email*

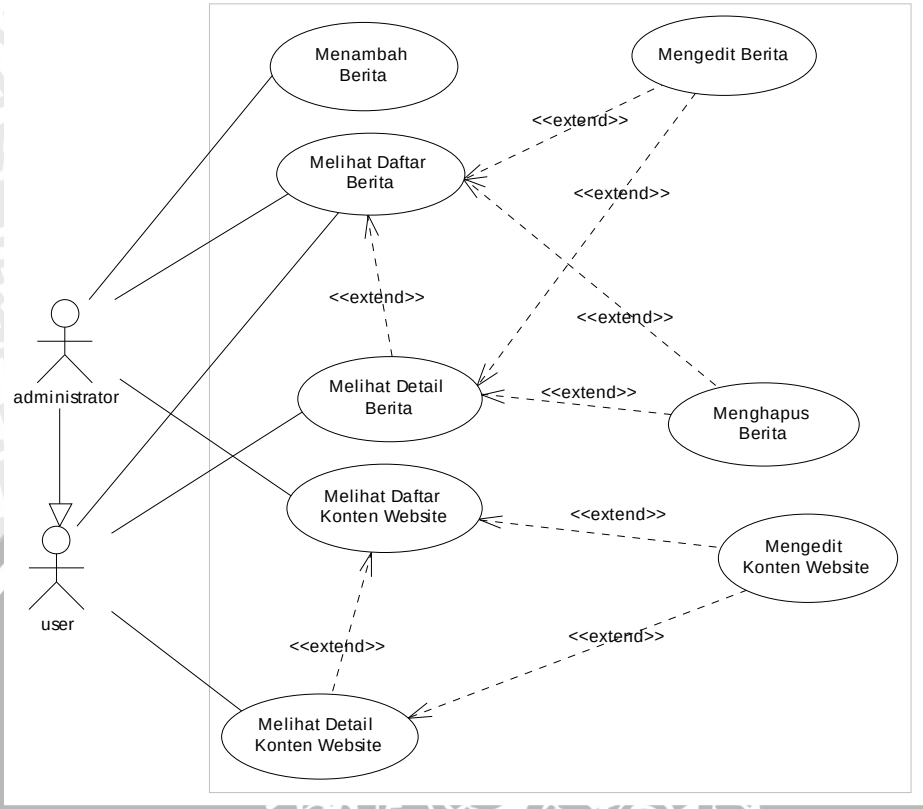
Nama *use case*	Konfirmasi *email*
Aktor	*Customer*
Deskripsi	Sistem harus menyediakan fasilitas untuk konfirmasi *email*. Dimana informasi tentang barang yang dipesan oleh *customer* dikirimkan ke alamat *email customer* tersebut.
Pra-kondisi	*Customer* telah melakukan *use case checkout shopping cart*.
Pasca-kondisi	Sistem mengirimkan informasi barang ke *email customer*.
Aliran Utama	
Aksi dari Aktor	**Tanggapan dari Sistem**
	1. Sistem mengirimkan informasi barang ke *email customer*.

Sumber : [Analisis]

4.1.2.2.3. *Use Case Diagram* untuk Modul Data *Website*

Use case diagram untuk modul data *website* melibatkan 2 buah aktor dan 8 buah *use case*. Aktor – aktor yang terlibat pada modul data *website* aplikasi *Web Service E-commerce* adalah :

1. *User*
2. *Administrator*



Gambar 4. 8 : *Use Case Diagram* untuk Modul Data *Website*
Sumber : [Analisis]

1. *Use Case Spesification* Melihat Daftar Berita.

Tabel 4. 32 : *Use case specification* melihat daftar berita/*event*

Nama *use case*	Melihat daftar berita/*event*
Aktor	*User*
Deskripsi	Sistem harus menyediakan fasilitas untuk melihat daftar berita aplikasi *e-commerce*. Data yang ditampilkan berupa judul berita, tanggal berita, dan sekilas isi berita.
Pra-kondisi	*User* mengakses halaman utama website atau *administrator* mengakses halaman administrasi *website*.
Pasca-kondisi	Sistem menampilkan daftar berita.
Aliran Utama	
Aksi dari Aktor	**Tanggapan dari Sistem**
1. *User* memilih menu "**berita**" pada halaman *website*. Atau *administrator* memilih menu "**berita**" pada halaman administrasi *website*.	2. Sistem menampilkan daftar berita berupa judul berita, tanggal berita, dan sekilas isi berita.

Sumber : [Analisis]

2. *Use Case Spesification* Melihat Detail Berita.

Tabel 4. 33 : *Use case specification* melihat detail berita

Nama *use case*	Melihat detail berita
Aktor	*User*
Deskripsi	Sistem harus menyediakan fasilitas untuk melihat detail berita. Data yang ditampilkan berupa judul berita, tanggal berita, isi detail berita, dan penulis berita.
Pra-kondisi	*User* telah melakukan *use case* melihat daftar berita atau *administrator* telah melakukan *use case* melihat daftar berita.
Pasca-kondisi	Sistem menampilkan data detail berita.
Aliran Utama	
Aksi dari Aktor	**Tanggapan dari Sistem**
1. *User* menekan tombol "**selengkapnya**" pada salah satu berita yang merupakan hasil melakukan *use case* melihat daftar berita. Atau *administrator* menekan tombol "**detail**" pada salah satu berita yang merupakan hasil melakukan *use case* melihat daftar berita.	2. Sistem menampilkan data detail berita berupa judul berita, tanggal berita, isi detail berita, dan penulis berita.

Sumber : [Analisis]

3. *Use Case Spesification* Menambah Berita.

Tabel 4. 34 : *Use case specification* manambah berita

Nama *use case*	Menambah berita
Aktor	*Administrator*
Deskripsi	Sistem harus menyediakan fasilitas untuk menambah berita. Data yang disimpan pada proses ini berupa judul berita, tanggal berita, isi berita, dan penulis berita. Fasilitas ini hanya dapat diakses oleh *administrator*.
Pra-kondisi	*Administrator* mengakses halaman administrasi *website*.
Pasca-kondisi	Sistem menyimpan data berita baru.
Aliran Utama	
Aksi dari Aktor	**Tanggapan dari Sistem**
1. *Administrator* memilih menu "**tambah berita**" pada halaman administrasi *website*.	2. Sistem menampilkan *form* berupa judul berita, tanggal berita, dan isi berita.
3. *Administrator* memasukkan data judul berita, tanggal berita, dan isi berita.	4. Sistem memvalidasi data masukan bahwa judul berita, tanggal berita, dan isi berita tidak boleh kosong.

	5. Sistem menampilkan pesan bahwa data berhasil diproses.
Aliran Alternatif 1 : judul berita, tanggal berita, dan isi berita kosong (*Administrator* pada langkah nomer 3 aliran utama memasukkan judul berita, tanggal berita, dan isi berita kosong.)	
	1. Sistem menampilkan pesan bahwa judul berita, tanggal berita, dan isi berita harus diisi.
	2. Kembali pada langkah nomer 2 aliran utama.

Sumber : [Analisis]

4. *Use Case Spesification* Mengedit Berita.

Tabel 4. 35 : *Use case specification* mengedit berita

Nama *use case*	Mengedit berita
Aktor	*Administrator*
Deskripsi	Sistem harus menyediakan fasilitas untuk mengedit. Data yang dirubah pada proses ini berupa judul berita, dan isi berita. Fasilitas ini hanya dapat diakses oleh *administrator*.
Pra-kondisi	*Administrator* telah melakukan *use case* melihat daftar berita atau telah melakukan *use case* melihat detail berita.
Pasca-kondisi	Data berita telah diedit dan disimpan kedalam sistem.
Aliran Utama	
Aksi dari Aktor	**Tanggapan dari Sistem**
1. *Administrator* menekan tombol "edit" pada salah satu data berita yang merupakan hasil melakukan *use case* melihat daftar berita atau menekan tombol "edit" pada data berita yang merupakan hasil melakukan *use case* melihat detail berita.	2. Sistem menampilkan *form* judul berita dan isi berita.
3. *Administrator* mengedit data judul berita dan isi berita.	4. Sistem memvalidasi judul berita dan isi berita tidak boleh kosong.
	5. Sistem menampilkan pesan bahwa data berhasil diproses.
Aliran Alternatif 1 : judul berita atau isi berita kosong (*Administrator* pada langkah nomer 3 aliran utama memasukkan judul berita atau isi berita kosong.)	
	1. Sistem menampilkan pesan bahwa judul

	berita dan isi berita harus diisi.
	2. Kembali pada langkah nomer 2 aliran utama.

Sumber : [Analisis]

5. *Use Case Spesification* Menghapus Daftar Berita.

Tabel 4. 36 : *Use case specification* menghapus berita

Nama *use case*	Menghapus berita
Aktor	*Administrator*
Deskripsi	Sistem harus menyediakan fasilitas untuk menghapus berita. Dimana fasilitas ini hanya dapat diakses oleh *administrator*.
Pra-kondisi	*Administrator* telah melakukan *use case* melihat daftar berita atau *administrator* telah melakukan *use case* melihat detail berita.
Pasca-kondisi	Data berita dihapus dari sistem.
Aliran Utama	
Aksi dari Aktor	**Tanggapan dari Sistem**
1. *Administrator* menekan tombol "**hapus**" salah satu baris daftar berita hasil dari melakukan *use case* melihat daftar berita atau *administrator* menekan tombol "**hapus**" hasil dari melakukan *use case* melihat detail berita.	2. Sistem menampilkan pesan bahwa data berita telah dihapus dari sistem.

Sumber : [Analisis]

6. *Use Case Spesification* Melihat Daftar Konten Website.

Tabel 4. 37 : *Use case specification* melihat daftar konten *website*

Nama *use case*	Melihat daftar konten *website*
Aktor	*Administrator*
Deskripsi	Sistem harus menyediakan fasilitas untuk melihat daftar konten *website*. Fasilitas ini hanya dapat diakses oleh *administrator*. Dimana konten *website* ini berupa *site map, site credit,* kontak, dan cara *order*.
Pra-kondisi	*Administrator* mengakses halaman administrasi *website*.
Pasca-kondisi	Sistem menampilkan daftar konten *website*.
Aliran Utama	
Aksi dari Aktor	**Tanggapan dari Sistem**
1. *Administrator* memilih menu "**konten website**" pada halaman administrasi *website*.	2. Sistem menampilkan daftar konten *website* yakni *site map, site credit,* kontak, dan cara *order*. Data yang ditampilkan berupa judul konten *website* dan isi konten *website*.

Sumber : [Analisis]

7. *Use Case Spesification* Melihat Detail Konten *Website*.

Tabel 4. 38 : *Use case specification* melihat detail konten *website*

Nama *use case*	Melihat detail konten *website*
Aktor	*User*
Deskripsi	Sistem harus menyediakan fasilitas untuk melihat detail konten *website*. Dimana fasilitas ini dapat diakses oleh *user* di halaman *website*, dan diakses oleh *administrator* di halaman administrasi website.
Pra-kondisi	*User* mengakses halaman utama *website* atau *administrator* telah melakukan *use case* melihat daftar konten *website*.
Pasca-kondisi	Sistem menampilkan data detail konten *website*.
Aliran Utama	
Aksi dari Aktor	**Tanggapan dari Sistem**
1. *User* memilih menu "cara order", "kontak", "site map", atau "site credit" pada halaman *website*. Atau *administrator* menekan tombol "detail" pada salah satu data konten yang merupakan melakukan *use case* melihat daftar konten *website*.	2. Sistem menampilkan data detail konten *website* berupa judul konten dan isi konten.

Sumber : [Analisis]

8. *Use Case Spesification* Mengedit Konten *Website*.

Tabel 4. 39 : *Use case specification* mengedit konten *website*

Nama *use case*	Mengedit konten *website*
Aktor	*Administrator*
Deskripsi	Sistem harus menyediakan fasilitas untuk mengedit konten website. Fasilitas ini hanya dapat digunakan oleh *administrator*.
Pra-kondisi	*Administrator* telah melakukan *use case* melihat daftar konten *website* atau telah melakukan *use case* melihat detail konten *website*.
Pasca-kondisi	Data konten *website* telah diedit dan disimpan kedalam sistem.
Aliran Utama	
Aksi dari Aktor	**Tanggapan dari Sistem**
1. *Administrator* menekan tombol "edit" salah satu data konten *website* hasil melakukan *use case* melihat daftar konten *website* atau *administrator* menekan tombol "edit" pada	2. Sistem menampilkan data berupa judul konten dan isi konten.

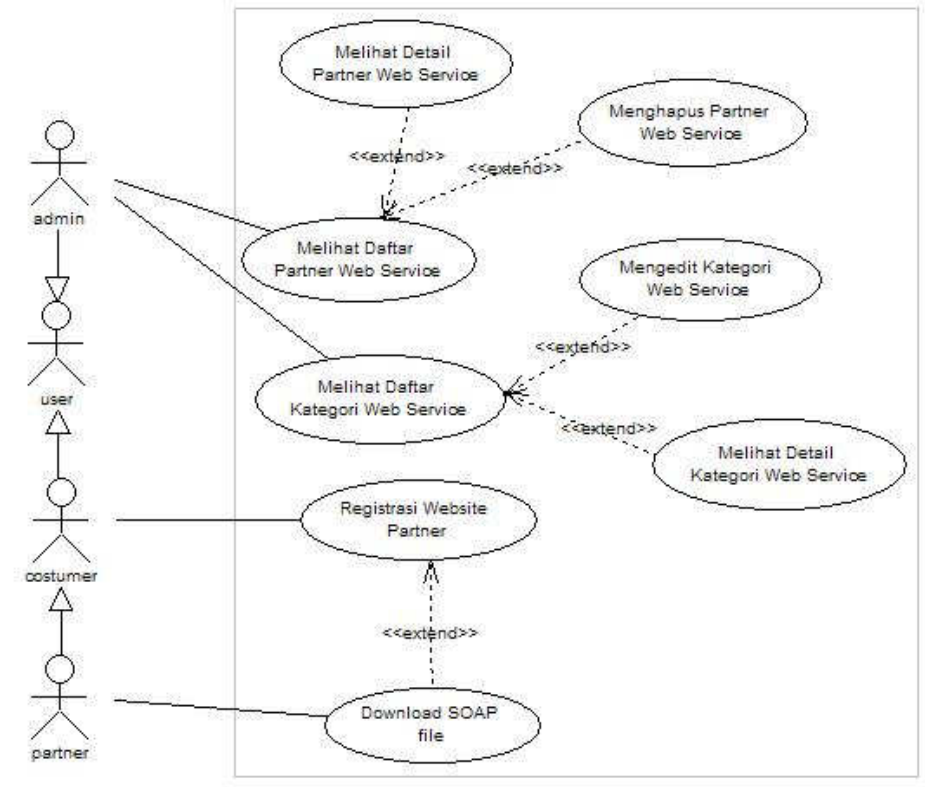
data konten *website* hasil melakukan *use case* melihat detail konten *website*.	
3. *Administrator* memasukkan data judul konten dan isi konten.	4. Sistem memvalidasi data judul konten atau isi konten tidak boleh kosong.
	5. Sistem menampilkan pesan bahwa data telah berhasil diproses.
Aliran Alternatif 1 : judul konten dan isi konten kosong (*Administrasi* pada langkah nomer 3 aliran utama memasukkan judul konten dan isi konten kosong.)	
	1. Sistem menampilkan pesan bahwa judul konten dan isi konten harus diisi.
	2. Kembali pada langkah nomer 2 aliran utama.

Sumber : [Analisis]

4.1.2.2.4. *Use Case Diagram* untuk Modul *Web Service*

Use case diagram untuk modul data *website* melibatkan 3 buah aktor dan 8 buah *use case*. Aktor – aktor yang terlibat pada modul data *website* aplikasi *Web Service E-commerce* adalah :

1. *Administrator*
2. *Customer*
3. *Partner*



Gambar 4. 9 : *Use Case Diagram* untuk Modul *Web Service*
Sumber : [Analisis]

1. *Use Case Spesification* Registrasi *Website Partner*.

Tabel 4. 40 : *Use case specification* registrasi *website partner*

Nama *use case*	Registrasi *website partner*
Aktor	*Customer*
Deskripsi	Sistem harus menyediakan fasilitas untuk registrasi *website*. Dimana yang didaftarkan oleh *partner* merupakan URL *website* milik *partner* dan *web service* yang akan digunakan pada situsnya.
Pra-kondisi	*Customer* melakukan *use case login*..
Pasca-kondisi	Sistem menyimpan data URL *website partner*.
Aliran Utama	
Aksi dari Aktor	**Tanggapan dari Sistem**
1. *Customer* menekan tombol "registrasi partner" pada halaman *website* di bagian *web service*.	2. Sistem menampilkan *form* berupa *field website partner*, dan jenis *web service*.
3. *Customer* memasukkan data *website partner* dan jenis *web service*.	4. Sistem memvalidasi bahwa *website partner* atau jenis *web service*. boleh kosong dan telah ada dalam sistem.
	5. Sistem menampilkan pesan bahwa proses

	registasi telah berhasil. Sistem juga menyertakan kode aktifasi serta *link* download SOAP *file*.
Aliran Alternatif 1 : *website partner atau jenis web service* kosong	
(*Customer* pada langkah nomer 3 aliran utama memasukkan *form website partner atau web service* kosong.)	
	1. Sistem menampilkan pesan bahwa *form website partner* atau jenis *web service* harus diisi.
	2. Kembali pada langkah nomer 2 aliran utama.
Aliran Alternatif 2 : *website partner* telah ada dalam sistem	
(User pada langkah nomer 3 aliran utama memasukkan *website partner* telah ada dalam sistem.)	
	1. Sistem menampilkan pesan bahwa *website* telah terdaftar.
	2. Kembali pada langkah nomer 2 aliran utama.

Sumber : [Analisis]

2. *Use Case Spesification Download* SOAP *file.*

Tabel 4. 41 : *Use case specification* download SOAP *file*

Nama *use case*	Download SOAP *file*
Aktor	*Partner*
Deskripsi	Sistem harus menyediakan fasilitas untuk *download* SOAP *file*. Dimana *file* tersebut merupakan SOAP *client* untuk mengakses *web service*.
Pra-kondisi	*Partner* telah berhasil melakukan *use case* registrasi *website partner*.
Pasca-kondisi	SOAP *file* telah berhasil di-*download* oleh *partner*.
Aliran Utama	
Aksi dari Aktor	**Tanggapan dari Sistem**
1. *Partner* menekan tombol ”klik disini” setelah berhasil melakukan *use case* registrasi *website partner*.	2. Sistem menampilkan kotak dialog untuk menyimpan SOAP *file* tersebut.
3. *Partner* menyimpannya ke PC *partner* sendiri.	

Sumber : [Analisis]

3. *Use Case Spesification* Melihat Daftar *Partner Web Service*.

Tabel 4. 42 : *Use case specification* melihat daftar *partner web service*

Nama *use case*	Melihat daftar *partner web service*	
Aktor	*Administrator*	
Deskripsi	Sistem harus menyediakan fasilitas untuk melihat daftar *parter web service*. Dimana data tersebut berupa *website partner* dan *username partner*. Fasilitas ini hanya dapat diakses oleh *administrator*.	
Pra-kondisi	*Administrator* mengakses halaman administrasi *website*.	
Pasca-kondisi	Sistem menampilkan daftar *partner web service*.	
Aliran Utama		
Aksi dari Aktor		**Tanggapan dari Sistem**
1. *Administrator* memilih menu "partner web service" pada halaman administrasi *website*.		2. Sistem menampilkan daftar *partner web service* yang terdaftar pada sistem.

Sumber : [Analisis]

4. *Use Case Spesification* Melihat Detail *Partner Web Service*.

Tabel 4. 43 : *Use case specification* melihat detail *partner web service*

Nama *use case*	Melihat detail *partner web service*	
Aktor	*Administrator*	
Deskripsi	Sistem harus menyediakan fasilitas untuk melihat detail *partner web service*. Dimana data yang dapat dilihat berupa *website partner*, *username partner*, tanggal pendaftaran *partner*, dan jenis *web service*. Fasilitas ini hanya dapat diakses oleh *administrator*.	
Pra-kondisi	*Administrator* telah melakukan *use case* melihat daftar *partner web service*.	
Pasca-kondisi	Sistem menampilkan data detail *partner web service*.	
Aliran Utama		
Aksi dari Aktor		**Tanggapan dari Sistem**
1. *Administrator* menekan tombol "detail" pada salah data *partner* yang merupakan hasil melakukan *use case* melihat daftar *partner web service*.		2. Sistem menampilkan data detail *partner* yang berupa nama *website partner*, *username partner*, tanggal pendaftaran *partner*, dan jenis *web service*.

Sumber : [Analisis]

5. *Use Case Spesification* Menghapus *Partner Web Service*

Tabel 4. 44 : *Use case specification* menghapus *partner web service*

Nama *use case*	Menghapus *website partner*
Aktor	*Administrator*
Deskripsi	Sistem harus menyediakan fasilitas menghapus *website partner*. Fasilitas ini hanya dapat diakses oleh *administrator*.
Pra-kondisi	*Administrator* telah melakukan *use case* melihat daftar *partner web service* atau telah melakukan *use case* melihat detail *partner web service*.
Pasca-kondisi	Sistem menghapus data *partner web service*.
Aliran Utama	
Aksi dari Aktor	**Tanggapan dari Sistem**
1. *Administrator* menekan tombol "**hapus**" pada salah satu data *partner* yang merupakan hasil melakukan *use case* melihat daftar *partner web service* atau menekan tombol "**hapus**" pada data *partner* yang merupakan hasil melakukan *use case* melihat detail *partner web service*.	2. Sistem menghapus data *partner web service* dan menampilkan pesan bahwa data *partner* berhasil dihapus.

Sumber : [Analisis]

6. *Use Case Spesification* Melihat Daftar Kategori *Web Service*.

Tabel 4. 45 : *Use case specification* melihat daftar kategori *web service*

Nama *use case*	Melihat daftar kategori *web service*
Aktor	*Administrator*
Deskripsi	Sistem harus menyediakan fasilitas untuk melihat daftar kategori *Web Service*. Dimana data yang dilihat berupa kategori *web service* dan sekilas isi *web service*.
Pra-kondisi	*Administrator* mengakses halaman administrasi *website.*
Pasca-kondisi	*Sistem* menampilkan daftar kategori *web service*.
Aliran Utama	
Aksi dari Aktor	**Tanggapan dari Sistem**
1. *Administrator* memilih menu "**kategori web service**" yang terdapat pada halaman administrasi *website.*	2. Sistem menampilkan daftar kategori *web service* yang berupa kategori *web service* dan *isi web service.*

Sumber : [Analisis]

7. *Use Case Spesification* Melihat Detail Kategori *Web Service*.

Tabel 4. 46 : *Use case specification* melihat detail kategori *web service*

Nama *use case*	Melihat detail kategori *web service*
Aktor	*Administrator*
Deskripsi	Sistem harus menyediakan fasiltas untuk melihat detail kategori *web service*. Dimana data tersebut berupa kategori *web service* dan detail isi *web service*.
Pra-kondisi	*Administrator* telah melakukan *use case* melihat daftar kategori *web service*.
Pasca-kondisi	*Sistem* menampilkan data detail kategori *web service*.
Aliran Utama	
Aksi dari Aktor	**Tanggapan dari Sistem**
1. *Administrator* menekan tombol "detail" pada salah satu data kategori *web service* yang merupakan hasil melakukan *use case* melihat daftar kategori *web service*.	2. Sistem menampilkan daftar kategori *web service* yang berupa kategori *web service* dan *isi web service*.

Sumber : [Analisis]

8. *Use Case Spesification* Mengedit Kategori *Web Service*.

Tabel 4. 47 : *Use case specification* mengedit kategori *web service*

Nama *use case*	Mengedit kategori *web service*
Aktor	*Administrator*
Deskripsi	Sistem harus menyediakan fasilitas untuk mengedit kategori *web service*. Data yang dapat dirubah berupa nama *web service* dan deskripsi *web service*. Fasilitas ini hanya dapat diakses oleh *administrator*.
Pra-kondisi	*Administrator* melakukan *use case* melihat daftar kategori *web service* atau melakukan *use case* melihat detail kategori *web service*.
Pasca-kondisi	Sistem menyimpan data perubahan kategori *web service*.
Aliran Utama	
Aksi dari Aktor	**Tanggapan dari Sistem**
1. *Administrator* menekan tombol "edit" pada salah satu data kategori *web service* yang merupakan hasil melakukan *use case* melihat daftar kategori *web service* atau menekan tombol "edit" pada data kategori *web service* yang merupakan hasil melakukan *use case* melihat detail kategori *web service*.	2. Sistem menampilkan *form* edit kategori *web service* yang *field*-nya berupa kategori *web service* dan isi *web service*.

3. *Administrator* memasukkan data kategori *web service* dan isi *web service*.	4. Sistem memvalidasi bahwa data kategori *web service* dan isi *web service* tidak boleh kosong.
	5. Sistem menyimpan data perubahan kategori *web service* dan menampilkan pesan bahwa data telah berhasil diproses.
Aliran Alternatif 1 : kategori *web service* atau isi *web service* kosong (*Administrator* pada langkah nomer 3 aliran utama memasukkan kategori *web service* atau isi *web service* kosong.)	
	1. Sistem menampilkan pesan bahwa kategori *web service* atau isi *web service* harus diisi.
	2. Kembali pada langkah nomer 2 aliran utama.

Sumber : [Analisis]

4.2. Perancangan

Perancangan berorientasi objek (*Object Oriented Design/OOD)* berhubungan dengan pengembangan model berorientasi objek dari sistem perangkat lunak untuk mengimplementasikan kebutuhan-kebutuhan yang telah teridentifikasi pada analisis kebutuhan (*Object Oriented Analysis*/*OOA*).

Proses perancangan perangkat lunak mempunyai dua tahap, yaitu perancangan umum dan perancangan detail. Perancangan umum menggambarkan relasi antar paket (*package)* dan klas (*class)* sebagai pemodelan sistem secara keseluruhan. Perancangan detail menggunakan *class diagram* dan *sequence diagram* sebagai pemodelan perangkat lunak, perancangan basis data, dan perancangan *user interface*. Perancangan detail melakukan perancangan terhadap pola hubungan antar komponen-komponen detail (dalam konteks berorientasi objek adalah klas dan objek) sehingga mampu membentuk sebuah fungsi kerja yang mampu memberikan pelayanan terhadap kebutuhan aktor.

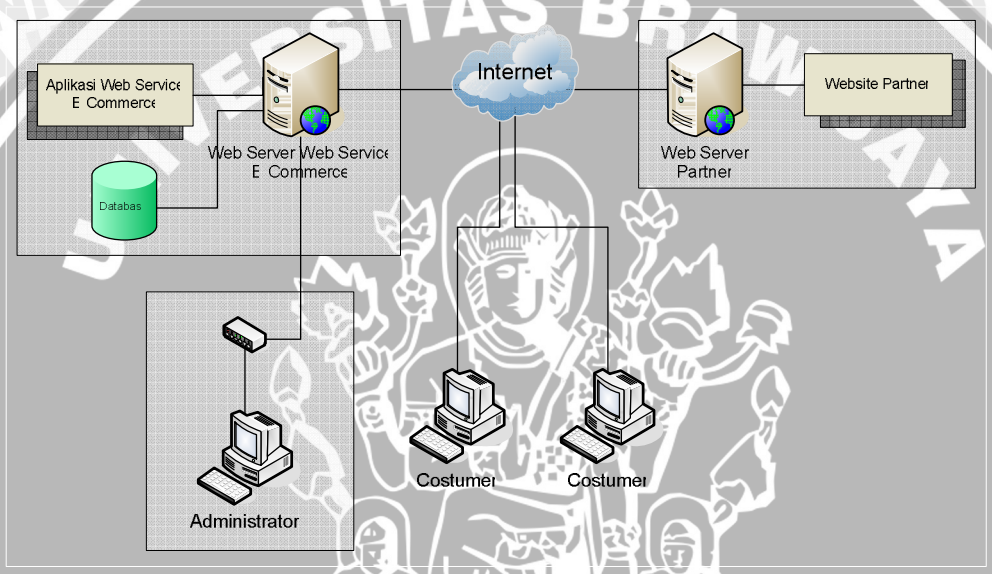
4.2.1. Perancangan Umum

Perancangan umum menggambarkan relasi antar paket dan klas sebagai pemodelan sistem secara keseluruhan. Perancangan umum customerikan pandangan keseluruhan sistem tanpa melihat detil dari masing-masing subsistem

yang ada.

4.2.1.1. Arsitektur Jaringan

Aplikasi *Web Service E-Commerce* menggunakan sebuah *web server* Apache dan *database server* MySQL. *Administrator* dapat mengakses aplikasi ini melalui jaringan lokal yang langsung terhubung dengan *server*. Sedangkan untuk *cosetumer* dan *partner* dapat mengaksesnya melalui internet. Untuk arsitektur jaringan dari aplikasi *Web Service E-Commerce* ini dapat dilihat pada gambar 4.9 dibawah ini.

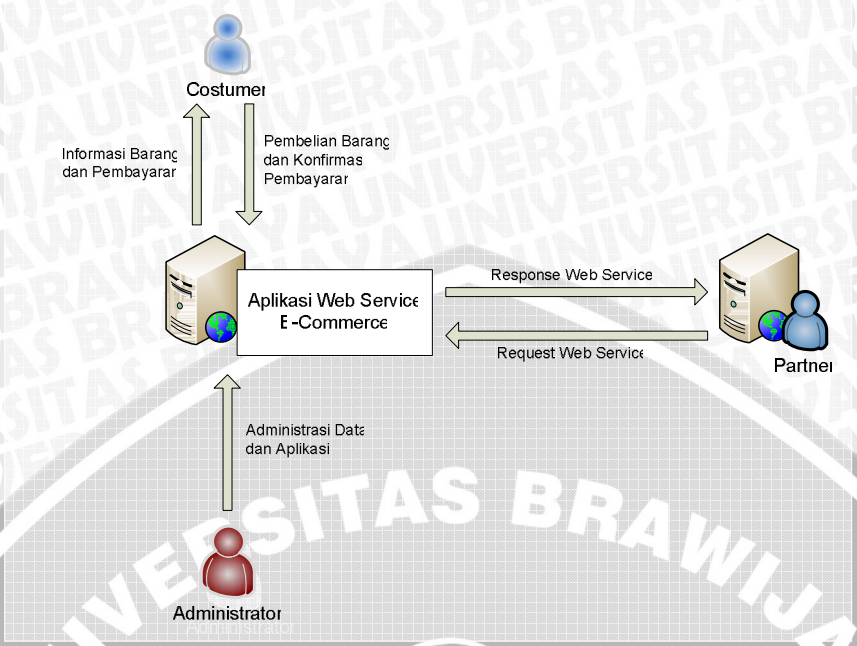


Gambar 4. 10 : Arsitektur jaringan aplikasi *Web Service E-Commerce*
Sumber : [Perancangan]

Pada gambar 4.9 terlihat bahwa hubungan antara aplikasi *Web Service E-Commerce* dengan situs *partner* melalui *web server* masing – masing aplikasi. Hubungan antara 2 *web server* itulah yang dikatakan sebagai proses *web service*. Dimana Aplikasi *Web Service* sebagai *provider entity* (penyedia) modul *web service* dan situs *partner* sebagai *requester entity* (peminta) modul *web service*.

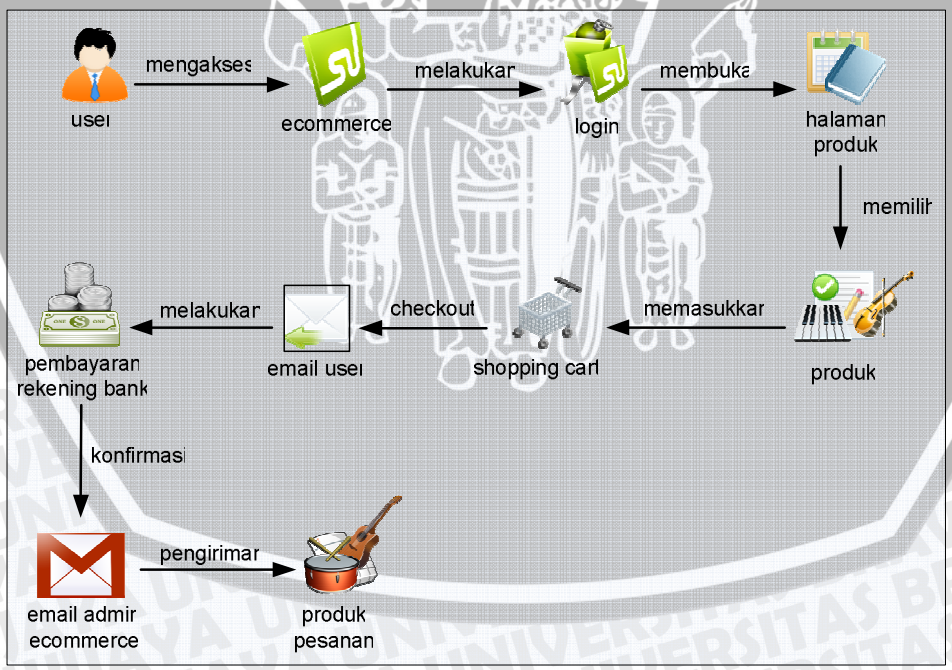
4.2.1.2. Blok Diagram Sistem

Blok diagram sistem untuk Aplikasi *Web Service E-Commerce* merupakan alur proses *user* dalam berinteraksi dengan sistem. Secara umum blok diagram sistem ditunjukkan pada gambar 4.11. Dimana pada blok diagram tersebut ditunjukkan interaksi general antara *user* dan aplikasi.



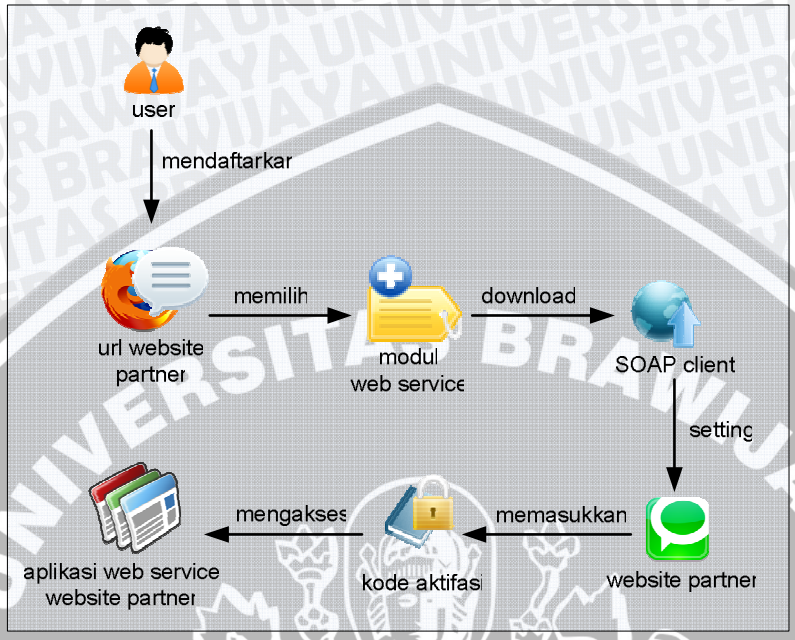
Gambar 4. 11 : Blok Diagram Aplikasi *Web Service E-Commerce* Secara Umum
Sumber : [Perancangan]

Blok diagram Aplikasi *Web Service E-Commerce* dibagi menjadi 2, yakni blok diagram *E-Commerce* dan blok diagram *Web Service*. Gambar 4.12 menunjukkan blok diagram *E-Commerce*.



Gambar 4. 12 : Blok Diagram Sistem *E-Commerce*
Sumber : [Perancangan]

Blok diagram sistem *E-Commerce* merupakan alur proses *user* dalam melakukan pembelanjaan produk dan pembayaran produk melalui sistem. Sedangkan blok diagram *web service* ditunjukkan pada gambar 4.13 di bawah ini.



Gambar 4. 13 : Blok Diagram Sistem *Web Service*
Sumber : [Perancangan]

Blok diagram sistem *web service* merupakan bagian dari Aplikasi *Web Service E-Commerce*. Dimana gambar 4.13 menunjukkan alur proses *user* untuk melakukan pendaftaran *website partner* dan penggunaan modul *web service* ke dalam aplikasi *patner*.

4.2.2. Perancangan Detail

Pada perancangan detil dilakukan spesifikasi dari tipe-tipe atribut, bagaimana fungsi-fungsi beroperasi, dan bagaimana suatu obyek berhubungan dengan obyek lainnya. Perancangan detil dimodelkan dengan *class diagram* dan *sequence diagram*.

4.2.2.1. *Class Diagram*

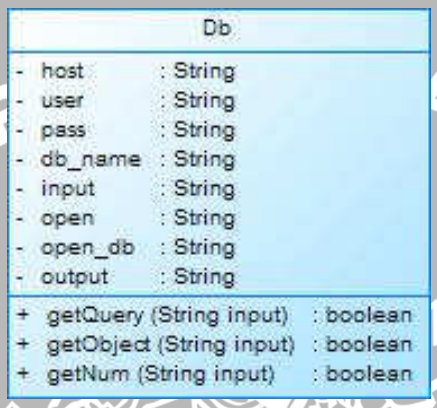
Klas adalah kumpulan dari obyek yang berbagi atribut, operasi relasi dan semantik yang sama. *Class diagram* adalah representasi berbentuk diagram dari klas - klas yang membentuk perangkat lunak.

4.2.2.1.1. *Class Diagram* untuk *Model*

Model merepresentasikan struktur data yang dibangun. Umumnya klas *model* berisi fungsi - fungsi yang membantu dalam mengelola, memasukkan, dan meng-*update* informasi pada basis data sistem. Klas *model* tersebut meliputi :

1. Klas `Db`.

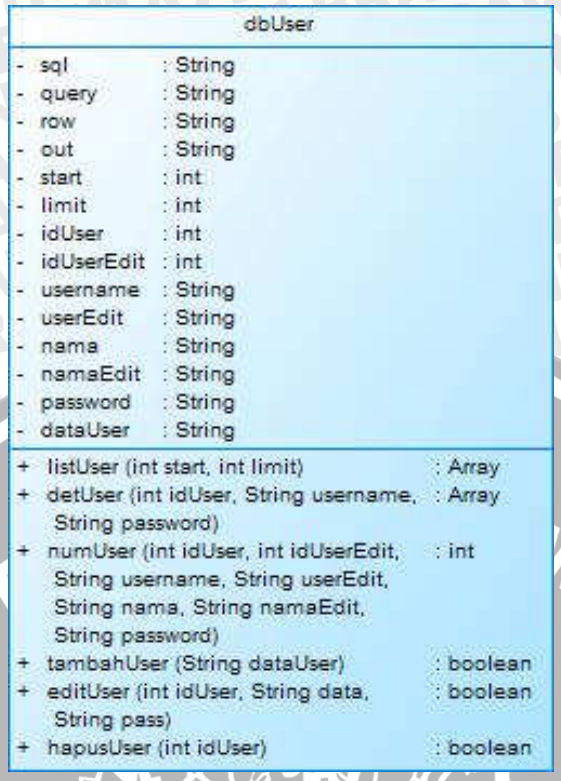
Klas `Db` memodelkan konfigurasi basis data. Klas ini bertugas untuk melakukan konektifitas terhadap *server* basis data.



Gambar 4. 14 : Kelas `Db`
Sumber : [Perancangan]

2. Klas `dbUser`.

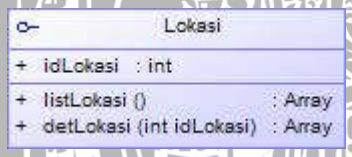
Klas `dbUser` memodelkan *user* yang berhak mengakses sistem. Klas ini bertugas mengambil *query* data *user*, melakukan *insert*, *update*, dan *delete*.



Gambar 4. 15 : Kelas dbUser
Sumber : [Perancangan]

3. Klas `Lokasi`

Interface `Lokasi` adalah interface yang berisi *method - method* abstrak yang berhubungan dengan klas `dbPropinsi` dan klas `dbKota`.



Gambar 4. 16 : Kelas `Lokasi`
Sumber : [Perancangan]

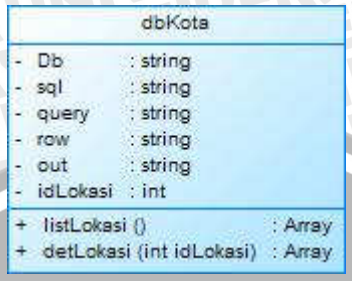
4. Klas `dbPropinsi`.

Klas `dbPropinsi` memodelkan data propinsi – propinsi yang ada di Indonesia. Klas ini melakukan *query* untuk *list* data dan *detail* data.

Gambar 4. 17 : Kelas `dbPropinsi`
Sumber : [Perancangan]

5. Klas `dbKota`.

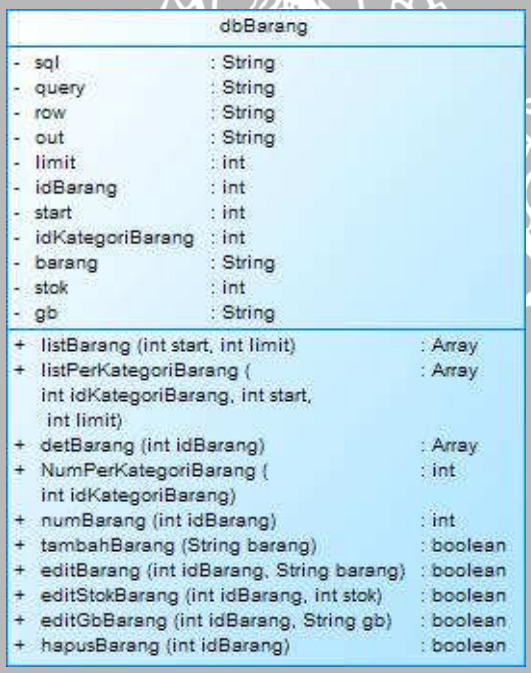
Klas `dbKota` memodelkan data kota – kota yang ada di Indonesia. Klas ini melakukan *query* untuk *list* data dan *detail* data.



Gambar 4. 18 : Kelas `dbKota`
Sumber : [Perancangan]

6. Klas `dbBarang`.

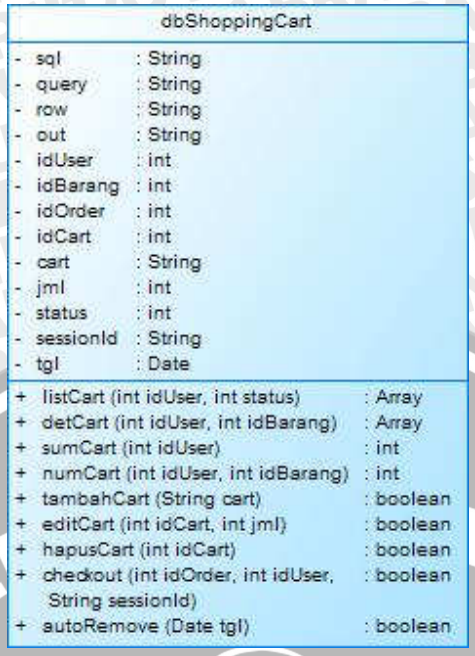
Klas `dbBarang` memodelkan produk – produk yang dijual pada aplikasi ini. Klas ini bertugas melakukan manipulasi data terhadap basis data barang.



Gambar 4. 19 : Kelas `dbBarang`
Sumber : [Perancangan]

7. Klas `dbShoppingCart`

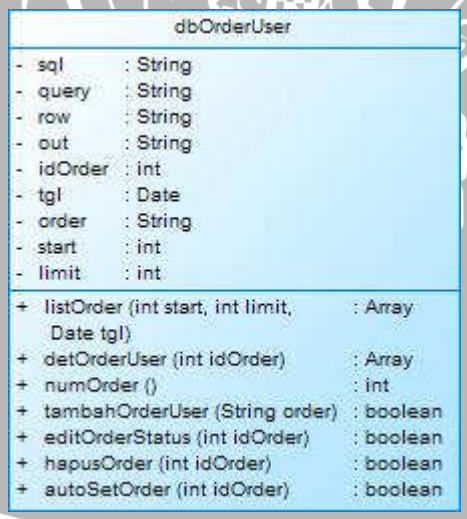
Klas `dbShoppingCart` memodelkan data *shopping cart*. Klas ini memproses perubahan data yang terjadi dalam basis data ketika *user* sedang berbelanja.



Gambar 4. 20 : Kelas `dbShoppingCart`
Sumber : [Perancangan]

8. Klas `dbOrderUser`.

Klas `dbOrderUser` memodelkan data order pemesanan barang. Klas ini mengatur data – data pemesanan barang yang terdapat dalam basis data.



Gambar 4. 21 : Kelas `dbOrderUser`
Sumber : [Perancangan]

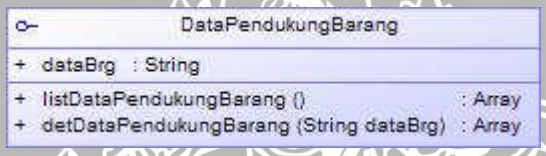
9. Klas `dbDataKirim`.

Klas `dbDataKirim` memodelkan data jasa kirim yang digunakan dalam pengiriman barang. Klas ini bertugas customerikan *query* tentang informasi jasa pengiriman barang yang berada di dalam basis data.

Gambar 4. 22 : Kelas `dbDataKirim`
Sumber : [Perancangan]

10. Klas `DataPendukungBarang`.

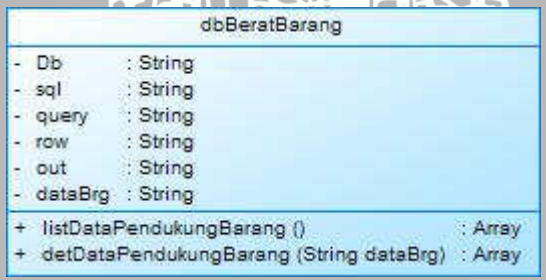
Klas `DataPendukungBarang` merupakan *interface* yang berisi *method* - *method* abstrak yang berhubungan dengan klas `dbBeratBarang` dan klas `dbKategoriBarang`.



Gambar 4. 23 : Kelas `DataPendukungBarang`
Sumber : [Perancangan]

11. Klas `dbBeratBarang`.

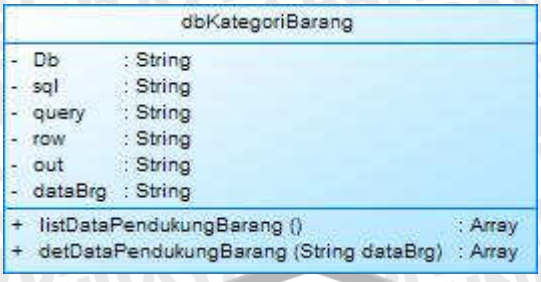
Klas `dbBeratBarang` memodelkan data berat barang. Klas ini bertugas melakukan *query* untuk jenis berat barang/produk.



Gambar 4. 24 : Kelas `dbBeratBarang`
Sumber : [Perancangan]

12. Klas `dbKategoriBarang`.

Klas `dbKategoriBarang` memodelkan data kategori barang/produk. Klas ini betugas melakukan *query* untuk jenis kategori barang/produk yang terdapat dalam basis data.



Gambar 4. 25 : Kelas dbKategoriBarang
Sumber : [Perancangan]

13. Klas dbBerita.

Klas dbBerita memodelkan data berita. Klas ini bertugas melakukan manipulasi terhadap data berita yang berada dalam basis data.

Gambar 4. 26 : Kelas dbBerita
Sumber : [Perancangan]

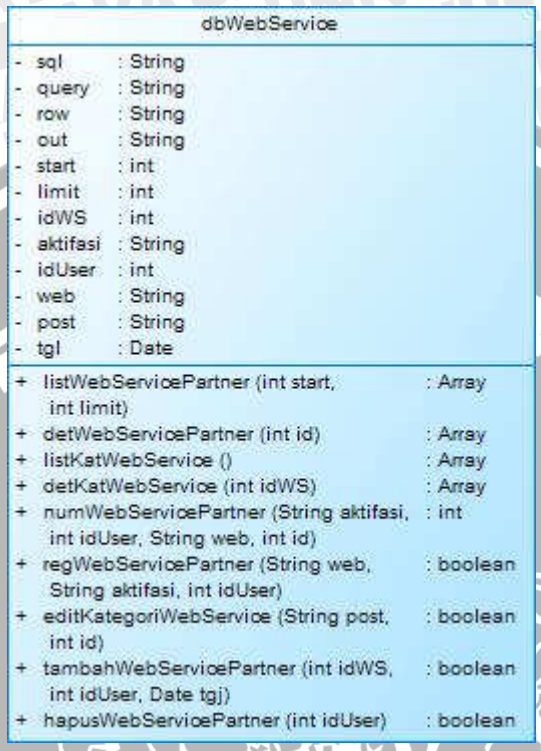
14. Klas dbKonten.

Klas dbKonten memodelkan data konten *website*. Klas ini berfungsi melakukan *query* dan memanipulasi data konten *website* yang berada dalam basis data.

Gambar 4. 27 : Kelas dbKonten
Sumber : [Perancangan]

15. Klas `dbWebService`.

Klas `dbWebService` memodelkan data – data kategori *web service* dan data *partner web service*. Klas ini bertugas mengatur administrasi data yang berhubungan dengan tabel `web_service` dan tabel `kategori_webservice`.



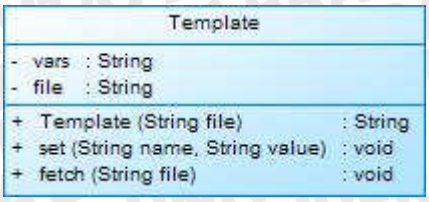
Gambar 4. 28 : Kelas `dbWebService`
Sumber : [Perancangan]

4.2.2.1.2. *Class Diagram* untuk *View*

View adalah informasi yang disajikan untuk *user*, berupa tampilan atau *user interface*. *View* umumnya berupa tampilan halaman *website* itu sendiri. Klas – klas yang merupakan *view* terdiri atas :

1. Klas `Template`

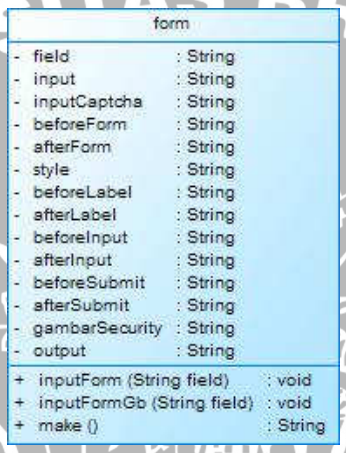
Klas `Template` merupakan klas yang mengatur tampilan halaman *website* secara keseluruhan. Klas ini hanya bertugas menampilkan data berupa halaman HTML.



Gambar 4. 29 : Kelas `Template`
Sumber : [Perancangan]

2. Klas `form`

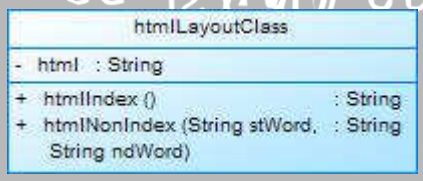
Klas `form` merupakan klas yang men-*generate form* HTML. Klas ini mengatur jumlah *field* dan bentuk *input* data dari *form* yang digunakan.



Gambar 4. 30 : Kelas `form`
Sumber : [Perancangan]

3. Klas `htmlLayoutClass`

Klas `htmlLayoutClass` merupakan klas yang mengatur susunan data pada halaman HTML. Klas ini bertugas mengatur layout halaman *website* dimana membaginya menjadi halaman *index* dan halaman *non-index*.



Gambar 4. 31 : Kelas `htmlLayoutClass`
Sumber : [Perancangan]

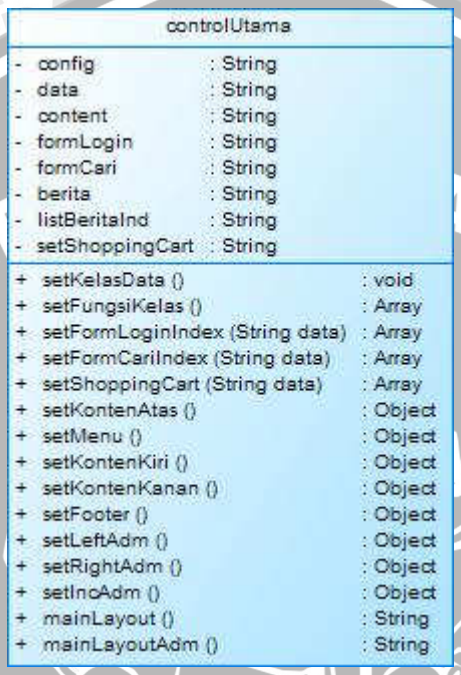
4.2.2.1.3. *Class Diagram* untuk *Controller*

Controller bertugas sebagai penghubung antara *model*, *view*, dan beberapa *resource* lainnya yang dibutuhkan untuk memproses HTTP *request* dan untuk *generate* sebuah halaman *website*. Klas – klas yang merupakan bagian *controller*

pada Aplikasi *Web Service E-Commerece* terdiri atas :

1. Klas `controlUtama`

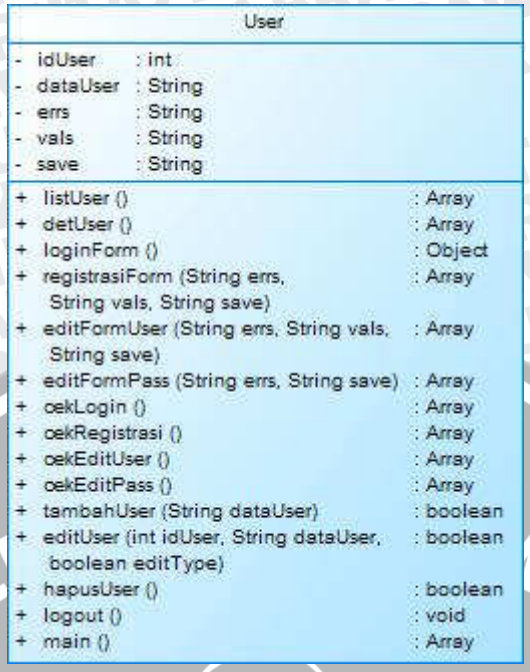
Klas `controlUtama` merupakan klas controller yang berperan penting dalam sistem ini. Klas ini mengatur hubungan antara semua klas – klas yang ada pada bagian *model*, *view*, dan *controller* itu sendiri.



Gambar 4. 32 : Kelas `controlUtama`
Sumber : [Perancangan]

2. Klas `User`

Klas `User` merupakan klas yang mengatur hubungan antara *user* dengan sistem. Klas ini memiliki referensi data pada klas `dbUser` di bagian *model*.



Gambar 4. 33 : Kelas User
Sumber : [Perancangan]

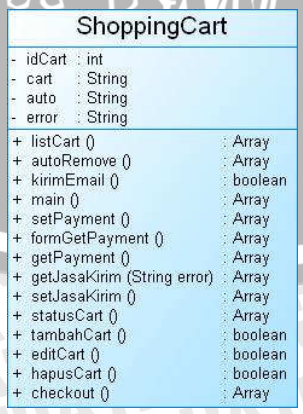
3. Klas `Barang`

Klas `Barang` merupakan klas yang mengatur tentang manipulasi data barang. Klas ini memiliki referensi data pada klas `dbBarang` di bagian *model*.

Gambar 4. 34 : Kelas `Barang`
Sumber : [Perancangan]

4. Klas `ShoppingCart`

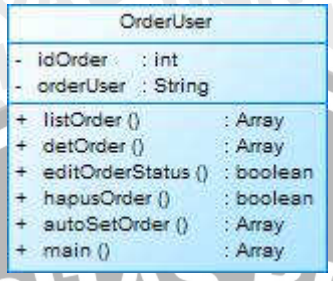
Klas `ShoppingCart` merupakan klas yang mengatur penambahan, pengeditan, penghapusan, pengiriman *email*, dan penyimpanan data *shopping cart*. Klas ini berhubungan dengan klas `dbShoppingCart` pada bagian *model*.



Gambar 4. 35 : Kelas `ShoppingCart`
Sumber : [Perancangan]

5. Klas `OrderUser`

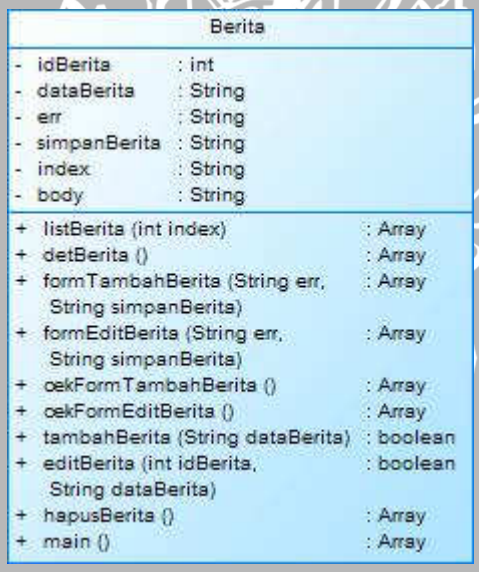
Klas `OrderUser` merupakan klas yang mengatur data order *user* yang telah berbelanja di *website*. Klas ini berhubungan dengan klas `dbOrderUser` di bagian *model*.



Gambar 4. 36 : Kelas `OrderUser`
Sumber : [Perancangan]

6. Klas `Berita`

Klas `Berita` merupakan klas yang mengatur data berita. Klas ini berhubungan dengan klas `dbBerita` di bagian *model*.



Gambar 4. 37 : Kelas `Berita`
Sumber : [Perancangan]

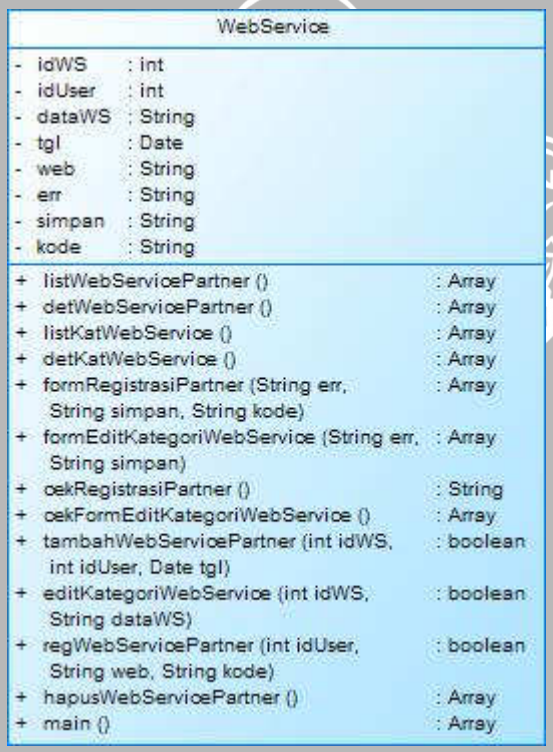
7. Klas `Konten`

Klas `Konten` merupakan klas yang mengatur data konten *website*. Klas ini berhubungan dengan klas `dbKonten` pada bagian *model*.

Gambar 4. 38 : Kelas `Konten`
Sumber : [Perancangan]

8. Klas `WebService`

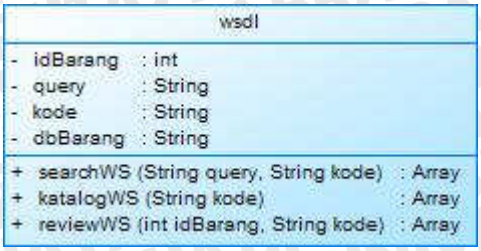
Klas `WebService` merupakan klas yang mengatur data web service. Klas ini berhubungan dengan klas pada `dbWebService` pada bagian *model*.



Gambar 4. 39 : Kelas `WebService`
Sumber : [Perancangan]

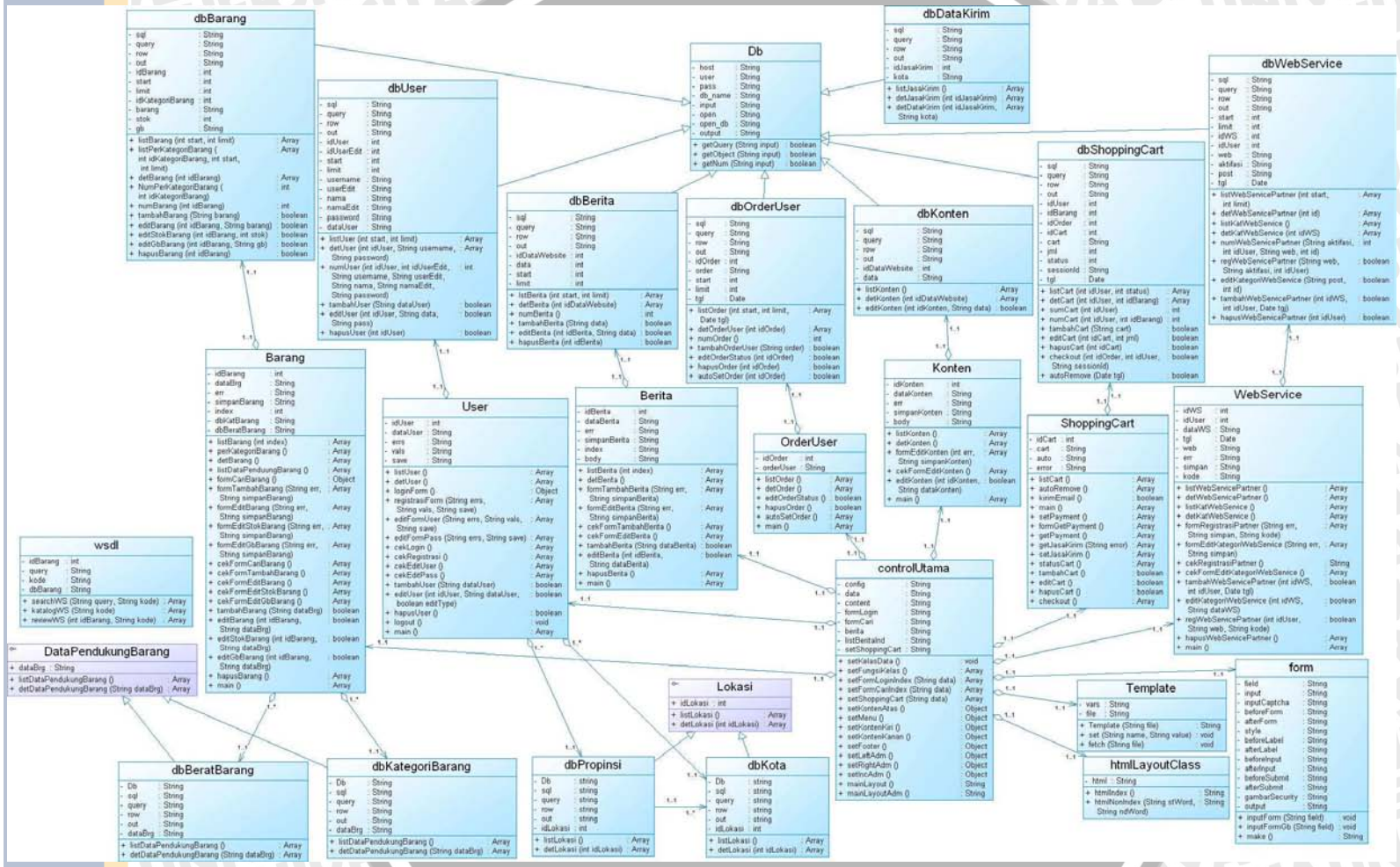
9. Klas `WSDL`

Klas `WSDL` ini merupakan klas yang mengatur data – data atau fungsi yang digunakan dalam dokumen WSDL. Klas ini merupakan referensi untuk SOAP *server*.



Gambar 4. 40 : Kelas WSDL
Sumber : [Perancangan]

Semua klas diagram diatas merepresentasikan segala proses yang terjadi pada Aplikasi *Web Service E-Commerce*. Gambar 4.41 menunjukkan diagram klas keseluruhan dari Aplikasi *Web Service E-Commerce*.



Gambar 4. 41 : Diagram Kelas Aplikasi *Web Service E-Commerce*
Sumber : [Perancangan]

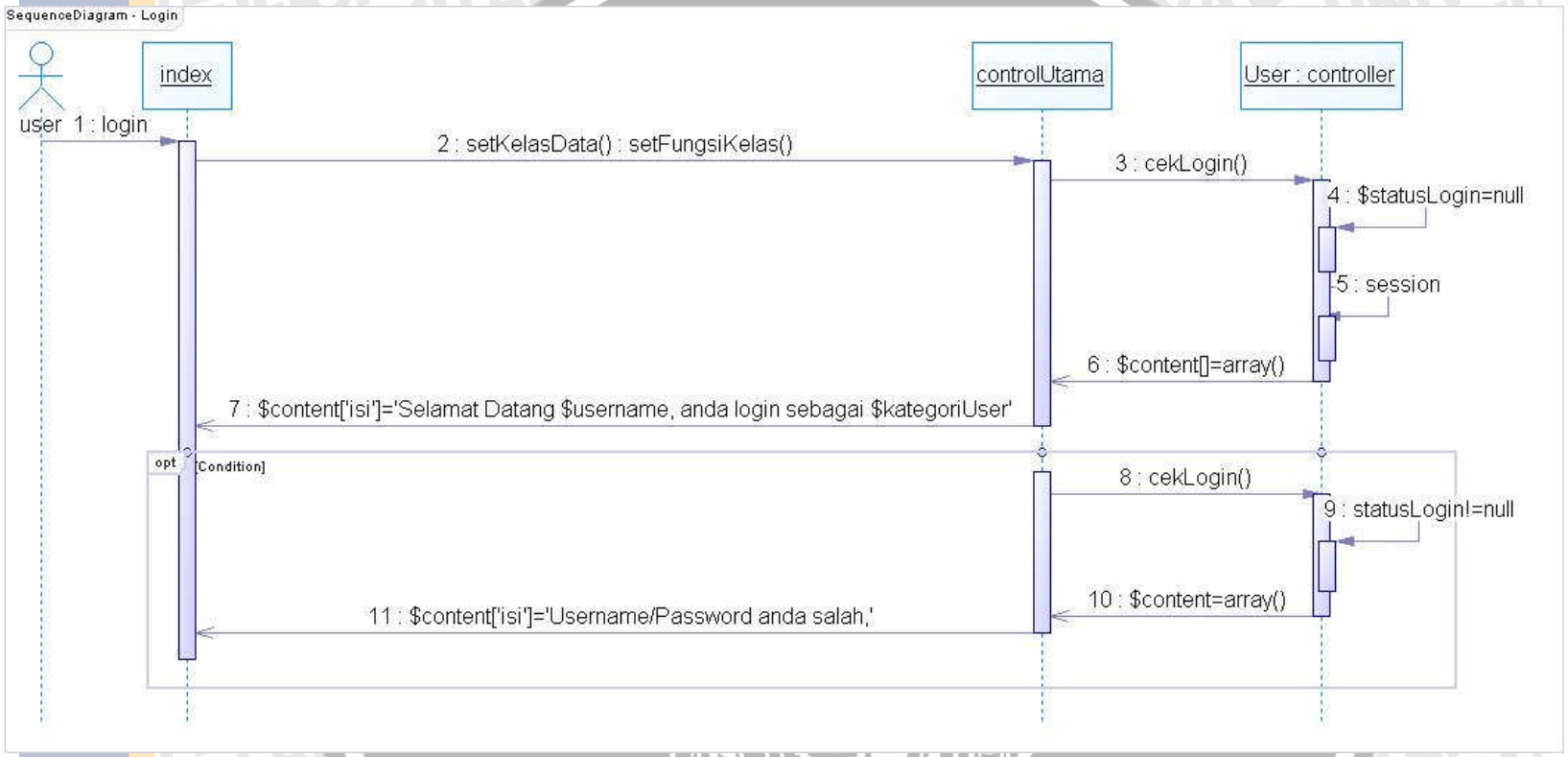
4.2.2.2. *Sequence Diagram*

Hubungan antar klas yang digambarkan pada diagram klas termasuk dalam pemodelan statis, tanpa disertai dengan gambaran aliran proses atau data antarklas yang satu dengan klas yang lain. *Sequence diagram* digunakan untuk memodelkan aliran jalannya proses yang ditunjukkan dengan interaksi antar objek atau klas, dan disusun berdasarkan urutan waktu. *Sequence diagram* disusun dengan mengambil acuan pada *use case* dan klas - klas yang telah diturunkan untuk membentuk fungsionalitas yang digambarkan pada *use case* tersebut.

Subbab ini tidak memodelkan keseluruhan *sequence diagram* untuk tiap *use case*, tetapi hanya beberapa contoh *sequence diagram* untuk *use case* tertentu. Subbab ini memodelkan *sequence diagram* untuk *use case* - *use case* berikut :

4.2.2.2.1. *Use case Login*

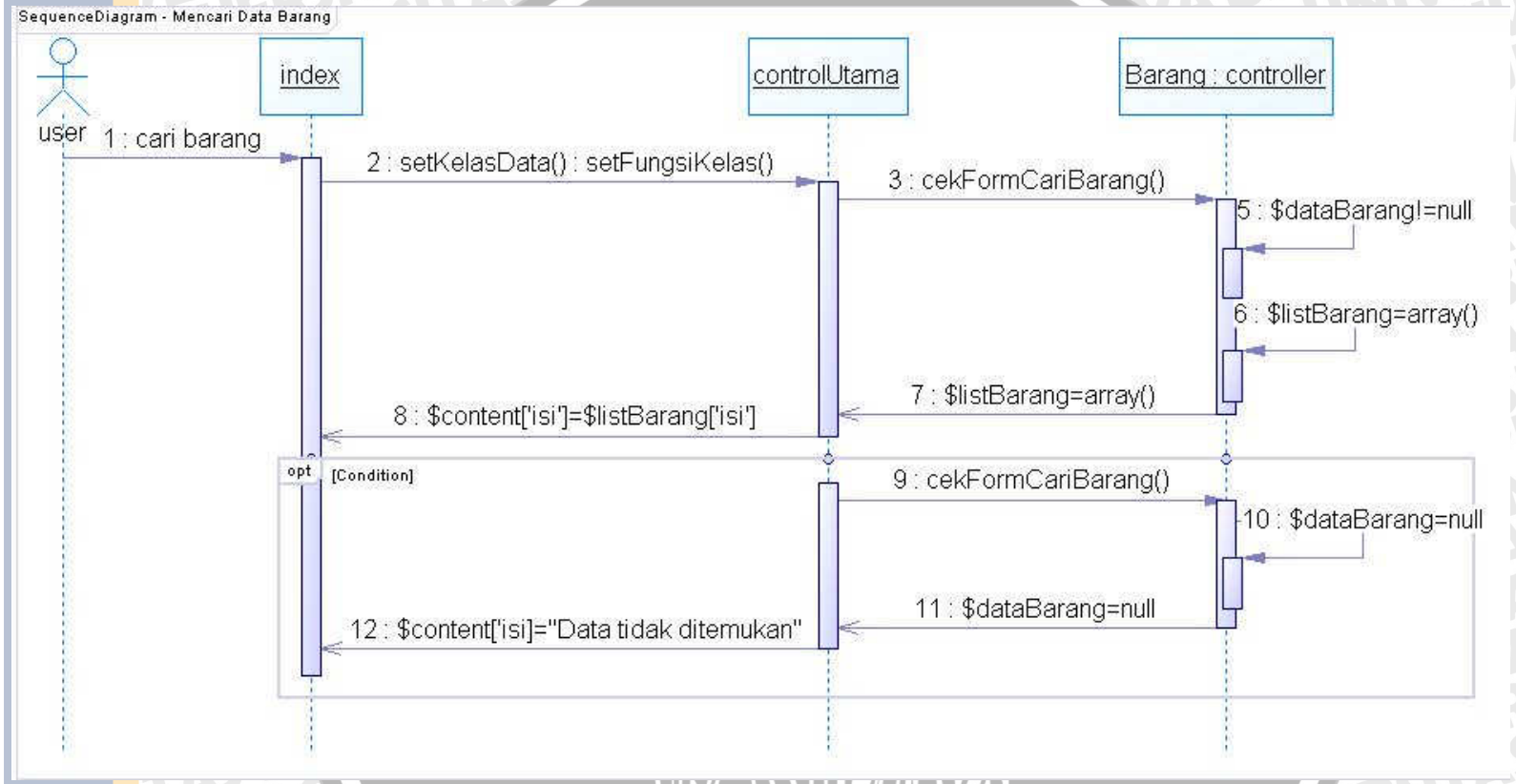
Gambar 4.42 merupakan *sequence diagram* untuk *use case Login*. Proses *login* menggunakan aktor *user* dan pembentukan beberapa objek. Objek – objek tersebut adalah `index`, `controlUtama` dan `User`.



Gambar 4. 42 : Sequence Diagram untuk use case *Login*
Sumber : [Perancangan]

4.2.2.2.2. *Use case* Mencari Data Barang

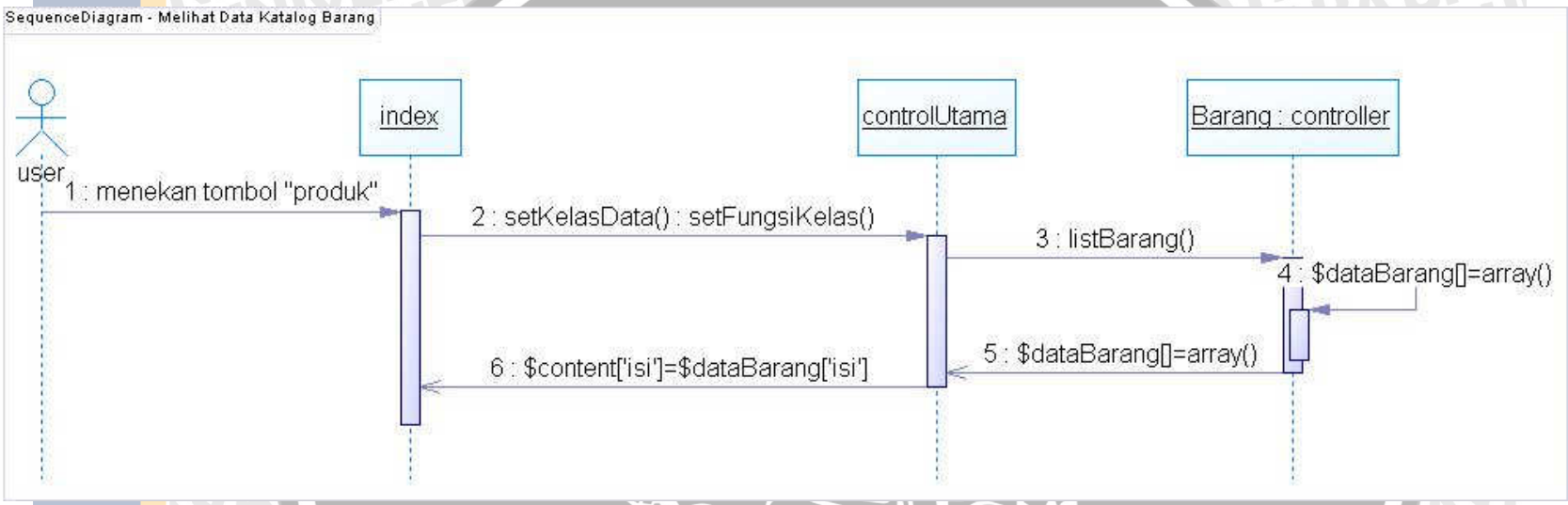
Gambar 4.43 merupakan *sequence diagram* untuk *use case* Mencari Data Barang. Proses pancarian data barang ini menggunakan aktor *user* dan pembentukan beberapa objek. Objek – objek tersebut adalah `index`, `controlUtama` dan `Barang`.



Gambar 4. 43 : *Sequence Diagram* untuk *use case* Mencari Data Barang
Sumber : [Perancangan]

4.2.2.2.3. *Use case* Melihat Data Katalog Barang

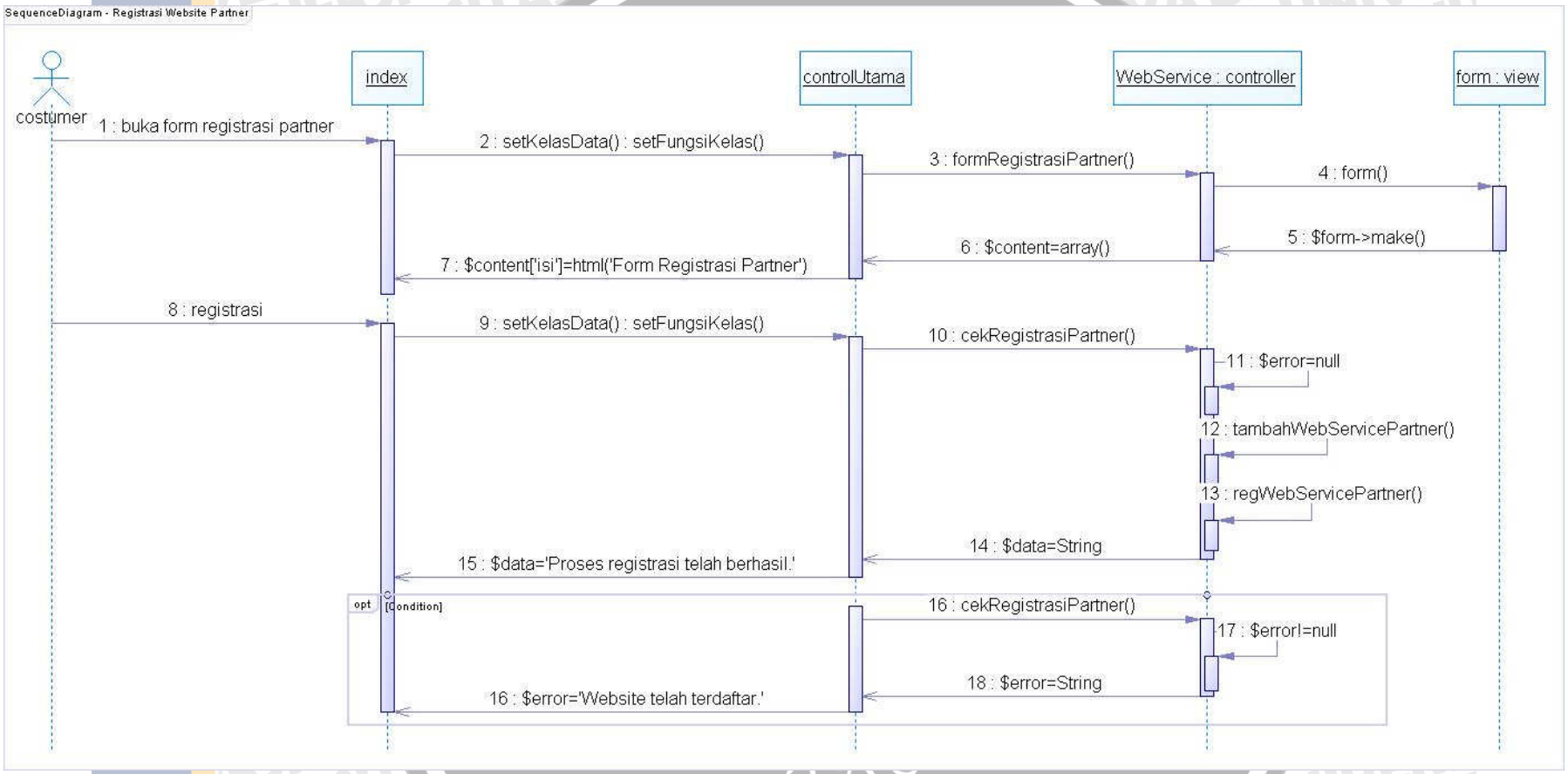
Gambar 4.44 merupakan *sequence diagram* untuk *use case* Melihat Data Katalog Barang. Proses ini menggunakan aktor *user* dan pembentukan beberapa objek. Objek – objek tersebut adalah `index`, `controlUtama` dan `Barang`.



Gambar 4. 44 : *Sequence Diagram* untuk *use case* Melihat Data Katalog Barang
Sumber : [Perancangan]

4.2.2.2.4. *Use case* Registrasi *Website Partner*

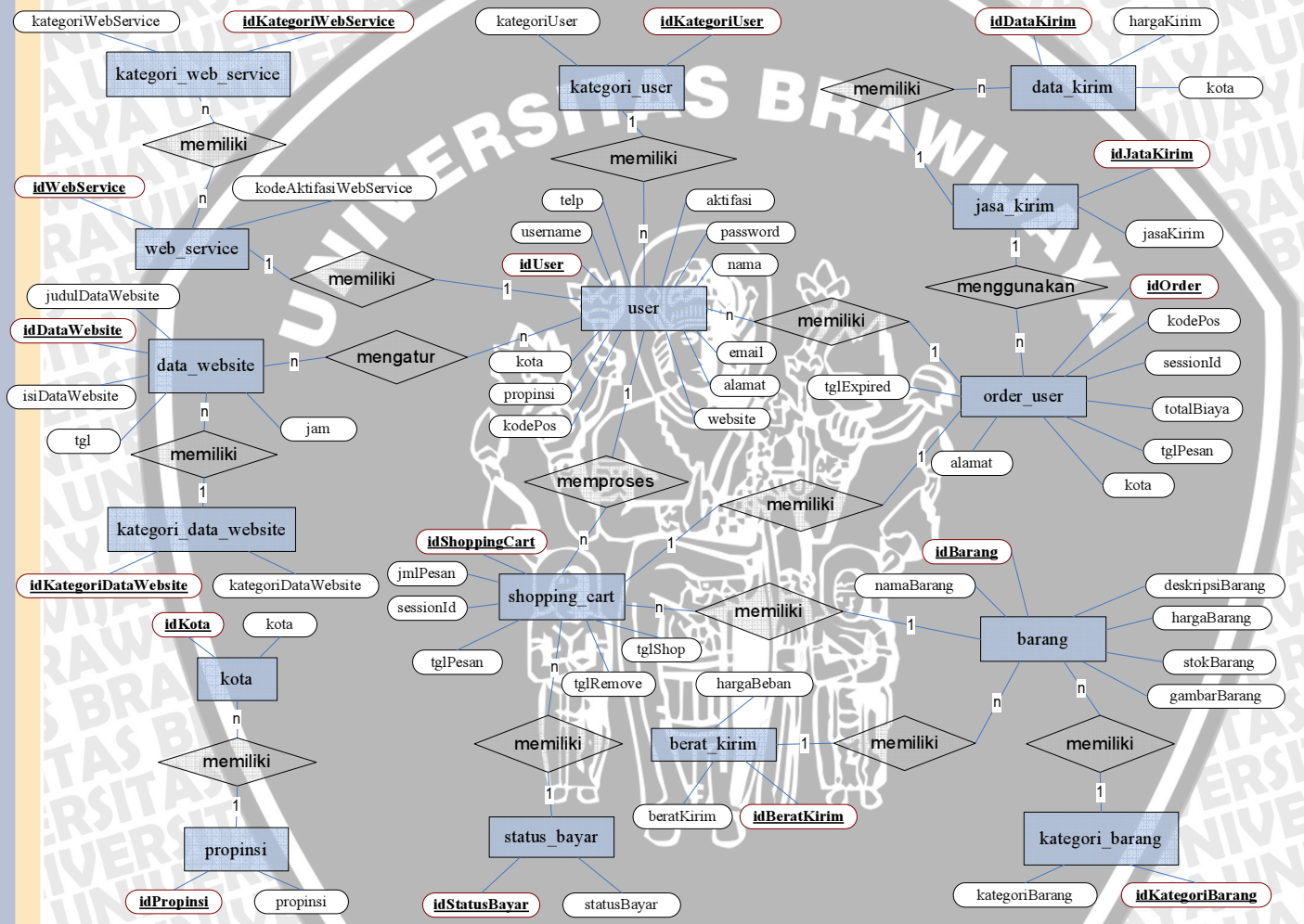
Gambar 4.45 merupakan *sequence diagram* untuk *use case* Registrasi *Website Partner*. Proses ini menggunakan aktor *customer* dan pembentukan beberapa objek. Objek – objek tersebut adalah `index`, `controlUtama`, `WebService`, dan `form`.



Gambar 4. 45 : *Sequence Diagram* untuk *use case* Registrasi *Website Partner*
Sumber : [Perancangan]

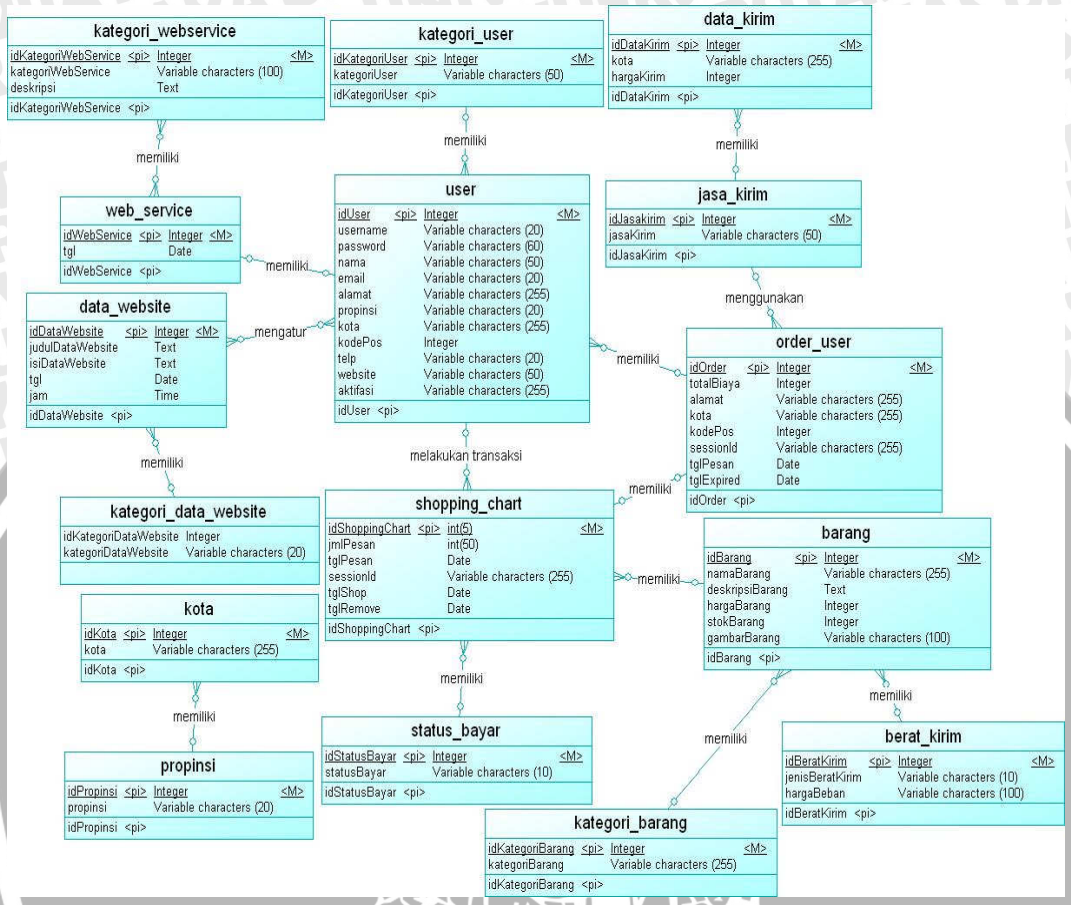
4.2.2.3. Perancangan Basis Data

Aplikasi *Web Service E-Commerce* memerlukan sebuah basis data. Basis data ini berfungsi sebagai tempat menyimpan data. Pemodelan basis data dapat dilakukan dengan menggunakan *Entity-Relationship diagram*, *conceptual data model*, dan *physical data model*. Rancangan basis data yang berupa *Entity-Relationship diagram* ditunjukkan pada gambar 4.46.



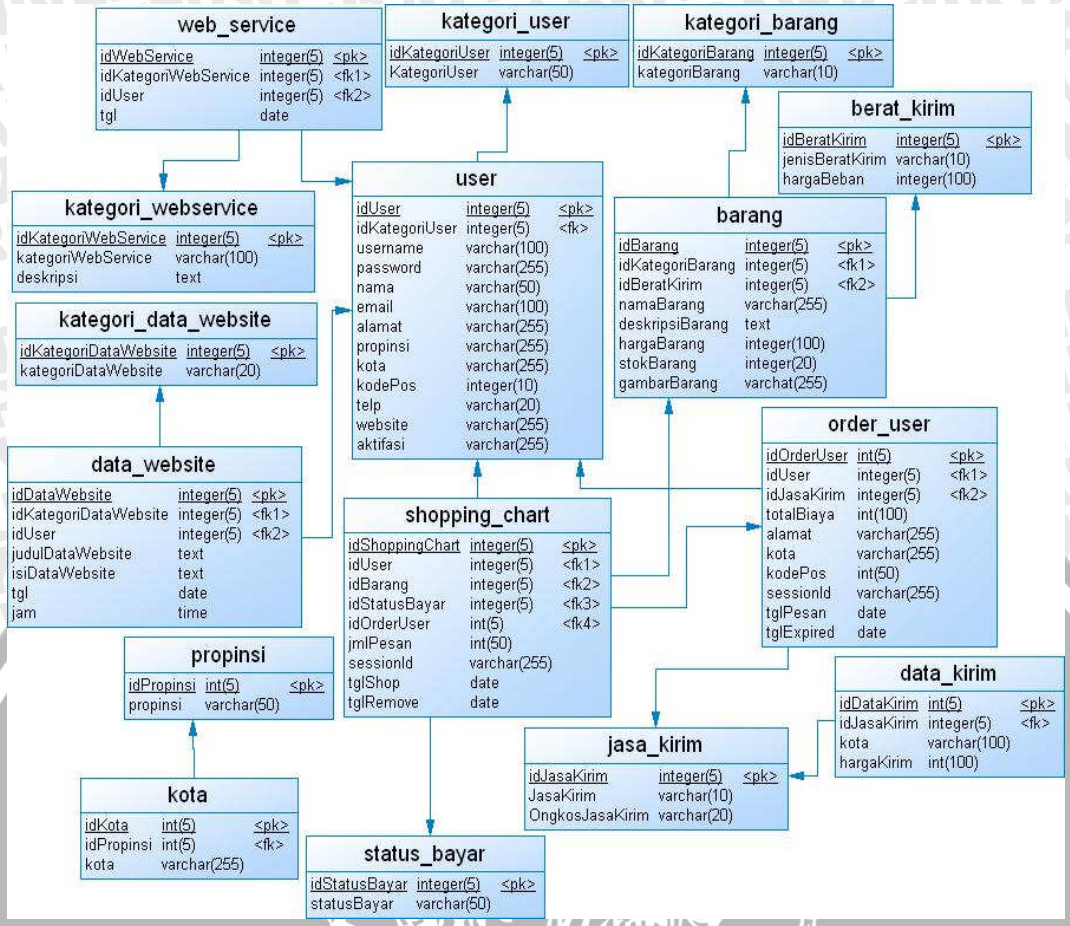
Gambar 4. 46 : *Entity Relationship Diagram*
Sumber : [Perancangan]

Conceptual data model merupakan konsep model basis data Aplikasi *Web Service E-Commerce*. Menunjukkan *conceptual data model* dari *aplikasi Web Service E-Commerce* dapat ditunjukkan pada gambar 4.47.



Gambar 4. 47 : *Conceptual Data Model*
Sumber : [Perancangan]

Physical data model menggambarkan hubungan tabel secara fisik. Gambar *physical data model* basis data aplikasi *Web Service E-Commerce* dapat dilihat pada gambar 4.48.



Gambar 4. 48 : *Physical Data Model*
Sumber : [Perancangan]

4.2.2.3.1. *Data Object Description*

Data Object Description menjelaskan secara rinci mengenai atribut-atribut yang dimiliki oleh masing-masing tabel yang ada pada basis data Aplikasi *Web Service E-Commerce*. Entitas tabel yang ada pada basis data ini masing-masing memiliki struktur tabel basis data.

Rancangan struktur tabel terdiri atas 5 (lima) kolom, yaitu *key*, *column name*, *data type*, *size*, *allow*. Kolom *key* menjelaskan status *field* yang dijadikan *Primary Key* (PK) atau *Foreign Key* (FK). *Column name* berisi nama-nama *field* yang akan digunakan pada masing-masing tabel. *Data type* menjelaskan tipe data yang digunakan oleh setiap *field*. *Size* berisi panjang dari tipe data yang digunakan. *Allow* berarti data yang dimasukan apakah boleh bernilai *Null* (kosong). N berarti tidak diperbolehkan bernilai *null*. Y berarti boleh bernilai *null*.

1. *Data Object Description* untuk Tabel `user`

Tabel 4. 48 : *Data Object Description* untuk Tabel `user`

KEY	COLUMN NAME	DATA TYPE	SIZE	ALLOW
PK	idUser	integer	5	N
FK	idKategoriUser	integer	5	N
	username	varchar	100	N
	password	varchar	50	N
	nama	varchar	255	N
	email	varchar	100	N
	alamat	varchar	255	N
	propinsi	varchar	255	N
	kota	varchar	255	N
	kodePos	integer	10	N
	telp	varchar	20	N
	website	varchar	255	N
	aktifasi	varchar	255	N

Sumber : [Perancangan]

Tabel `user` terdiri dari 13 *field*. *Field* `idUser` merupakan *primary key* dari tabel `user`. *Field* `idKategoriUser` merupakan *foreign key* dari tabel `kategori_user`. Semua *field* tidak boleh kosong. Tabel `user` berisi data - data *user* yang terdaftar pada Aplikasi *Web Service E-Commerce*.

2. *Data Object Description* untuk Tabel `kategori_user`

Tabel 4. 49 : *Data Object Description* untuk Tabel `kategori_user`

KEY	COLUMN NAME	DATA TYPE	SIZE	ALLOW
PK	idKategoriUser	integer	5	N
	kategoriUser	varchar	50	N

Sumber : [Perancangan]

Tabel `kategori_user` terdiri dari 2 *field*. *Field* `idKategoriUser` merupakan *primary key* dari tabel `kategori_user`. Semua *field* tidak boleh kosong. Tabel `kategori_user` berisi jenis – jenis kategori *user* yang berhak mengakses Aplikasi *Web Service E-Commerce*.

3. *Data Object Description* untuk Tabel `barang`

Tabel 4. 50 : *Data Object Description* untuk Tabel `barang`

KEY	COLUMN NAME	DATA TYPE	SIZE	ALLOW
PK	idBarang	integer	5	N
FK	idKategoriBarang	integer	5	N
FK	idBeratKirim	integer	5	N
	namaBarang	varchar	255	N
	deskripsiBarang	text		N
	hargaBarang	integer	100	N
	stokBarang	integer	20	N
	gambarBarang	varchar	100	N

Sumber : [Perancangan]

Tabel `barang` terdiri dari 8 *field*. *Field* `idBarang` merupakan *primary key* dari tabel `barang`. *Field* `idKategoriBarang` merupakan *foreign key* dari tabel `kategori_barang`. *Field* `idBeratBarang` merupakan *foreign key* dari tabel `berat_kirim`. Semua *field* tidak boleh kosong. Tabel `barang` berisi data - data tentang produk yang dijual pada Aplikasi *Web Service E-Commerce*.

4. *Data Object Description* untuk Tabel `kategori_barang`

Tabel 4. 51 : *Data Object Description* untuk Tabel `kategori_barang`

KEY	COLUMN NAME	DATA TYPE	SIZE	ALLOW
PK	idKategoriBarang	integer	5	N
	kategoriBarang	varchar	255	N

Sumber : [Perancangan]

Tabel `kategori_barang` terdiri dari 2 *field*. *Field* `idKategoriBarang` merupakan *primary key* dari tabel `kategori_barang`. Semua *field* tidak boleh kosong. Tabel `kategori_barang` berisi tentang jenis – jenis produk yang dijual pada Aplikasi *Web Service E-Commerce*.

5. *Data Object Description* untuk Tabel `berat_kirim`

Tabel 4. 52 : *Data Object Description* untuk Tabel `berat_kirim`

KEY	COLUMN NAME	DATA TYPE	SIZE	ALLOW
PK	idBeratKirim	integer	5	N
	jenisBeratKirim	varchar	10	N
	hargaBeban	integer	100	N

Sumber : [Perancangan]

Tabel berat_kirim terdiri dari 3 *field*. *Field* idBeratKirim merupakan *primary key* dari tabel berat_kirim. Semua *field* tidak boleh kosong. Tabel berat_kirim berisi tentang data jenis – jenis berat produk untuk proses pengirimiman barang.

6. *Data Object Description* untuk Tabel data_kirim

Tabel 4. 53 : *Data Object Description* untuk Tabel data_kirim

KEY	COLUMN NAME	DATA TYPE	SIZE	ALLOW
PK	idDataKirim	integer	5	N
FK	idJasaKirim	integer	5	N
	kota	varchar	100	N
	hargaKirim	integer	100	N

Sumber : [Perancangan]

Tabel data_kirim terdiri dari 4 *field*. *Field* idDataKirim merupakan *primary key* dari tabel data_kirim. *Field* idJasaKirim merupakan *foreign key* dari tabel jasa_kirim. Semua *field* tidak boleh kosong. Tabel data_kirim berisi data – data harga pengiriman barang untuk tiap – tiap kota berdasarkan jenis jasa kirimnya.

7. *Data Object Description* untuk Tabel jasa_kirim

Tabel 4. 54 : *Data Object Description* untuk Tabel jasa_kirim

KEY	COLUMN NAME	DATA TYPE	SIZE	ALLOW
PK	idJasaKirim	integer	10	N
	jasaKirim	varchar	50	N

Sumber : [Perancangan]

Tabel jasa_kirim terdiri dari 2 *field*. *Field* idJasaKirim merupakan *primary key* dari tabel jasa_kirim. Semua *field* tidak boleh kosong. Tabel jasa_kirim berisi data – data jenis jasa kirim yang disediakan oleh Aplikasi *Web Service E-Commerce* untuk pengiriman barang kepada *customer*.

8. *Data Object Description* untuk Tabel order_user

Tabel 4. 55 : *Data Object Description* untuk Tabel order_user

KEY	COLUMN NAME	DATA TYPE	SIZE	ALLOW
PK	idOrder	integer	5	N

FK	idUser	integer	5	N
FK	idJasaKirim	integer	5	N
	totalBiaya	integer	100	N
	alamat	varchar	255	N
	kota	varchar	255	N
	kodePos	integer	50	N
	sessionId	varchar	255	N
	tglPesan	date		N
	tglExpired	date		N

Sumber : [Perancangan]

Tabel `order_user` terdiri dari 10 *field*. *Field* `idOrder` merupakan *primary key* dari tabel `order_user`. *Field* `idUser` merupakan *foreign key* dari tabel `user`. *Field* `idJasaKirim` merupakan *foreign key* dari tabel `jasa_kirim`. Semua *field* tidak boleh kosong. Tabel `order_user` berisi data – data order untuk pemesanan barang oleh *customer* pada Aplikasi *Web Service E-Commerce*.

9. *Data Object Description* untuk Tabel `shopping_cart`

Tabel 4. 56 : *Data Object Description* untuk Tabel `shopping_cart`

KEY	COLUMN NAME	DATA TYPE	SIZE	ALLOW
PK	idShoppingCart	integer	5	N
FK	idUser	integer	5	N
FK	idBarang	integer	5	N
FK	idStatusBayar	integer	5	N
FK	idOrder	integer	5	N
	jmlPesan	integer	50	N
	sessionId	varchar	255	N
	tglShop	date		N
	tglRemove	date		N

Sumber : [Perancangan]

Tabel `shopping_cart` terdiri dari 9 *field*. *Field* `idShoppingCart` merupakan *primary key* dari tabel `shopping_cart`. *Field* `idUser` merupakan *foreign key* dari tabel `user`. *Field* `idBarang` merupakan *foreign key* dari tabel `barang`. *Field* `idStatusBayar` merupakan *foreign key* dari tabel `status_bayar`. *Field* `idOrder` merupakan *foreign key* dari tabel `order_user`. Semua *field* tidak boleh kosong. Tabel `shopping_cart` berisi data – data barang

yang akan dibeli oleh *customer* pada Aplikasi *Web Service E-Commerce*.

10. *Data Object Description* untuk Tabel `status_bayar`

Tabel 4. 57 : *Data Object Description* untuk Tabel `status_bayar`

KEY	COLUMN NAME	DATA TYPE	SIZE	ALLOW
PK	idStatusBayar	integer	5	N
	statusBayar	varchar	50	N

Sumber : [Perancangan]

Tabel `status_bayar` terdiri dari 2 *field*. *Field* `idStatusBayar` merupakan *primary key* dari tabel `status_bayar`. Semua *field* tidak boleh kosong. Tabel `status_bayar` berisi jenis - jenis status order pemesanan barang yang dibeli oleh *customer* pada Aplikasi *Web Service E-Commerce*.

11. *Data Object Description* untuk Tabel `data_website`

Tabel 4. 58 : *Data Object Description* untuk Tabel `data_website`

KEY	COLUMN NAME	DATA TYPE	SIZE	ALLOW
PK	idDataWebsite	integer	5	N
FK	idKategoriDataWebsite	integer	5	N
FK	idUser	integer	5	N
	judulDataWebsite	text		N
	isiDataWebsite	text		N
	tgl	date		N
	jam	time		N

Sumber : [Perancangan]

Tabel `data_website` terdiri dari 7 *field*. *Field* `idDataWebsite` merupakan *primary key* dari tabel `data_website`. *Field* `idKategoriDataWebsite` merupakan *foreign key* dari tabel `kategori_data_website`. *Field* `idUser` merupakan *foreign key* dari tabel `user`. Semua *field* tidak boleh kosong. Tabel `data_website` berisi data – data konten dinamis atau konten statis pada Aplikasi *Web Service E-Commerce*.

12. *Data Object Description* untuk Tabel `kategori_data_website`

Tabel 4. 59 : *Data Object Description* untuk Tabel `kategori_data_website`

KEY	COLUMN NAME	DATA TYPE	SIZE	ALLOW
PK	idKategoriDataWebsite	integer	5	N
	kategoriDataWebsite	varchar	20	N

Sumber : [Perancangan]

Tabel `kategori_data_website` terdiri dari 2 *field*. *Field* `idKategoriDataWebsite` merupakan *primary key* dari tabel `kategori_data_website`. Semua *field* tidak boleh kosong. Tabel `kategori_data_website` berisi jenis – jenis data konten pada Aplikasi *Web Service E-Commerce*.

13. *Data Object Description* untuk Tabel `web_service`

Tabel 4. 60 : *Data Object Description* untuk Tabel `web_service`

KEY	COLUMN NAME	DATA TYPE	SIZE	ALLOW
PK	idWebService	integer	5	N
FK	idKategoriWebService	integer	5	N
FK	idUser	integer	5	N
	tgl	date		N

Sumber : [Perancangan]

Tabel `web_service` terdiri dari 4 *field*. *Field* `idWebService` merupakan *primary key* dari tabel `web_service`. *Field* `idKategoriWebService` merupakan *foreign key* dari tabel `kategori_webservice`. *Field* `idUser` merupakan *foreign key* dari tabel `user`. Semua *field* tidak boleh kosong. Tabel `web_service` berisi data – data *user* yang menggunakan fasilitas *web service* pada Aplikasi *Web Service E-Commerce*.

14. *Data Object Description* untuk Tabel `kategori_webservice`

Tabel 4. 61 : *Data Object Description* untuk Tabel `kategori_webservice`

KEY	COLUMN NAME	DATA TYPE	SIZE	ALLOW
PK	idKategoriWebService	integer	5	N
	kategoriWebService	varchar	100	N
	deskripsi	text		N

Sumber : [Perancangan]

Tabel `kategori_webservice` terdiri dari 3 *field*. *Field* `idKategoriWebService` merupakan *primary key* dari tabel `kategori_webservice`. Semua *field* tidak boleh kosong. Tabel `kategori_webservice` berisi jenis – jenis fasilitas *web service* pada Aplikasi *Web Service E-Commerce*.

15. Data Object Description untuk Tabel `propinsi`

Tabel 4. 62 : *Data Object Description* untuk Tabel `propinsi`

KEY	COLUMN NAME	DATA TYPE	SIZE	ALLOW
PK	idPropinsi	integer	5	N
	propinsi	varchar	50	N

Sumber : [Perancangan]

Tabel `propinsi` terdiri dari 2 *field*. *Field* `idPropinsi` merupakan *primary key* dari tabel `propinsi`. Semua *field* tidak boleh kosong. Tabel `propinsi` berisi nama – nama propinsi se-Indonesia.

16. Data Object Description untuk Tabel `kota`

Tabel 4. 63 : *Data Object Description* untuk Tabel `kota`

KEY	COLUMN NAME	DATA TYPE	SIZE	ALLOW
PK	idKota	integer	5	N
FK	idPropinsi	integer	5	N
	kota	varchar	255	N

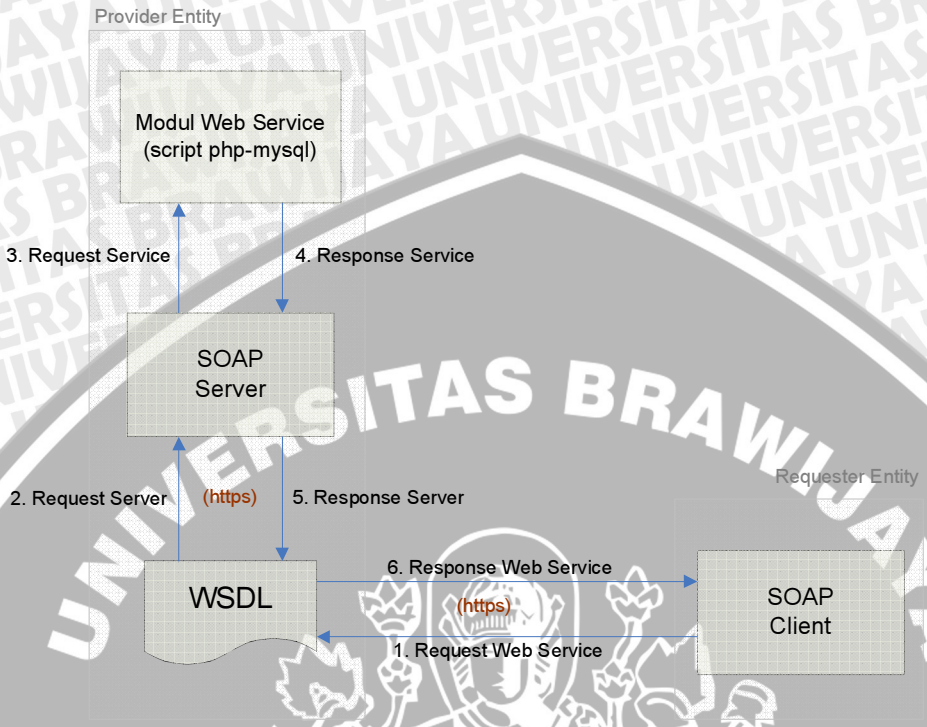
Sumber : [Perancangan]

Tabel `kota` terdiri dari 3 *field*. *Field* `idKota` merupakan *primary key* dari tabel `kota`. *Field* `idPropinsi` merupakan *foreign key* dari tabel `propinsi`. Semua *field* tidak boleh kosong. Tabel `kota` berisi nama – nama kota se-Indonesia berdasarkan propinsinya.

4.2.2.4. Perancangan *Web Service*

Proses *web service* terjadi antara *provider entity* dan *requester entity*. Proses ini merupakan permintaan dan respon terhadap dokumen WSDL (*Web Service Describe Language*) dengan menggunakan protokol HTTPS. Dokumen WSDL berisi fungsi – fungsi aplikasi atau modul aplikasi yang dapat digunakan

oleh kedua entitas tersebut dengan memanfaatkan SOAP (*Simple Object Access Protocol*). Proses tersebut dapat dilihat pada gambar 4.49.



Gambar 4. 49 : Proses permintaan dan respon aplikasi *Web Service E-Commerce*
Sumber : [Perancangan]

Proses permintaan *web service* dimulai dari *Requester Entity* (entitas peminta) dengan cara melakukan permintaan terhadap dokumen WSDL yang berada pada *Provider Entity* (entitas penyedia). Kemudian dokumen WSDL melakukan permintaan *setting server* terhadap SOAP *server*. Selanjutnya SOAP Server juga melakukan permintaan *service* terhadap Modul *Web Service* yang berhubungan langsung dengan sistem dan basis data untuk mengidentifikasi fungsi/aplikasi yang akan digunakan dalam *web service*.

Setelah proses permintaan tersebut selesai, modul *Web Service* memberikan respon *service* kepada SOAP *server* untuk mengaktifkan modul aplikasi *web service*. Dilanjutkan dengan SOAP *server* memberikan respon kepada dokumen WSDL untuk mengaktifkan modul – modul aplikasinya. Dan kemudian SOAP *client* menerima respon *web service* yang berasal dari dokumen WSDL.

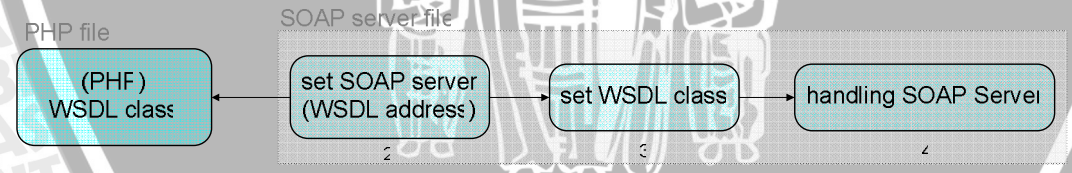
4.2.2.4.1. Perancangan SOAP *Server*

SOAP *server* merupakan komponen *web service* yang melakukan *request* dan *response* terhadap dokumen WSDL. Selain itu, SOAP *server* memerlukan referensi modul aplikasi yang terdapat pada file PHP yakni klas `wsdl` yang telah dijelaskan pada subbab *Class Diagram*.

Modul aplikasi *web service* tersebut terdiri dari *search*/pencarian produk, katalog produk, dan *review* produk. SOAP *server* ini melakukan proses *web service* pada *provider entity*. SOAP *server* ini berada pada bagian *provider entity* yakni pada Aplikasi *Web Service E-Commerce*.

Gambar 4.50 menunjukkan perancangan SOAP *server* pada Aplikasi *Web Service E-Commerce* saat terjadi *request* dan *response* terhadapnya. Pada perencangan tersebut, proses – proses yang terjadi adalah :

1. SOAP *server file* mengambil referensi modul dari PHP *file*, yakni fungsi – fungsi yang terdapat pada klas `wsdl`.
2. Pembentukan *object* SOAP *server* dan melakukan identifikasi untuk alamat dokumen WSDL.
3. Mengaktifkan klas `wsdl` dan fungsi – fungsi di dalamnya.
4. Mengaktifkan SOAP *server* beserta modul – modul *web service* yang terdapat pada dokumen WSDL berdasarkan fungsi – fungsi yang terdapat pada klas `wsdl`.



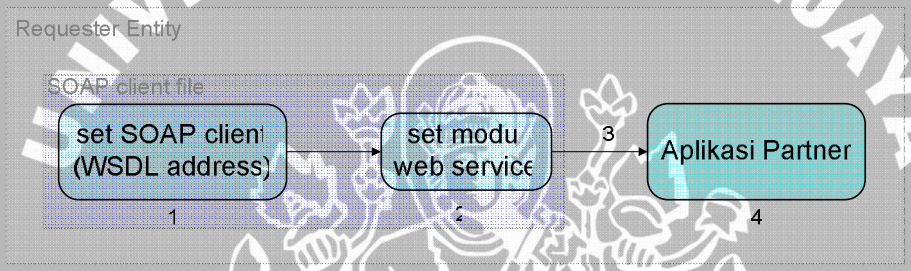
Gambar 4. 50 : Perancangan SOAP *server* pada Aplikasi *Web Service E-Commerce*
Sumber : [Perancangan]

4.2.2.4.2. Perancangan SOAP *Client*

SOAP *client* merupakan komponen *web service* yang melakukan *request* dan *response* terhadap dokumen WSDL yang melalui protokol HTTPS. SOAP *client* terdapat pada *requester entity* dimana dalam hal ini adalah *website partner*. SOAP *client* dapat menggunakan modul aplikasi *web service* yang terdapat pada dokumen WSDL dan telah diaktifkan oleh SOAP *server*.

Gambar 4.51 menunjukkan perancangan SOAP *client* untuk *website partner*. Pada perancangan SOAP *client* ini, modul aplikasi *web service* yang digunakan berdasarkan modul – modul pada dokumen WSDL dan yang telah diaktifkan oleh SOAP *server*. Proses – proses yang terjadi adalah :

1. Pembentukkan SOAP *client* dan identifikasi alamat WSDL yang terdapat pada *provider entity*.
2. Mengaktifkan modul *web service* yang terdapat pada dokumen WSDL.
3. Hasil keluaran SOAP *client file* diproses untuk digunakan pada aplikasi/*website partner*.
4. *Website Partner* telah menggunakan modul *web service* yang terdapat pada Aplikasi *Web Service E-Commerce*.

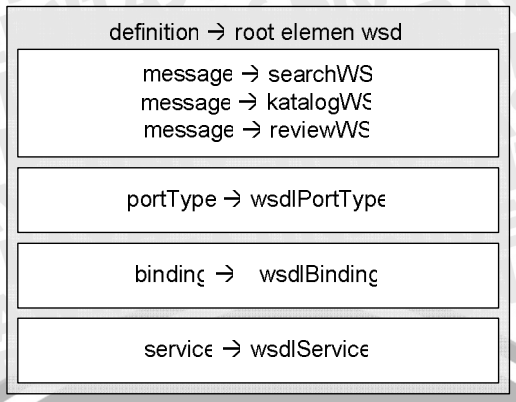


Gambar 4. 51 : Perancangan SOAP *client* untuk *Website Partner*
Sumber : [Perancangan]

4.2.2.4.3. Perancangan WSDL

WSDL (*Web Service Describe Language*) merupakan komponen *web service* menyimpan modul aplikasi *web service* dalam bentuk dokumen XML. Modul aplikasi tersebut yang digunakan oleh SOAP *server* atau SOAP *client*. Dokumen WSDL ini terdapat pada *provider entity* yakni pada Aplikasi *Web Service E-Commerce*.

Dokumen WSDL merupakan dokumen XML yang memiliki struktur utama yang terdiri dari *definition*, *message*, *porType*, *binding*, dan *service*. Struktur tersebut menentukan modul aplikasi yang akan digunakan untuk kebutuhan *web service*. Perancangan struktur utama WSDL pada Aplikasi *Web Service E-Commerce* ditunjukkan pada gambar 4.52.



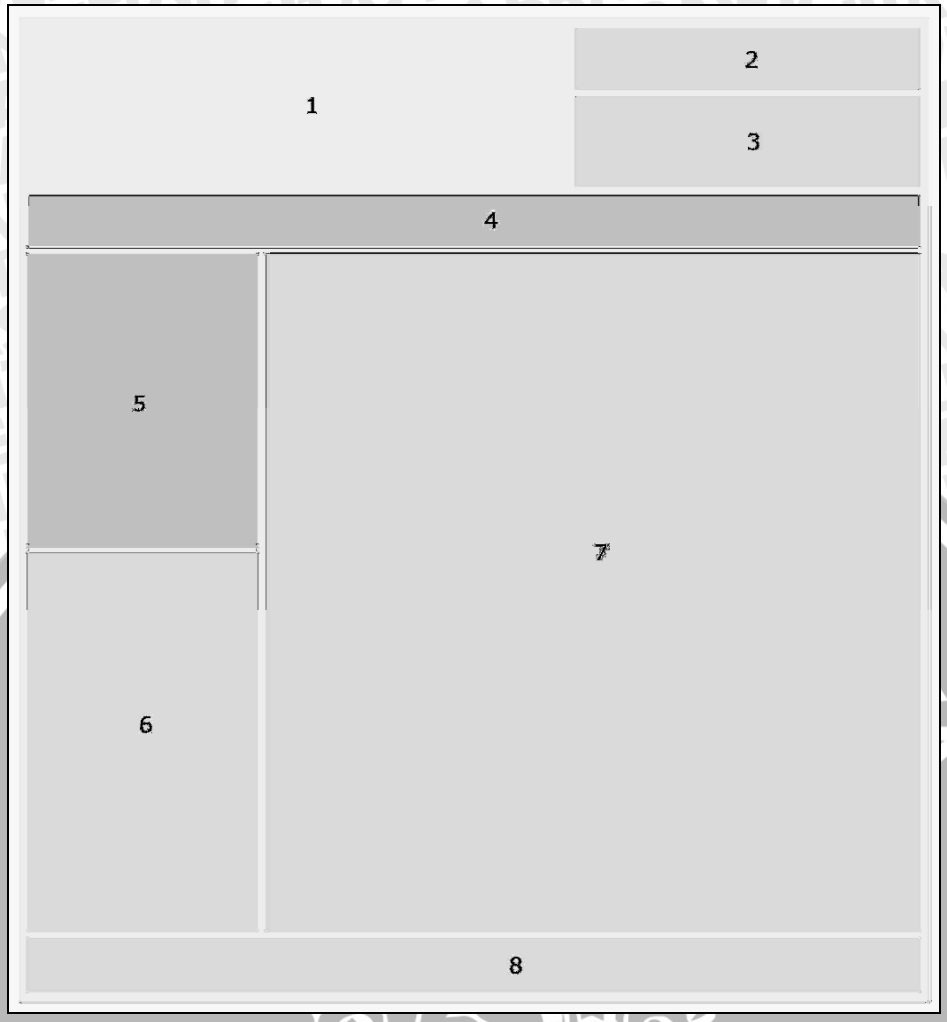
Gambar 4. 52 : Perancangan Struktur Utama WSDL pada Aplikasi *Web Service E-Commerce*
Sumber : [Perancangan]

4.2.2.5. Perancangan *User Interface*

User interface merupakan tampilan suatu aplikasi dimana *user* dapat berinteraksi dengan sistem. Aplikasi *Web Service E-Commerce* memiliki dua *user interface*, yaitu Halaman Utama *Website* dan Halaman Administrasi *Website*.

4.2.2.5.1. Perancangan Halaman Utama

Halaman utama Aplikasi *Web Service E-Commerce* merupakan halaman pertama yang diakses oleh *user* saat berinteraksi dengan sistem. Gambar 4.45 merupakan gambar rancangan *layout* halaman utama *website*.



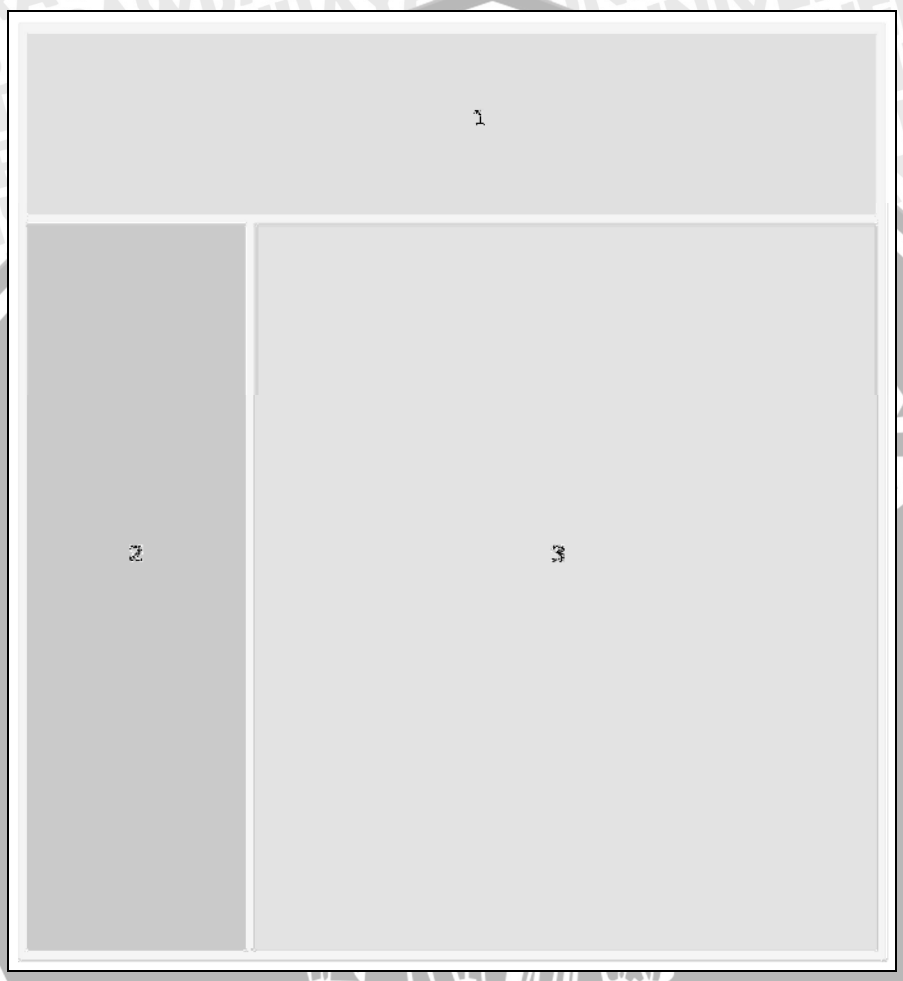
Gambar 4. 53 : *Layout* Halaman Utama
Sumber : [Perancangan]

Layout untuk halaman utama dari Aplikasi *Web Service E-Commerce* terdiri dari beberapa bagian sebagai berikut :

1. *Header*, berisi nama aplikasi.
2. *Login*, berisi *form login* yang berupa *field username* dan *field password*.
3. Status *Shopping Cart*, berisi jumlah barang/produk yang akan dipesan oleh *cosetumer*.
4. Menu Utama, berisi menu utama *website*.
5. Submenu Produk, berisi kategori produk.
6. Sekilas Berita, berisi sekilas berita terbaru.
7. Konten, berisi konten halaman utama *website* yang berupa *link web service* dan produk terbaru.
8. *Footer*, berisi *internal link* dan *copyright* aplikasi.

4.2.2.5.2. Perancangan Halaman Administrasi *Website*

Halaman Administrasi Aplikasi *Web Service E-Commerce* merupakan halaman yang hanya dapat diakses oleh *user administrator*. Halaman ini digunakan untuk proses administrasi *website*. Gambar 4.54 merupakan gambar rancangan layout halaman utama *website*.



Gambar 4. 54 : *Layout* Halaman Administrasi *Website*
Sumber : [Perancangan]

Layout untuk halaman administrasi dari Aplikasi *Web Service E-Commerce* terdiri dari beberapa bagian sebagai berikut :

1. *Header*, berisi nama halaman administrasi.
2. Menu Utama, berisi menu utama administrasi *website*.
3. Konten, berisi konten halaman administrasi *website*.

BAB V
IMPLEMENTASI

Bab ini membahas tentang implementasi Aplikasi *Web Service E-Commerce*. berdasarkan hasil yang telah didapatkan dari analisis kebutuhan dan proses perancangan aplikasi. Pembahasan bab ini merujuk pada subbab 4.2. Bab ini terdiri dari pembahasan implementasi algoritma dan implementasi antarmuka, ditambah dengan penjelasan tentang spesfikasi lingkungan dimana sistem diimplementasikan.

5.1. Spesifikasi Sistem

Dari hasil analisis kebutuhan dan perancangan yang telah diuraikan pada Bab 4, dilakukan implementasi menjadi sebuah Aplikasi *Web Service E-Commerce* yang nyata agar bisa berfungsi sesuai dengan kebutuhan. Aplikasi diimplementasikan pada spesfikasi *hardware, software* serta konfigurasi jaringan tertentu.

5.1.1. Spesifikasi Perangkat Keras (*Hardware*)

Aplikasi *Web Service* membutuhkan minimal 2 perangkat *server*. Perangkat *server* pertama digunakan untuk *provider entity* yakni Aplikasi *Web Service E-Commerce*. Dan perangkat *server* kedua digunakan untuk *requester entity* yakni Aplikasi *Partner*.

Perangkat *server* untuk Aplikasi *Web Service E-Commerce* menggunakan sebuah komputer dengan spesifikasi perangkat keras seperti terlihat pada Tabel 5.1.

Tabel 5. 1 : Spesifikasi perangkat keras untuk *server* Aplikasi *Web Service E-Commerce*

Nama Komponen	Spesifikasi
Prosesor	Intel® Pentium® IV 2.80 Hz
Memori (RAM)	1.024 MB
Hardisk	Maxtor ATA 80 GB
Motherboard	ASUSTek Computer Inc. P5PE-VM
VGA Card	ATI Radeon 9550 AGP 256 MB

Sumber : [Implementasi]

Perangkat *server* untuk Aplikasi *Partner* menggunakan sebuah komputer dengan spesifikasi perangkat keras seperti terlihat pada Tabel 5.2.

Tabel 5. 2 : Spesifikasi perangkat keras untuk *server* Aplikasi *Partner*

Nama Komponen	Spesifikasi
Prosesor	Intel® Pentium® IV 2.66 Hz
Memori (RAM)	512 MB
Hardisk	Maxtor ATA 80 GB
Motherboard	Gigabyte GA-945PL-S3P
VGA Card	-

Sumber : [Implementasi]

5.1.2. Spesifikasi Perangkat Lunak (*Software*)

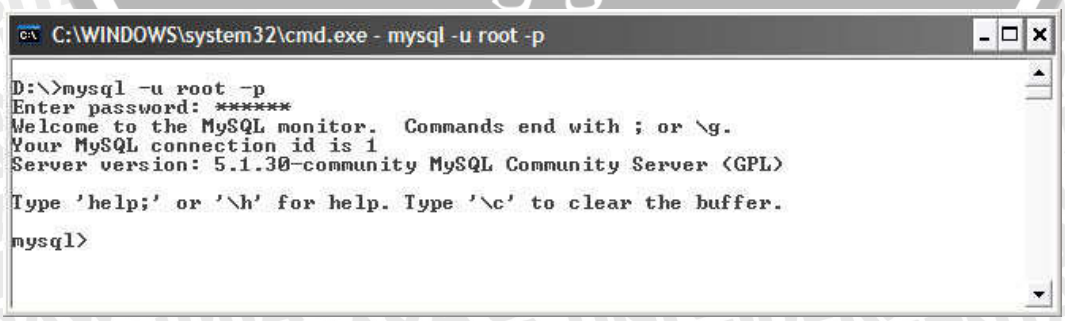
Untuk mengembangkan Aplikasi *Web Service E-Commerce*, digunakan *software* dengan spesifikasi seperti terlihat pada Tabel 5.3.

Tabel 5. 3 : Spesifikasi perangkat lunak komputer untuk Aplikasi *Web Service E-Commerce*

Perangkat Lunak	Spesifikasi
Sistem Operasi	Microsoft Windows XP SP2
Bahasa Pemrograman	PHP version 5.2.9
Basis Data	MySQL version 5.1.30
Web Server	Apache version 2.2.11 WIN32
Database Administrator Tool	SQLyog Enterprise version 5.21
Database Diagram Tool	Power Designer version 15.0
UML *Tool*	Power Designer version 15.0

Sumber : [Implementasi]

Untuk mengetahui versi dari basis data MySQL dapat digunakan command pada *command prompt* yaitu `mysql -u root`. Tampilan pada *command prompt* ditunjukkan pada gambar 5.1.



Gambar 5. 1 : Tampilan untuk *testing* Versi Basis Data MySQL yang Digunakan
Sumber : [Implementasi]

5.1.3. Spesifikasi Jaringan Komputer

Aplikasi *Web Service E-Commerce* ini diimplementasikan pada sebuah jaringan komputer lokal. SpesIF ikasi jaringan lokal *(Local Area Network)* tersebut dijelaskan melalui Tabel 5.4.

Tabel 5. 4 : Spesifikasi Jaringan Komputer

	Spesifikasi
Topologi Fisik	Bintang (Star)
Topologi Logik	10/100 Base T
LAN Card	3Com 3C905CX Etherlink XL 10/100 PCI
Kabel Jaringan	UTP cat 5e Belden USA
Switch	3Com 3C16470 16 port 10/100

Sumber : [Implementasi]

5.2. Batasan - Batasan Implementasi

Beberapa batasan dalam mengimplementasikan sistem adalah sebagai berkut :

1. *Web Browser* yang digunakan pada komputer *client* adalah Mozilla Firefox 3.0.10
2. Komputer *server* 1 (*provider entity*) dan komputer 2 (*requester entity*) dihubungkan oleh jaringan lokal.
3. Komputer *server* 1 (*provider entity*) dan komputer *client* dihubungkan oleh jaringan lokal.
4. Data - data yang digunakan merupakan data – data yang berasal dari situs – situs *E-Commerce* di *internet* (*http://www.nuansamusic.com* dan *http://www.deltamusik.com*).

5.3. Implementasi Basis Data

Implementasi perancangan basis data dilakukan sesuai dengan *entity relationship diagram*. Implementasi perancangan basis data Aplikasi *Web Service E-Commerce* menggunakan *Data Definition Languange* (DDL). DDL adalah struktur basis data yang menggambarkan desain basis data secara keseluruhan.

5.3.1. DDL untuk Membuat Basis Data `db_wse`

DDL yang digunakan untuk membentuk basis data `db_wse` adalah sebagai berikut :

```
create database db_wse;
```

5.3.2. DDL untuk Membuat Tabel `user`

DDL yang digunakan untuk membentuk tabel `user` adalah sebagai berikut:

```
CREATE TABLE `user` (
`idUser` int(5) NOT NULL AUTO_INCREMENT,
`idKategoriUser` int(5) NOT NULL DEFAULT '0',
`username` varchar(100) NOT NULL,
`password` varchar(255) NOT NULL,
`nama` varchar(50) NOT NULL DEFAULT '',
`email` varchar(100) NOT NULL,
`alamat` varchar(255) NOT NULL,
`propinsi` varchar(255) NOT NULL,
`kota` varchar(255) NOT NULL,
`kodePos` int(10) NOT NULL DEFAULT '0',
`telp` varchar(20) NOT NULL DEFAULT '',
`website` varchar(255) NOT NULL DEFAULT 'none',
`aktIF asi` varchar(255) NOT NULL DEFAULT 'none',
PRIMARY KEY (`idUser`),
FOREIGN KEY `idKategoriUser` REFERENCES kategori_user(`idKategoriUser`)
) ENGINE= MyISAM;
```

5.3.3. DDL untuk Membuat Tabel `kategori_user`

DDL yang digunakan untuk membentuk tabel `kategori_user` adalah sebagai berikut :

```
CREATE TABLE `kategori_user` (
`idKategoriUser` int(3) NOT NULL AUTO_INCREMENT,
`kategoriUser` varchar(50) NOT NULL,
PRIMARY KEY (`idKategoriUser)
) ENGINE= MyISAM;
```

5.3.4. DDL untuk Membuat Tabel barang

DDL yang digunakan untuk membentuk tabel barang adalah sebagai berikut :

```
CREATE TABLE `barang` (
`idBarang` int(5) NOT NULL AUTO_INCREMENT,
`idKategoriBarang` int(5) NOT NULL DEFAULT '0',
`idBeratKirim` int(5) NOT NULL DEFAULT '0',
`namaBarang` varchar(255) NOT NULL,
`deskripsiBarang` text NOT NULL,
`hargaBarang` int(100) NOT NULL DEFAULT '0',
`stokBarang` int(20) NOT NULL DEFAULT '0',
`gambarBarang` varchar(100) NOT NULL,
PRIMARY KEY (`idBarang`),
FOREIGN KEY `idKategoriBarang` REFERENCES
kategori_barang(`idKategoriBarang`)
) ENGINE= MyISAM;
```

5.3.5. DDL untuk Membuat Tabel kategori_barang

DDL yang digunakan untuk membentuk tabel kategori_barang adalah sebagai berikut :

```
CREATE TABLE `kategori_barang` (
`idKategoriBarang` int(5) NOT NULL AUTO_INCREMENT,
`kategoriBarang` varchar(255) NOT NULL,
PRIMARY KEY (`idKategoriBarang`)
) ENGINE= MyISAM;
```

5.3.6. DDL untuk Membuat Tabel berat_kirim

DDL yang digunakan untuk membentuk tabel berat_kirim adalah sebagai berikut :

```
CREATE TABLE `berat_kirim` (
`idBeratKirim` int(5) NOT NULL AUTO_INCREMENT,
`jenisBeratKirim` varchar(10) NOT NULL DEFAULT '0',
`hargaBeban` int(100) NOT NULL DEFAULT '0',
PRIMARY KEY (`idBeratKirim`)
) ENGINE= MyISAM;
```

5.3.7. DDL untuk Membuat Tabel `data_kirim`

DDL yang digunakan untuk membentuk tabel `data_kirim` adalah sebagai berikut :

```
CREATE TABLE `data_kirim` (
`idDataKirim` int(5) NOT NULL AUTO_INCREMENT,
`idJasaKirim` int(5) NOT NULL,
`kota` varchar(100) NOT NULL,
`hargaKirim` int(100) NOT NULL DEFAULT '0',
PRIMARY KEY (`idDataKirim`)
) ENGINE= MyISAM;
```

5.3.8. DDL untuk Membuat Tabel `jasa_kirim`

DDL yang digunakan untuk membentuk tabel `jasa_kirim` adalah sebagai berikut :

```
CREATE TABLE `jasa_kirim` (
`idJasaKirim` int(5) NOT NULL AUTO_INCREMENT,
`jasaKirim` varchar(50) NOT NULL DEFAULT '',
PRIMARY KEY (`idJasaKirim`)
) ENGINE= MyISAM;
```

5.3.9. DDL untuk Membuat Tabel `order_user`

DDL yang digunakan untuk membentuk tabel `order_user` adalah sebagai berikut :

```
CREATE TABLE `order_user` (
`idOrder` int(5) NOT NULL AUTO_INCREMENT,
`idUser` int(5) NOT NULL DEFAULT '0',
`idJasaKirim` int(5) NOT NULL DEFAULT '0',
`totalBiaya` int(100) NOT NULL DEFAULT '0',
`alamat` varchar(255) NOT NULL,
`kota` varchar(255) NOT NULL,
`kodePos` int(50) NOT NULL DEFAULT '0',
`sessionId` varchar(255) NOT NULL,
`tglPesan` date NOT NULL DEFAULT '0000-00-00',
`tglExpired` date NOT NULL DEFAULT '0000-00-00',
```

```
PRIMARY KEY (`idOrder`),
FOREIGN KEY `idUser` REFERENCES user(`idUser`),
FOREIGN KEY `idJasaKirim` REFERENCES jasa_kirim(`idJasaKirim`)
) ENGINE= MyISAM;
```

5.3.10. DDL untuk Membuat Tabel `shopping_cart`

DDL yang digunakan untuk membentuk tabel `shopping_cart` adalah sebagai berikut :

```
CREATE TABLE `shopping_cart` (
`idShoppingCart` int(5) NOT NULL AUTO_INCREMENT,
`idUser` int(5) NOT NULL DEFAULT '0',
`idBarang` int(5) NOT NULL DEFAULT '0',
`idStatusBayar` int(5) NOT NULL DEFAULT '0',
`jmlPesan` int(50) NOT NULL DEFAULT '0',
`idOrder` int(5) NOT NULL DEFAULT '0',
`sessionId` varchar(255) NOT NULL,
`tglShop` date NOT NULL DEFAULT '0000-00-00',
`tglRemove` date NOT NULL DEFAULT '0000-00-00',
PRIMARY KEY (`idShoppingCart`),
FOREIGN KEY `idUser` REFERENCES user(`idUser`),
FOREIGN KEY `idBarang` REFERENCES barang(`idBarang`),
FOREIGN KEY `idStatusBayar` REFERENCES status_bayar(`idStatusBayar`)
) ENGINE= MyISAM;
```

5.3.11. DDL untuk Membuat Tabel `status_bayar`

DDL yang digunakan untuk membentuk tabel `status_bayar` adalah sebagai berikut :

```
CREATE TABLE `status_bayar` (
`idStatusBayar` int(5) NOT NULL AUTO_INCREMENT,
`statusBayar` varchar(50) NOT NULL,
PRIMARY KEY (`idStatusBayar`)
) ENGINE= MyISAM;
```

5.3.12. DDL untuk Membuat Tabel `data_website`

DDL yang digunakan untuk membentuk tabel `data_website` adalah sebagai berikut :

```
CREATE TABLE `data_website` (
`idDataWebsite` int(5) NOT NULL AUTO_INCREMENT,
`idKategoriDataWebsite` int(5) NOT NULL DEFAULT '0',
`idUser` int(5) NOT NULL DEFAULT '0',
`judulDataWebsite` text NOT NULL,
`isiDataWebsite` text NOT NULL,
`tgl` date NOT NULL DEFAULT '0000-00-00',
`jam` time NOT NULL DEFAULT '00:00:00',
PRIMARY KEY (`idDataWebsite`),
FOREIGN KEY `idUser` REFERENCES user(`idUser`),
FOREIGN KEY `idKategoriDataWebsite` REFERENCES
kategori_data_website(`idKategoriDataWebsite`)
) ENGINE= MyISAM;
```

5.3.13. DDL untuk Membuat Tabel `kategori_data_website`

DDL yang digunakan untuk membentuk tabel `kategori_data_website` adalah sebagai berikut :

```
CREATE TABLE `kategori_data_website` (
`idKategoriDataWebsite` int(5) NOT NULL AUTO_INCREMENT,
`kategoriDataWebsite` varchar(20) NOT NULL DEFAULT '',
PRIMARY KEY (`idKategoriDataWebsite`)
) ENGINE= MyISAM;
```

5.3.14. DDL untuk Membuat Tabel `web_service`

DDL yang digunakan untuk membentuk tabel `web_service` adalah sebagai berikut :

```
CREATE TABLE `web_service` (
`idWebService` int(5) NOT NULL AUTO_INCREMENT,
`idKategoriWebService` int(5) NOT NULL DEFAULT '0',
`idUser` int(5) NOT NULL DEFAULT '0',
`tgl` date NOT NULL DEFAULT '0000-00-00',
```

```
PRIMARY KEY (`idWebService`),
FOREIGN KEY `idUser` REFERENCES user(`idUser`),
FOREIGN KEY `idKategoriWebService` REFERENCES
kategori_webservice(`idKategoriWebService`)
) ENGINE= MyISAM;
```

5.3.15. DDL untuk Membuat Tabel `kategori_webservice`

DDL yang digunakan untuk membentuk tabel `kategori_webservice` adalah sebagai berikut :

```
CREATE TABLE `kategori_webservice` (
`idKategoriWebService` int(5) NOT NULL AUTO_INCREMENT,
`kategoriWebService` varchar(100) NOT NULL,
`deskripsi` text NOT NULL,
PRIMARY KEY (`idKategoriWebService`)
) ENGINE= MyISAM;
```

5.3.16. DDL untuk Membuat Tabel `propinsi`

DDL yang digunakan untuk membentuk tabel `propinsi` adalah sebagai berikut :

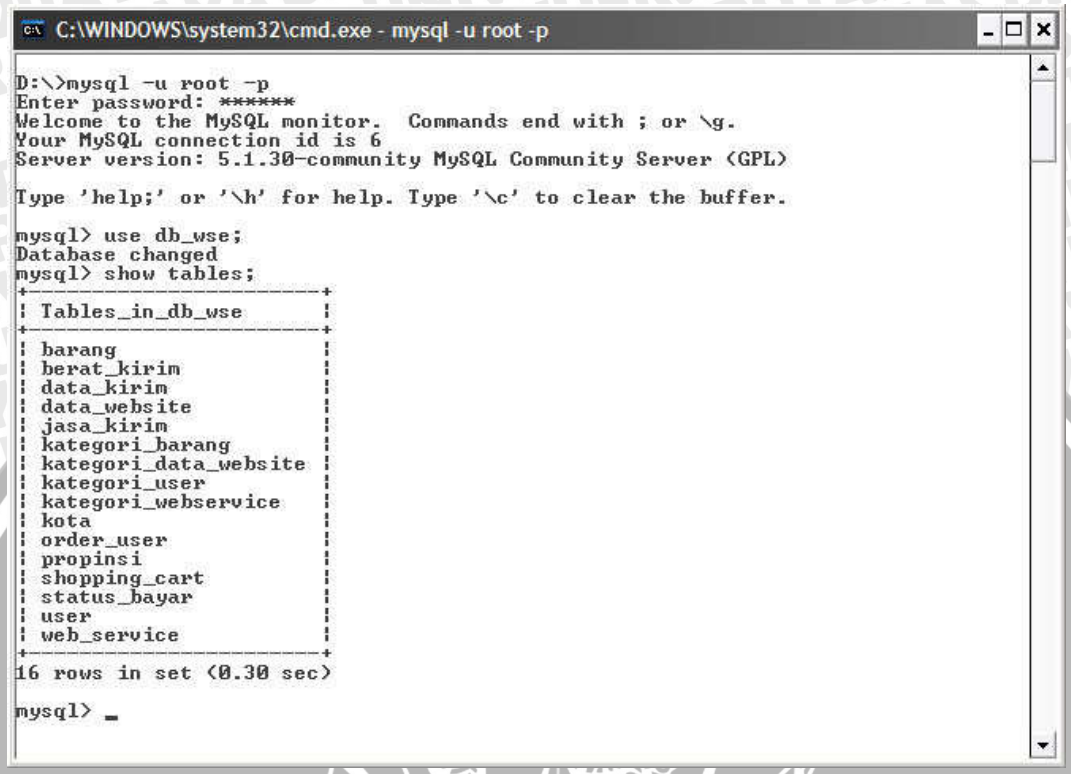
```
CREATE TABLE `propinsi` (
`idPropinsi` int(5) NOT NULL AUTO_INCREMENT,
`propinsi` varchar(50) NOT NULL DEFAULT '',
PRIMARY KEY (`idPropinsi`)
) ENGINE= MyISAM
```

5.3.17. DDL untuk Membuat Tabel `kota`

DDL yang digunakan untuk membentuk tabel `kota` adalah sebagai berikut:

```
CREATE TABLE `kota` (
`idKota` int(5) NOT NULL AUTO_INCREMENT,
`idPropinsi` int(5) NOT NULL DEFAULT '0',
`kota` varchar(255) NOT NULL,
PRIMARY KEY (`idKota`),
FOREIGN KEY `idPropinsi` REFERENCES propinsi(`idPropinsi`)
) ENGINE= MyISAM;
```

Basis data db_wse yang telah diimplementasikan pada basis data MySQL ditunjukkan dalam Gambar 5.2.



Gambar 5. 2 : Implementasi Basis Data pada MySQL
Sumber : [Implementasi]

5.4. Implementasi Antarmuka dan Algoritma

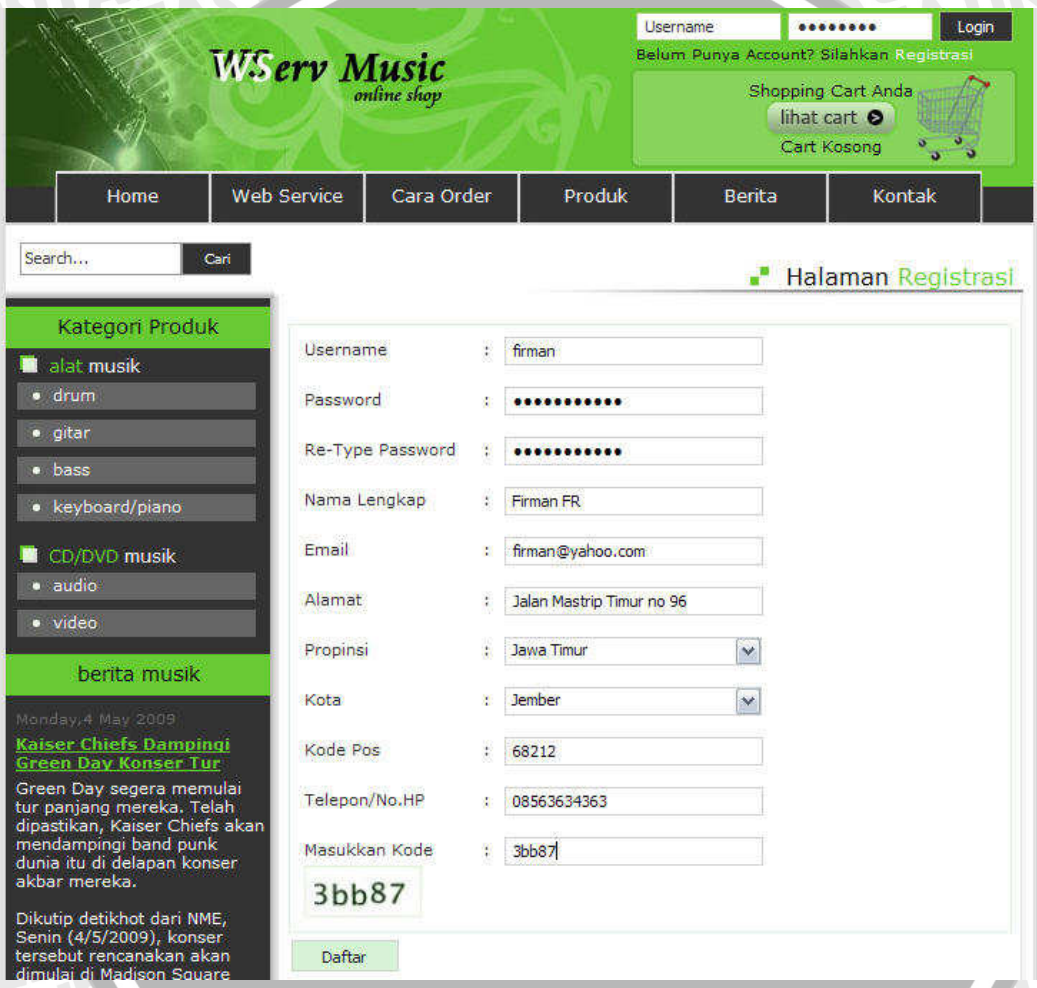
Subbab 5.4 membahas semua implementasi antarmuka yang didapat dari daftar kebutuhan fungsional. Subbab ini membahas seluruh implementasi antarmuka yang di dapat dari tabel 4.2.

5.4.1. Implementasi Antarmuka dan Algoritma untuk Aplikasi *Web Service E-Commerce*

Pada subbab 5.4.1 ini membahas tentang implementasi antarmuka dan algoritama untuk Aplikasi *Web Service E-Commerce*. Aplikasi ini merupakan *Provider Entity* dalam proses *Web Service*. Dimana *Provider Entity* merupakan bagian *web service* yang menyediakan modul – modul *web service*.

5.4.1.1. Implementasi Antarmuka Kebutuhan Fungsional Registrasi

Aktor yang mengakses Aplikasi *Web Service E-Commerce* disebut sebagai *User*. Sedangkan aktor yang memiliki *account* pada Aplikasi *Web Service E-Commerce* terdiri atas administrator dan customer. Seorang *user* yang ingin melakukan proses 'belanja' pada Aplikasi *Web Service E-Commerce* harus melakukan pendaftaran melalui proses registrasi. Gambar 5.3 menunjukkan implementasi antarmuka untuk registrasi.



Gambar 5. 3 : Implementasi Antarmuka untuk Registrasi
Sumber : [Implementasi]

Proses registrasi *user* untuk menaikkan level aktor, yakni dari *user* menjadi *customer*. Aktor *administrator* tidak memerlukan proses registrasi. Setiap *user* yang akan melakukan proses registrasi harus menekan tombol '`registrasi`'. Apabila proses registrasi telah berhasil, sistem akan menampilkan pesan yang ditunjukkan pada gambar 5.4.

Gambar 5. 4 : Hasil Implementasi Antarmuka untuk Registrasi
Sumber : [Implementasi]

Implementasi algoritma untuk registrasi ditunjukkan dalam tabel berikut :

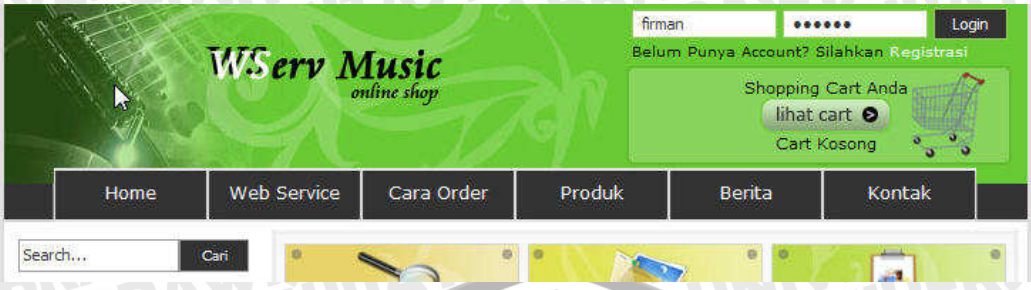
Algoritma 5. 1 : Algoritma Registrasi

```
FUNCTION cekRegistrasi

VARIABLE dbUser = new FUNCTION  dbUser
VARIABLE filter = new FUNCTION  inputFilter

IF FUNCTION isset VARIABLE _POST[‘daftar’] THEN
  VARIABLE formKosong = FUNCTION Array()
  foreach VARIABLE formKosong as VARIABLE keyFormKosong => VARIABLE valFormKosong
  BEGIN
  IF FUNCTION empty VARIABLE _POST[VARIABLE keyFormKosong] THEN
    VARIABLE error[VARIABLE keyFormKosong] = VARIABLE valFormKosong.” harus diisi”
  ELSE
    VARIABLE error[VARIABLE keyFormKosong] = “”
  ENDIF
  END

  VARIABLE formKarakter = FUNCTION array()
  FUNCTION foreach VARIABLE formKarakter as VARIABLE keyFormKarakter =>
  VARIABLE valFormKarakter
    BEGIN
    IF  FUNCTION not empty VARIABLE _POST[VARIABLE keyFormKarakter] AND
    VARIABLE filter SET FUNCTION karakterForm BASED ON VARIABLE
    _POST[VARIABLE keyFormKarakter] THEN
      VARIABLE error[VARIABLE keyFormKarakter] = “Karakter ”.VARIABLE
      valFormKarakter.” tidak diperbolehkan”
    ENDIF
    END

  IF VARIABLE dbUser SET FUNCTION numUser BASED ON VARIABLE _POST[‘userReg’]
    VARIABLE error[‘userReg’] = “Username telah terdaftar”
  ENDIF
  IF  VARIABLE dbUser SET FUNCTION numUser BASED ON VARIABLE _POST[‘namaReg’]
  THEN
    VARIABLE error[‘namaReg’] = “Nama telah terdaftar”
  ENDIF
  IF  FUNCTION not empty VARIABLE _POST[‘rePass’] AND VARIABLE
  _POST[‘rePass’] not = VARIABLE _POST[‘passReg’] THEN
    VARIABLE error[‘rePass’] = “Password harus sama”
  ENDIF
  IF  FUNCTION not empty VARIABLE _POST[‘namaReg’] AND VARIABLE filter SET
  FUNCTION karakterFormLainnya BASED ON VARIABLE _POST[‘namaReg’] THEN
    VARIABLE error[‘namaReg’] =  “Karakter nama tidak diperbolehkan”
  ENDIF
  IF  FUNCTION not empty VARIABLE _POST[‘emailReg’] AND VARIABLE filter SET
  FUNCTION karakterEmail BASED ON VARIABLE _POST[‘emailReg’] THEN
    VARIABLE error[‘emailReg’] = “Karakter email tidak diperbolehkan”
  ENDIF
  IF  FUNCTION not empty VARIABLE _POST[‘kodePosReg’] AND FUNCTION not
  is_numeric BASED ON VARIABLE _POST[‘kodePosReg’] THEN
    VARIABLE error[‘kodePosReg’] = “Kode pos anda salah”
```

```
  ENDIF
  IF  FUNCTION not empty VARIABLE _POST['telpReg'] AND VARIABLE filter SET
  FUNCTION karakterTelp BASED ON VARIABLE _POST['telpReg'] THEN
    VARIABLE error['telpReg'] = "Isi nomor telepon/hp anda dengan benar"
  ENDIF
  IF  FUNCTION not empty VARIABLE _POST['kode'] AND VARIABLE _POST['kode'] not =
  VARIABLE _SESSION['code'] THEN
    VARIABLE error['kode'] = "Kode keamanan tidak sesuai"
  ENDIF

  VARIABLE msgForm =  array()
  foreach VARIABLE error as VARIABLE keyError => VARIABLE valError
  BEGIN
  IF  VARIABLE valError = " " THEN
    IF  VARIABLE keyError = "rePass" OR VARIABLE keyError = "kode"
      VARIABLE keyError = ""
    ELSE
      VARIABLE post[VARIABEL keyError] = VARIABLE _POST[VARIABLE keyError]
      VARIABLE post['passReg'] = FUNCTION md5 BASED ON VARIABLE
      _POST['passReg']
    ENDIF
  ELSE
    VARIABLE post[VARIABLE keyError] = ""
  ENDIF
  END

  foreach VARIABLE post as VARIABLE kpost => VARIABLE vpost
  BEGIN
  IF  FUNCTION empty BASED ON VARIABLE vpost THEN
    VARIABLE msgPost = ""
  ELSE
    VARIABLE msgPost = "Terima kasih, proses registrasi telah berhasil."
  ENDIF
  END

  SET FUNCTION tambahUser BASED ON VARIABLE post
  VARIABLE data = SET FUNCTION regsitrasIF orm BASED ON VARIABLE msgForm,
  VARIABLE post,VARIABLE msgPost
ENDIF
RETURN VARIABLE data
```

5.4.1.2. Implementasi Antarmuka Kebutuhan Fungsional *Login*

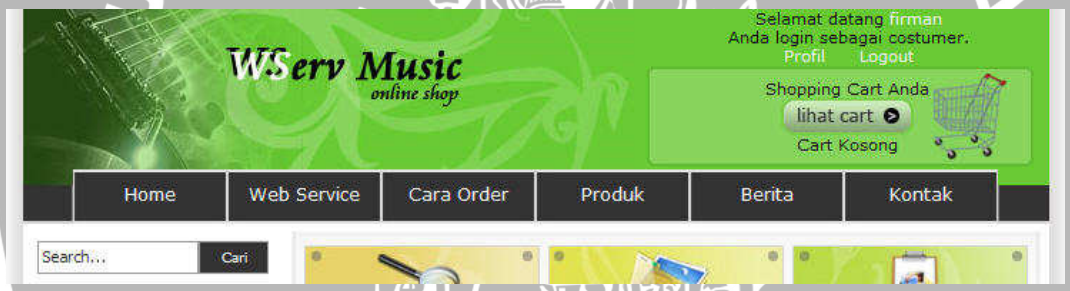
Aktor yang telah terdaftar pada Aplikasi *Web Service E-Commerce* dapat melakukan *login* ke dalam aplikasi. Proses *login* dapat digunakan oleh aktor *administrator*, aktor *customer*, dan aktor *partner*. Gambar 5.5 menunjukkan interaksi antarmuka untuk *login*.



Gambar 5. 5 : Implementasi Antarmuka untuk *Login*
Sumber : [Implementasi]

Aktor administrator harus melakukan proses login untuk melakukan proses administrasi data. Aktor customer harus melakukan proses login untuk melakukan pemesanan barang(pembelanjaan produk). Dan aktor partner melakukan proses login untuk proses registrasi web service.

Setelah aktor berhasil melakukan proses *login*, sistem menampilkan status *login*. Gambar 5.6 menunjukkan hasil implementasi antarmuka untuk *login*.



Gambar 5. 6 : Hasil Implementasi Antarmuka untuk *Login*
Sumber : [Implementasi]

Implementasi algoritma untuk *login* ditunjukkan dalam tabel berikut :

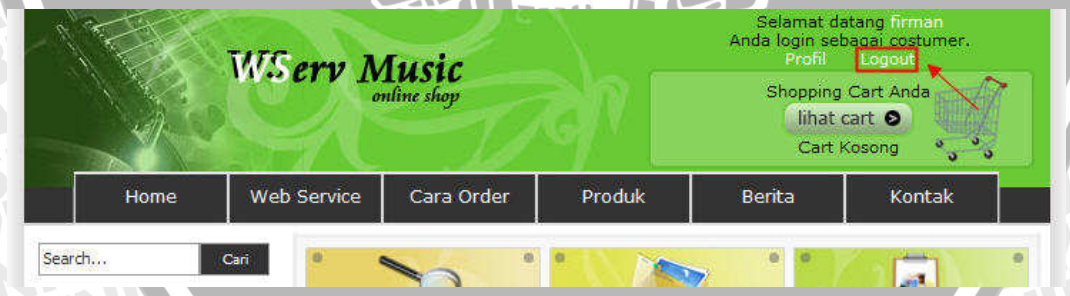
Algoritma 5. 2 : Algoritma *Login*

```
FUNCTION cekLogin

VARIABLE auth =  new FUNCTION  FUNCTION auth
VARIABLE dbUser =  new FUNCTION  FUNCTION dbUser
VARIABLE formLogin =  VARIABLE this SET FUNCTION loginForm

IF  FUNCTION isset BASED ON VARIABLE _POST[‘login’] THEN
  VARIABLE userLogin =  VARIABLE _POST[‘userLogin’]
  VARIABLE passLogin =  VARIABLE _POST[‘passLogin’]
  VARIABLE cryptPass =  FUNCTION md5 BASED ON VARIABLE passLogin

  IF  VARIABLE userLogin =  “Username” OR VARIABLE passLogin =  “password” THEN
    VARIABLE statusLogin =  “Username/Password harus diisi.”
  ENDIF

  IF  VARIABLE userLogin not =  “Username” AND VARIABLE passLogin not =
“password”
  AND VARIABLE dbUser SET FUNCTION numUser BASED ON VARIABLE userLogin,
```

```
  VARIABLE cryptPass THEN
    VARIABLE statusLogin =  "Username/Password anda salah."
  ENDIF

  IF  FUNCTION empty BASED ON VARIABLE statusLogin THEN
    VARIABLE lsUser =  VARIABLE dbUser SET FUNCTION detUser BASED ON VARIABLE
    userLogin,VARIABLE cryptPass
    VARIABLE _SESSION['idUser'] =  VARIABLE lsUser SET idUser
    VARIABLE _SESSION['idKategoriUser'] =  VARIABLE lsUser SET idKategoriUser
    VARIABLE _SESSION['kategoriUser'] =  VARIABLE lsUser SET kategoriUser
    VARIABLE _SESSION['username'] =  VARIABLE lsUser SET username
    VARIABLE _SESSION['login'] =  true
  ENDIF
ELSE
  VARIABLE statusLogin =  ''
ENDIF

IF  VARIABLE auth SET FUNCTION set =  true THEN
  VARIABLE statusLogin =  ''
  VARIABLE dataFormLogin =  "Selamat Datang ".VARIABLE _SESSION['username'].
  " Anda Login sebagai ".VARIABLE _SESSION['kategoriUser']
ELSE
  VARIABLE dataFormLogin =  VARIABLE formLogin
ENDIF

VARIABLE formData['getForm'] =  VARIABLE dataFormLogin
VARIABLE formData['statusUser'] =  VARIABLE statusLogin

RETURN VARIABLE formData
```

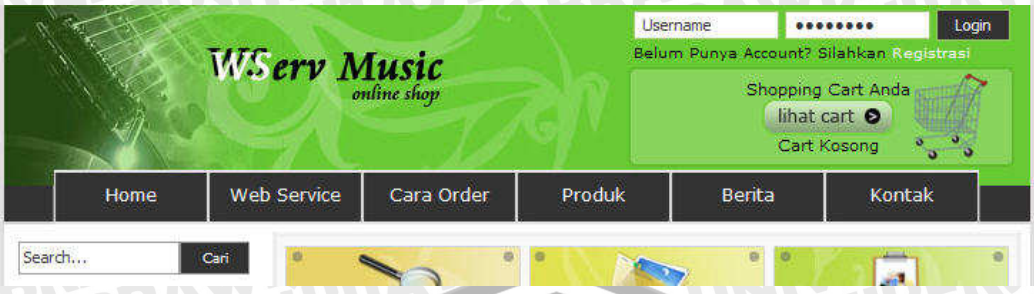
5.4.1.3. Implementasi Antarmuka Kebutuhan Fungsional *Logout*

Setiap aktor yang telah melakukan proses *login* Aplikasi *Web Service E-Commerce* dapat keluar dari aplikasi dengan menggunakan proses *logout*. Gambar 5.7 merupakan implementasi antarmuka untuk *logout*.



Gambar 5. 7 : Implementasi Antarmuka untuk *Logout*
Sumber : [Implementasi]

Apabila aktor telah melakukan proses *logout*, maka sistem akan menampilkan halaman utama dan tampilan status *login* kembali menjadi tampilan *form login*. Gambar hasil implementasi antramuka untuk *logout* ditunjukkan pada gambar 5.8.



Gambar 5. 8 : Hasil Implementasi Antarmuka untuk *Logout*
Sumber : [Implementasi]

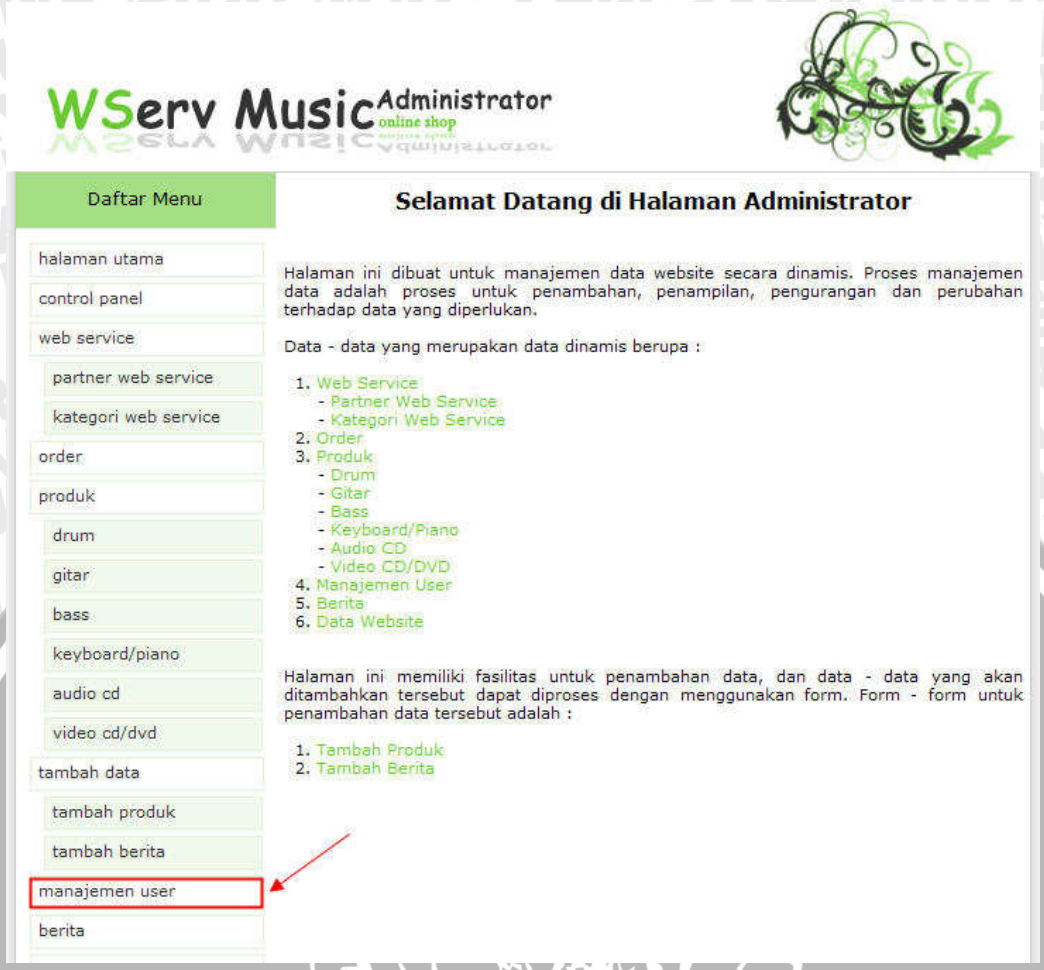
Implementasi algoritma untuk *logout* ditunjukkan dalam tabel berikut :

Algoritma 5. 3 : Algoritma *Logout*

```
FUNCTION logout

VARIABLE _SESSION =  FUNCTION Array()
FUNCTION session_unset
FUNCTION session_destroy
FUNCTION header BASED ON “Location:index.php”
```

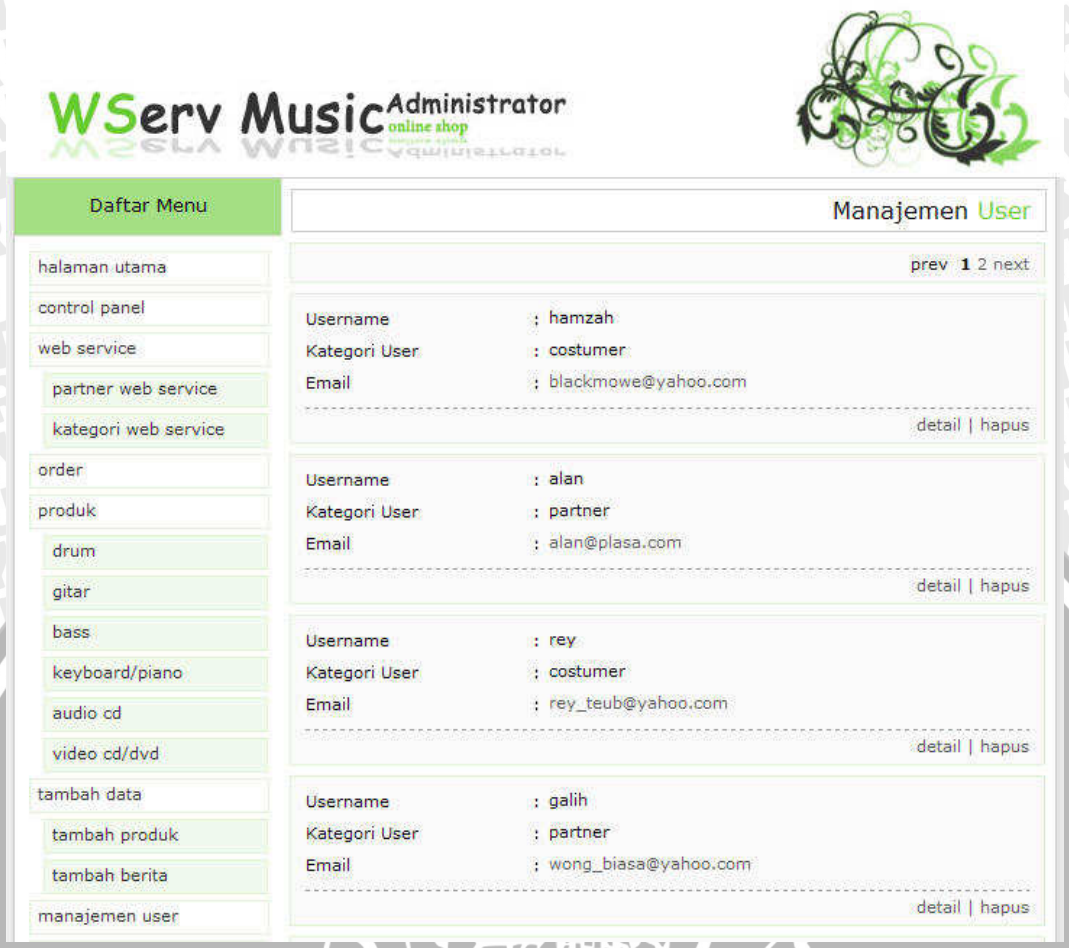
5.4.1.4. Implementasi Antarmuka Kebutuhan Fungsional Melihat Daftar *User*

Aplikasi *Web Service E-Commerce* memiliki fasilitas untuk melihat daftar *user*. Fasilitas ini hanya dapat diakses oleh *administrator* melalui halaman administrasi. Gambar 5.9 menunjukkan implementasi antarmuka untuk Melihat Daftar *User*.



Gambar 5. 9 : Implementasi Antarmuka untuk Melihat Daftar *User*
Sumber : [Implementasi]

Setelah *administrator* menekan tombol '`manajemen user`', sistem menampilkan halaman daftar *user* yang terdaftar pada Aplikasi *Web Service E-Commerce*. Gambar 5.10 menunjukkan hasil implementasi antarmuka untuk melihat daftar *user*.



Gambar 5. 10 : Hasil Implementasi Antarmuka untuk Melihat Daftar *User*
Sumber : [Implementasi]

Implementasi algoritma untuk Melihat Daftar *User* ditunjukkan dalam tabel berikut :

Algoritma 5. 4 : Algoritma Melihat Daftar *User*

```
FUNCTION listUser

VARIABLE auth =  new FUNCTION  auth
VARIABLE dbUser =  new FUNCTION  dbUser
VARIABLE paging =  new FUNCTION  paging

VARIABLE limit =  6
VARIABLE start =  VARIABLE paging SET FUNCTION getStart BASED ON
VARIABLE limit
VARIABLE listUser =  VARIABLE dbUser SET FUNCTION listUser BASED ON
VARIABLE start,VARIABLE limit
VARIABLE numUser =  VARIABLE dbUser SET FUNCTION numUser

foreach VARIABLE listUser as VARIABLE valListUser
  BEGIN
  VARIABLE ls[] =  VARIABLE valListUser SET username
  VARIABLE ls[].=  VARIABLE valListUser SET kategoriUser
  VARIABLE ls[].=  VARIABLE valListUser SET email
```

```
  END

VARIABLE query =  "index.php?admin= cpanel&amp;page= User&amp;"
IF  VARIABLE numUser > VARIABLE limit THEN
  VARIABLE setHtml =  VARIABLE paging SET FUNCTION setPaging BASED
  ON VARIABLE numUser,VARIABLE limit, VARIABLE query
ENDIF

foreach VARIABLE ls as VARIABLE dataUser
  BEGIN
  VARIABLE setHtml.=  VARIABLE dataUser
  END

IF  VARIABLE auth SET FUNCTION akses =  true THEN
  RETURN VARIABLE setHtml
ELSE
  VARIABLE setHtml =  ''
  RETURN VARIABLE setHtml
ENDIF
```

5.4.1.5. Implementasi Antarmuka Kebutuhan Fungsional Melihat Data Profil *User*

Aplikasi *Web Service E-Commerce* menyediakan fasilitas untuk melihat data profil *user*. Fasilitas ini dapat diakses oleh *administrator* dan *customer*. Aktor *administrator* dapat mengakses fasilitas ini di halaman administrasi *website*. Gambar 5.10 menunjukkan implementasi antarmuka untuk melihat data profil *user* yang dilakukan oleh *administrator*.

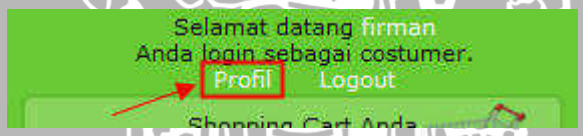
Gambar 5. 11 : Implementasi Antarmuka untuk Melihat Data Profil *User* oleh *Administrator*
Sumber : [Implementasi]

Setelah *administrator* menekan tombol '`detail`', sistem menampilkan halaman data profil *user* yang dilihat oleh *administrator*. Gambar 5.12 menunjukkan hasil implementasi antarmuka untuk melihat data profil *user* yang

dilakukan oleh *administrator*.

Gambar 5. 12 : Hasil Implementasi Antarmuka untuk Melihat Data Profil *User* oleh *Administrator*
Sumber : [Implementasi]

Sedangkan aktor *customer* dapat mengakses fasilitas ini dengan menekan tombol '`profil`' pada status *login*. Gambar 5.13 menunjukkan implementasi antarmuka untuk melihat data profil *user* yang dilakukan oleh *customer*.



Gambar 5. 13 : Implementasi Antarmuka untuk Melihat Data Profil *User* oleh *Customer*
Sumber : [Implementasi]

Setelah menekan tombol '`profil`', sistem menampilkan data profil *customer* sendiri. Gambar 5.14 menunjukkan hasil implementasi antarmuka untuk melihat data profil *user* yang dilakukan oleh *customer*.

Gambar 5. 14 : Hasil Implementasi Antarmuka untuk Melihat Data Profil *User* oleh *Customer*
Sumber : [Implementasi]

Implementasi algoritma untuk Melihat Data Profil *User* ditunjukkan dalam tabel berikut :

Algoritma 5. 5 : Algoritma Melihat Data Profil *User*

```
FUNCTION detUser

VARIABLE auth =  new FUNCTION auth
VARIABLE htmlLayout =  new FUNCTION htmlLoyout
VARIABLE dbUser =  new FUNCTION dbUser

IF  VARIABLE auth SET FUNCTION set =  true THEN
  VARIABLE idUserProfil =  VARIABLE _SESSION['idUser']
ELSE
  VARIABLE idUserProfil =  ''
ENDIF

IF  VARIABLE auth SET FUNCTION admin THEN
  VARIABLE idUserProfil =  VARIABLE _GET['id'];
  VARIABLE detUser =  VARAIBLE dbUser SET detUser BASED ON VARIABLE _GET['id']
  VARIABLE setHtml =  VARIABLE detUser SET username
  VARIABLE setHtml.=  VARIABLE detUser SET kategoriUser
  VARIABLE setHtml.=  VARIABLE detUser SET nama
  VARIABLE setHtml.=  VARIABLE detUser SET email
  VARIABLE setHtml.=  VARIABLE detUser SET alamat
  VARIABLE setHtml.=  VARIABLE detUser SET propinsi
  VARIABLE setHtml.=  VARIABLE detUser SET kota
  VARIABLE setHtml.=  VARIABLE detUser SET kodePos

  IF  VARIABLE detUser SET website =  "none" THEN
    VARIBLE setHtml.=  ''
  ELSE
```

```
    VARIABLE setHtml.=  VARIABLE detUser SET website
    VARIABLE setHtml.=  VARIABLE detUser SET aktifasi
  ENDIF

  VARIABLE data[‘isi’] =  VARIABLE setHtml
ELSE
  VARIABLE detUser =  VARIABLE dbUser SET FUNCTION detUser BASED ON VARIABLE
  idUserProfil
  VARIABLE getHtmlIndex =  VARIABLE htmlIndex SET FUNCTION htmlNonIndex BASED ON
  'Profil',VARIABLE detUser SET nama
  VARIABLE setHtml =  VARIABLE getHtmlIndex[0]
  VARIABLE setHtml =  VARIABLE detUser SET username
  VARIABLE setHtml.=  VARIABLE detUser SET kategoriUser
  VARIABLE setHtml.=  VARIABLE detUser SET nama
  VARIABLE setHtml.=  VARIABLE detUser SET email
  VARIABLE setHtml.=  VARIABLE detUser SET alamat
  VARIABLE setHtml.=  VARIABLE detUser SET propinsi
  VARIABLE setHtml.=  VARIABLE detUser SET kota
  VARIABLE setHtml.=  VARIABLE detUser SET kodePos

  IF  VARIABLE detUser SET website =  “none” THEN
    VARIBLE setHtml.=  ‘’
  ELSE
    VARIABLE setHtml.=  VARIABLE detUser SET website
    VARIABLE setHtml.=  VARIABLE detUser SET aktIF asi
  ENDIF
  VARAIBLE setHtml.=  VARAIBLE getHtmlIndex[1]
  VARAIBLE data =  VARAIBLE setHtml
ENDIF

IF  VARIABLE auth FUNCTION set = =  true THEN
  RETURN VARIABLE data
ELSE
  VARIABLE data =  "";
  RETURN VARIABLE data
ENDIF
```

5.4.1.6. Implementasi Antarmuka Kebutuhan Fungsional Mengedit Data Profil *User*

Aplikasi *Web Service E-Commerce* menyediakan fasilitas untuk mengedit data profil *user*. Fasilitas ini dapat diakses oleh *administrator* dan *customer*. Aktor *administrator* dapat mengakses fasilitas ini di halaman administrasi *website*.

Sedangkan aktor *customer* dapat mengakses fasilitas ini di halaman profil *user* itu sendiri. Gambar 5.15 menunjukkan implementasi antarmuka untuk mengedit data profil *user*.

Gambar 5. 15 : Implementasi Antarmuka untuk Mengedit Data Profil *User*
Sumber : [Implementasi]

Setelah menekan tombol '`Edit Profil`', sistem menampilkan *form* edit profil *user*. Gambar 5.16 menunjukkan hasil implementasi antarmuka untuk mengedit data profil *user*.

Gambar 5. 16 : Hasil Implementasi Antarmuka untuk Mengedit Data Profil *User*
Sumber : [Implementasi]

Implementasi algoritma untuk Mengedit Data Profil *User* ditunjukkan dalam tabel berikut :

Algoritma 5. 6 : Algoritma Mengedit Data Profil *User*

```
FUNCTION cekEditUser

VARIABLE auth =  new FUNCTION  FUNCTION auth
VARIABLE dbUser =  new FUNCTION  FUNCTION dbUser
```

```
VARIABLE filter =  new FUNCTION  FUNCTION inputFilter

IF  VARIABLE auth SET FUNCTION set =  true THEN
  VARIABLE idUserProfil =  VARIABLE _SESSION[‘idUser’]
ELSE
  VARIABLE idUserProfil =  ‘’
END

VARIABLE detUser =  VARIABLE dbUser FUNCTION detUser BASED ON VARIABLE
idUserProfil
IF  FUNCTION isset BASED ON _POST[‘editUser’] THEN
  VARIABLE varEdit =  array()
  FUNCTION foreach VARIABLE varEdit as VARIABLE keyVarEdit = > VARIABLE
  valVarEdit
  BEGIN
    IF  FUNCTION empty BASED ON _POST[VARIABLE keyVarEdit] THEN
      VARIABLE error[VARIABLE keyVarEdit] =  VARIABLE valVarEdit.” harus
      diisi”
    ELSE
      VARIABLE error[VARIABLE keyVarEdit] =  “ ”
    ENDIF
  END
  IF  FUNCTION not empty BASED ON _POST[‘userEdit’] AND VARIABLE filter
  SET FUNCTION karakterForm BASED ON _POST[‘userEdit’] THEN
    VARIABLE error[‘userEdit’] =  “Karakter username tidak diperbolehkan.”
  ENDIF
  IF  FUNCTION not empty BASED ON _POST[‘userEdit’] AND VARIABLE dbUser SET
  FUNCTION numUser BASED ON VARIABLE idUserProfil,VARIABLE _POST[‘userEdit’]
  =  1 THEN
    VARIABLE error[‘userEdit’] =  “Username telah terdaftar.”
  ENDIF
  IF  FUNCTION not empty BASED ON _POST[‘namaEdit’] AND VARIABLE dbUser SET
  FUNCTION numUser BASED ON VARIABLE idUserProfil,VARIABLE _POST[‘namaEdit’]
  =  1 THEN
    VARIABLE error[‘namaEdit’] =  “nama telah terdaftar.”
  ENDIF
  IF  FUNCTION not empty BASED ON _POST[‘namaEdit’] AND VARIABLE filter
  SET FUNCTION karakterFormLainnya BASED ON _POST[‘namaEdit’] THEN
    VARIABLE error[‘namaEdit’] =  “Karakter nama tidak diperbolehkan.”
  ENDIF
  IF  FUNCTION not empty BASED ON _POST[‘emailEdit’] AND VARIABLE filter
  SET FUNCTION karakterEmail BASED ON _POST[‘emailEdit’] THEN
    VARIABLE error[‘emailEdit’] =  “Karakter email tidak diperbolehkan.”
  ENDIF
  IF  FUNCTION not empty BASED ON _POST[‘kodePosEdit’] AND FUNCTION not
  is_numuric BASED ON _POST[‘kodePosEdit’] THEN
    VARIABLE error[‘kodePosEdit’] =  “Kode pos anda salah.”
  ENDIF
  IF  FUNCTION not empty BASED ON _POST[‘telpEdit’] AND VARIABLE filter
  SET FUNCTION karakterEmail BASED ON _POST[‘telpEdit’] THEN
    VARIABLE error[‘telpEdit’] =  “Isi nomor telepon/hp anda dengan benar.”
  ENDIF
  FUNCTION foreach VARIABLE error as VARIABLE keyError = > VARIABLE valError
  BEGIN
    IF  VARIABLE error[] =  “ ” THEN
      VARIABLE post['userEdit'] =  VARIABLE detUser SET username
      VARIABLE post['namaEdit'] =  VARIABLE detUser SET nama
      VARIABLE post['emailEdit'] =  VARIABLE detUser SET email;
      VARIABLE post['alamatEdit'] =  VARIABLE detUser SET alamat;
      VARIABLE post['propEdit'] =  VARIABLE detUser SET propinsi;
```

```
      VARIABLE post['kotaEdit'] =  VARIABLE detUser SET kota;
      VARIABLE post['kodePosEdit'] =  VARIABLE detUser SET kodePos;
      VARIABLE post['telpEdit'] =  VARIABLE detUser SET telp;
      VARIABLE msgPost =  '';
    ELSE
      VARIABLE post[VARIABLE keyErr] =  VARIABLE _POST[VARIABLE keyErr]
      VARAIBLE msgPost =  "Data anda telah berhasil diproses."
    ENDIF
  END
  VARIABLE edit =  VARIABLE this SET editUser BASED ON VARIABLE
  idUserProfil,VARIABLE post,'data'
  VARIABLE data =  VARIABLE this SET editFormUser VARIABLE error,VARIABLE
  post,VARIABLE msgPost
ENDIF

RETURN VARAIBLE data
```

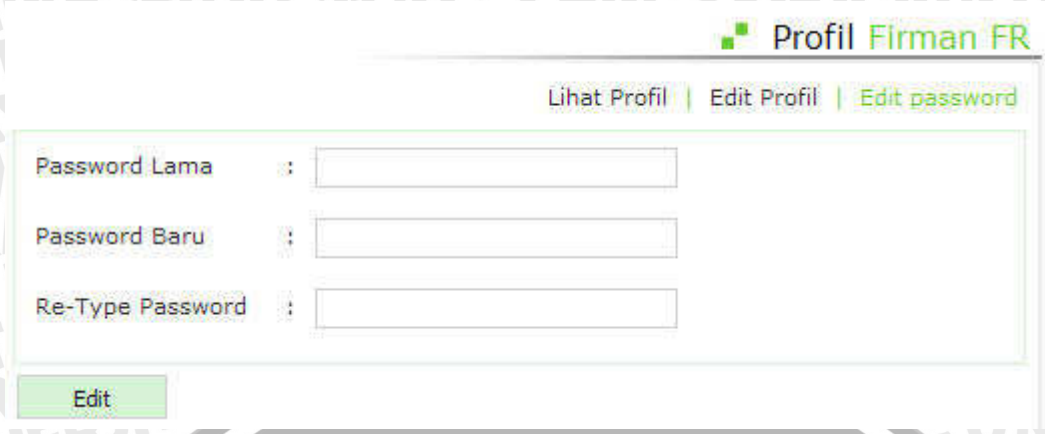
5.4.1.7. Implementasi Antarmuka Kebutuhan Fungsional Mengedit *Password User*

Aplikasi *Web Service E-Commerce* menyediakan fasilitas untuk mengedit *password user*. Fasilitas ini dapat diakses oleh *administrator* dan *customer*. Aktor *administrator* dapat mengakses fasilitas ini di halaman administrasi *website*.

Sedangkan aktor *customer* dapat mengakses fasilitas ini dengan menekan tombol '`edit Password`' pada halaman profil *user*. Gambar 5.17 menunjukkan implementasi antarmuka untuk mengedit *password user*.

Gambar 5. 17 : Implementasi Antarmuka untuk Mengedit *Password User*
Sumber : [Implementasi]

Setelah menekan tombol '`Edit Password`', sistem menampilkan *form* edit *password user*. Gambar 5.18 menunjukkan hasil implementasi antarmuka untuk mengedit *password user*.



Gambar 5. 18 : Hasil Implementasi Antarmuka untuk Mengedit *Password User*
Sumber : [Implementasi]

Implementasi algoritma untuk Mengedit *Password User* ditunjukkan dalam tabel berikut :

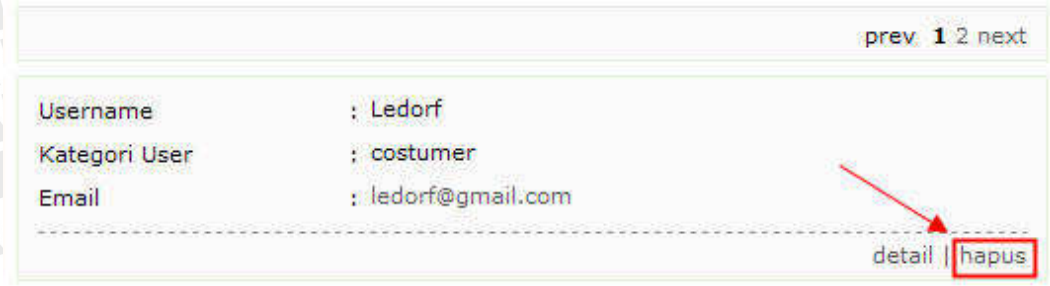
Algoritma 5. 7 : Algoritma Mengedit *Password User*

```
FUNCTION cekEditPass

VARIABLE auth =  new FUNCTION  FUNCTION auth
VARIABLE dbUser =  new FUNCTION  FUNCTION dbUser
VARIABLE filter =  new FUNCTION  FUNCTION inputFilter

IF  VARIABLE auth SET FUNCTION set =  true THEN
  VARIABLE idUserProfil =  VARIABLE _SESSION['idUser']
  VARIABLE passLama =  FUNCTION md5 BASED ON VARIABLE _POST['passLama']
ELSE
  VARIABLE idUserProfil =  ''
  VARIABLE passLama =  ''
ENDIF

IF  FUNCTION isset BASED ON VARIABLE _POST['editPassUser'] THEN
  VARIABLE varPassEdit =  array()
  FUNCTION foreach VARIABLE varPassEdit as VARIABLE keyVarPassEdit = >
  VARIABLE valVarPassEdit
  BEGIN
  IF  FUNCTION not empty BASED ON VARIABLE _POST[VARAIBLE keyVarPassEdit]
  THEN
    VARIABLE err[VARIABLE keyVarPassEdit] =  VARIABLE valVarPassEdit." harus
    diisi"
  ELSE
    VARIABLE err[VARAIBLE keyVarPassEdit] =  " "
  ENDIF

  VARIABLE varPassEditKarakter =  array()
  FUNCTION foreach VARIABLE varPassEditKarakter as VARIABLE
  keyVarPassEditKarakter = > VARIABLE valVarPassEditKarakter
  BEGIN
    IF  FUNCTION not empty BASED ON VARIABLE _POST[VARIABLE
    keyVarPassEditKarakter] AND VARIABLE filter SET
    FUNCTION karakterForm BASED ON VARIABLE _POST[VARIABLE
    keyVarPassEditKarakter] THEN
      VARIABLE err[VARIABLE keyVarPassEditKarakter] =  "Karakter ".VARIABLE
```

```
      valVarPassEditKarakter.” tidak diperbolehkan.”
   ENDIF
  END

  IF  VARIABLE dbUser SET FUNCTION numUser BASED ON VARIABLE idUserProfil,
  VARIABLE passLama =  0 THEN
    VARIABLE err['passLama'] =  'Password lama anda salah.'
  ENDIF
  IF  FUNCTION not empty BASED ON VARIABLE _POST['passFixEdit'] AND VARIABLE
  _POST['passFixEdit'] not =  VARIABLE _POST['passBaru'] THEN
    VARIABLE err['passFixEdit']= 'Password harus sama.'
  ENDIF

  FUNCTION foreach VARIABLE err as VARIABLE keyErr = > VARIBLE valErr
  BEGIN
  IF  VARIABLE valErr= ' ' THEN
    IF  VARIABLE keyErr =  'passLama' OR VARIABLE keyErr =  'passFixEdit' THEN
      VARIABLE keyErr =  ''
    ELSE
      VARIABLE post[VARIABLE keyErr] =  FUNCTION md5 BASED ON VARIABLE
      _POST[VARIABLE keyErr]
    ENDIF
  ELSE
    VARIABLE post[VARIABLE keyErr] =  ''
  ENDIF
  END

  FUNCTION foreach VARIABLE post as VARIABLE kpost = > VARIABLE vpost
  BEGIN
  IF  FUNCTION empty BASED ON VARIABLE vpost THEN
    VARIABLE msgPost =  ""
  ELSE
    VARIABLE msgPost =  "Data anda telah berhasil diproses."
  ENDIF
  END

  VARIABLE edit =  VARIABLE this SET FUNCTION editUser BASED ON VARIABLE
  idUserProfil,VARIABLE post,'pass'
  VARIABLE data =  VARIABLE this SET FUNCTION editFormPass BASED ON VARIABLE
  err,VARIABLE msgPost
ENDIF

RETURN VARIABLE data
```

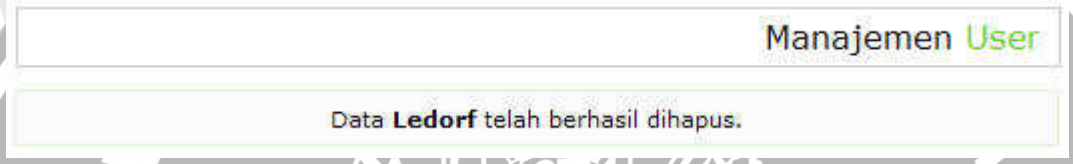
5.4.1.8. Implementasi Antarmuka Kebutuhan Fungsional Menghapus User

Aplikasi *Web Service E-Commerce* menyediakan fasilitas untuk menghapus *user*. Fasilitas ini dapat hanya diakses oleh *administrator*. Aktor *administrator* mengakses fasilitas ini melalui halaman administrasi *website*. Gambar 5.19 menunjukkan implementasi antarmuka untuk menghapus *user*.



Gambar 5. 19 : Implementasi Antarmuka untuk Menghapus *User*
Sumber : [Implementasi]

Setelah menekan tombol 'hapus', sistem menampilkan pesan bahwa *user* tersebut telah dihapus dari sistem. Gambar 5.20 menunjukkan hasil implementasi antarmuka untuk menghapus *user*.



Gambar 5. 20 : Hasil Implementasi Antarmuka untuk Menghapus *User*
Sumber : [Implementasi]

Implementasi algoritma untuk Menghapus *User* ditunjukkan dalam tabel berikut :

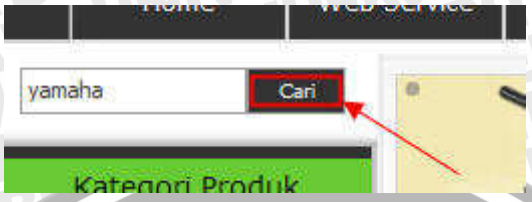
Algoritma 5. 8 : Algoritma Menghapus *User*

```
FUNCTION hapusUser

VARIABLE auth =  new FUNCTION  FUNCTION auth
VARIABLE dbUser =  new FUNCTION  FUNCTION dbUser
VARIABLE id =  VARIABLE _GET['id']

VARIABLE detUser =  VARIABLE dbUser SET FUNCTION detUser BASED ON VARIABLE id
VARIABLE setHtml =  VARIABLE detUser SET username
VARIABLE hapusUser =  VARIABLE dbUser SET FUNCTION hapusUser BASED ON
VARIABLE id

IF  VARIABLE auth SET FUNCTION set =  true THEN
  VARIABLE data['isi'] =  VARIABLE setHtml
ELSE
  VARIABLE data['isi'] =  ""
ENDIF

RETURN VARIABLE data
```

5.4.1.9. Implementasi Antarmuka Kebutuhan Fungsional Mencari Data Barang

Aplikasi *Web Service E-Commerce* memiliki fasilitas pencarian data

barang. Fasilitas ini dapat digunakan oleh semua aktor. Proses pencarian barang pada aplikasi ini berdasarkan nama barang dan deskripsi barang. Gambar 5.21 menunjukkan implementasi antarmuka untuk mencari data barang.



Gambar 5. 21 : Implementasi Antarmuka untuk Mencari Data Barang
Sumber : [Implementasi]

Setelah menekan tombol '`cari`', sistem menampilkan data barang yang dicari oleh *user*. Gambar 5.20 menunjukkan hasil implementasi antarmuka untuk mencari data barang.

Gambar 5. 22 : Hasil Implementasi Antarmuka untuk Mencari Data Barang
Sumber : [Implementasi]

Implementasi algoritma untuk Mencari Data Barang ditunjukkan dalam tabel berikut :

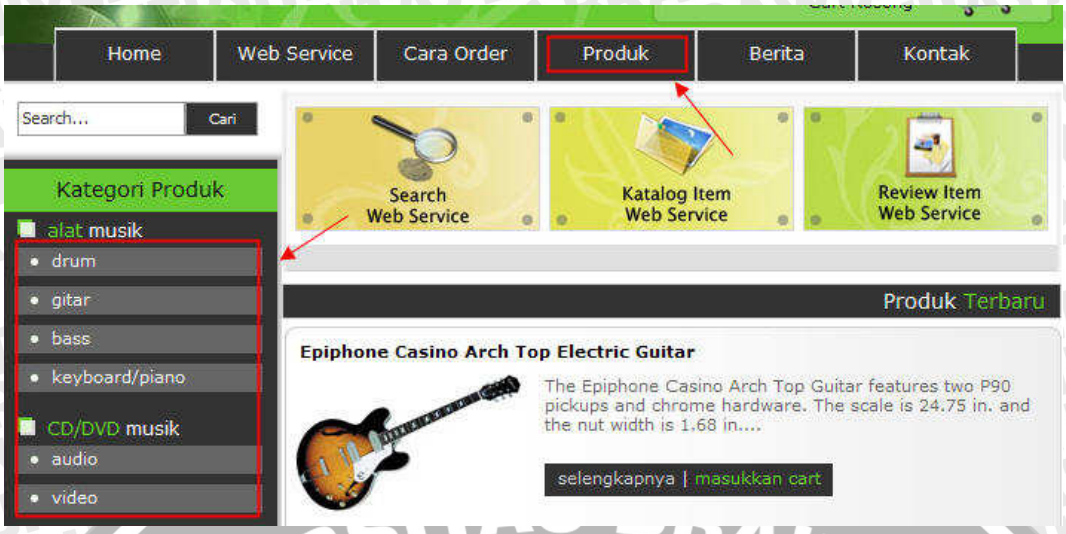
Algoritma 5. 9 : Algoritma Mencari Data Barang

```
FUNCTION cekFormCariBarang

VARIABLE db =  new FUNCTION  FUNCTION Db
VARIABLE cutTxt =  new FUNCTION  FUNCTION textCutting
VARIABLE paging =  new FUNCTION  FUNCTION paging
VARIABLE htmlIndex =  new FUNCTION  FUNCTION htmlLayout
VARIABLE getHtmlNonIndex =  VARIABLE htmlIndex SET FUNCTION htmlNonIndex BASED ON
'Pencarian','Produk Musik'
VARIABLE formCari =  VARIABLE this SET FUNCTION formCariBarang
IF  FUNCTION isset BASED ON VARIABLE _GET['q']
  VARIABLE getQ =  FUNCTION base64_decode BASED ON VARIABLE _GET['q']
  VARIABLE filterData =  FUNCTION trim BASED ON VARIABLE getQ
  VARIABLE getArrayData =  FUNCTION explode BASED ON " ",VARIABLE filterData
  VARIABLE joinData =  ""
  VARIABLE jmlData =  FUNCTION count BASED ON VARIABLE getArrayData
  FOR(VARIABLE i= 0;VARIABLE i<VARIABLE jmlData;VARIABLE i++)
    VARIABLE joinData.=  FUNCTION ucfirst BASED ON VARIABLE getArrayData[VARIABLE
i]." "
  ENDFOR
  VARIABLE getArrayDataLain =  FUNCTION explode BASED ON " ",VARIABLE joinData
  VARIABLE joinAllData =  FUNCTION array_merge BASED ON VARIABLE
getArrayData,VARIABLE getArrayDataLain
  VARIABLE sql =  "select *from barang where ";
  FOR(VARIABLE i= 0;VARIABLE i<VARIABLE jmlData;VARIABLE i++){
    VARIABLE sql.=  "namaBarang like '%".VARIABLE getArrayData[VARIABLE i]."%' or
deskripsiBarang like '%".VARIABLE getArrayData[VARIABLE i]."%'"
    IF  VARIABLE i<VARIABLE jmlData
       FOR(VARIABLE i= 1;VARIABLE i<VARIABLE jmlData;VARIABLE i++
         VARIABLE sql.=  " or namaBarang like '%".VARIABLE getArrayData[VARIABLE
i]."%' or deskripsiBarang like '%".VARIABLE getArrayData[VARIABLE i]."%'"

       ENDFOR
    ENDIF
  ENDFOR


  VARIABLE sql.=  "order by idBarang desc"
  VARIABLE limit =  4
  VARIABLE start =  VARIABLE paging SET FUNCTION getStart BASED ON VARIABLE limit
  VARIABLE query =  "index.php?page= Barang&amp;go= cekFormCariBarang&amp;q=
".VARIABLE _GET['q']
  VARIABLE sql2 =  VARIABLE sql." limit ".VARIABLE start.",".VARIABLE limit
  VARIABLE resultData =  ""
  FOR(VARIABLE i= 0;VARIABLE i<count(VARIABLE joinAllData);VARIABLE i++)
    VARIABLE resultData.=  VARIABLE joinAllData[VARIABLE i]
  ENDFOR
  VARIABLE getArrayResult =  FUNCTION explode BASED ON " ",VARIABLE resultData
  VARIABLE qryBarang =  VARIABLE db SET FUNCTION getQuery BASED ON VARIABLE sql
  VARIABLE numBrg =  VARIABLE db SET FUNCTION getNum BASED ON VARIABLE qryBarang
  VARIABLE qryBarangList= VARIABLE db->getQuery(VARIABLE sql2);
  VARIABLE error =  VARIABLE getHtmlNonIndex[0]
  IF  VARIABLE _GET['q']= = "Search..."
    VARIABLE error.=  "Data tidak ditemukan"
  ELSEIF  VARIABLE numBrg =  0
    VARIABLE error.=  "Data tidak ditemukan"
  ELSE
    IF  VARIABLE numBrg>VARIABLE limit
      VARIABLE error.=  VARIABLE paging SET FUNCTION setPaging BASED ON VARIABLE
```

```
numBrg,VARIABLE limit,VARIABLE query
    ELSE
      VARIABLE error.=  ''
    ENDIF

    WHILE VARIABLE rowBarang =  VARIABLE db SET FUNCTION getObject BASED ON
VARIABLE qryBarangList
      VARIABLE namaBarang =  VARIABLE rowBarang SET namaBarang
      VARIABLE descBarang =  VARIABLE rowBarang SET deskripsiBarang
      VARIABLE cariNamaBrg =  FUNCTION str_replace BASED ON VARIABLE
joinAllData,VARIABLE getArrayResult,VARIABLE namaBarang
      VARIABLE cariDescBrg =  FUNCTION str_replace BASED ON VARIABLE
joinAllData,VARIABLE getArrayResult,VARIABLE descBarang
      VARIABLE show[] =  VARIABLE cariNamaBrg
      VARIABLE show[].=  VARIABLE cutTxt SET FUNCTION cut BASED ON VARIABLE
cariDescBrg,42
    ENDWHILE

    VARIABLE error.=  "Data yang ditemukan sebanyak : ".VARIABLE numBrg
    FOREACH (VARIABLE show as VARIABLE lsData) VARIABLE error.=  VARIABLE lsData
  ENDIF

  VARIABLE error.=  VARIABLE getHtmlNonIndex[1]
  VARIABLE formData['isi'] =  VARIABLE error
ELSE
  VARIABLE formData['isi']=  ''
ENDIF
VARIABLE formData['getFormCari'] =  VARIABLE formCari

RETURN VARIABLE formData
```

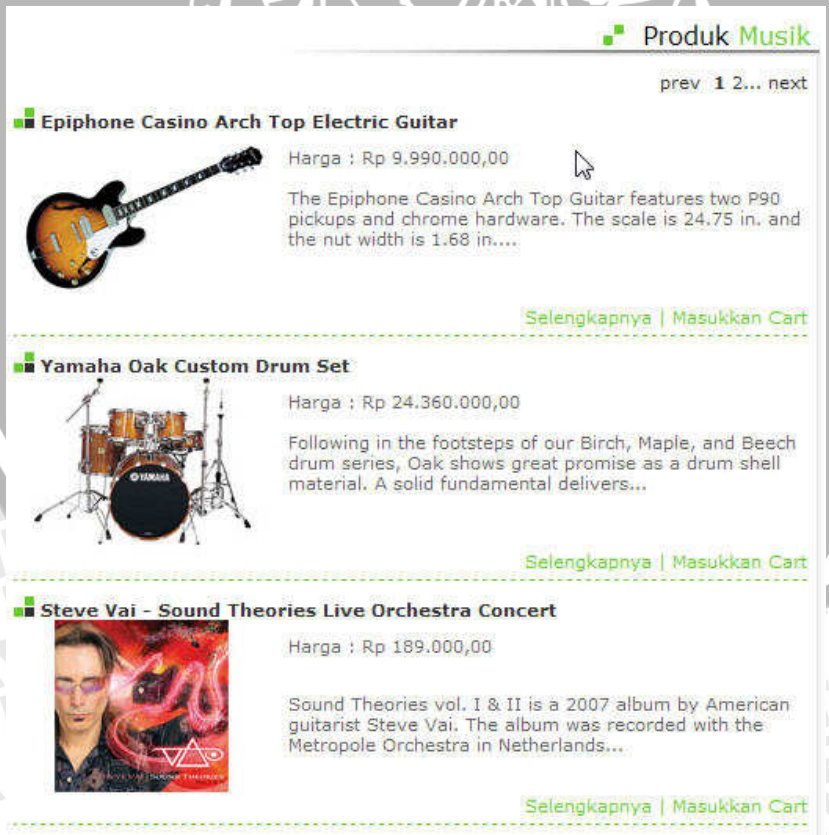
5.4.1.10. Implementasi Antarmuka Kebutuhan Fungsional Melihat Data Katalog Barang

Semua aktor dari Aplikasi *Web Service E-Commerce* dapat melihat data katalog barang. Untuk melakukan proses tersebut, *user* harus memilih menu ‘produk’ pada halaman *website*. Atau memilih salah satu submenu kategori produk yang terdapat di bagian kiri halaman *website*. Gambar 5.23 menunjukkan implementasi antarmuka untuk melihat data katalog barang.



Gambar 5. 23 : Implementasi Antarmuka untuk Melihat Data Katalog Barang
Sumber : [Implementasi]

Sistem menampilkan data barang terbaru apabila *user* memilih menu 'produk'. Dan sistem akan menampilkan data barang per-kategori apabila *user* memilih salah satu submenu kategori produk. Gambar 5.24 menunjukkan hasil implementasi antarmuka untuk mencari data barang.



Gambar 5. 24 : Hasil Implementasi Antarmuka untuk Melihat Data Katalog Barang
Sumber : [Implementasi]

Implementasi algoritma untuk Melihat Data Katalog Barang ditunjukkan dalam tabel berikut :

Algoritma 5. 10 : Algoritma Melihat Data Katalog Barang

```
FUNCTION listBarang

VARIABLE paging = new FUNCTION paging
VARIABLE dbBarang = new FUNCTION dbBarang

IF FUNCTION not isset BASED ON VARIABLE _GET['admin'] THEN
  IF VARIABLE index = 1 THEN
    VARIABLE cutTxt = new FUNCTION textCutting
    VARIABLE listBarang = VARIABLE dbBarang SET FUNCTION listBarang BASED ON 0,3
    VARIABLE htmlIndex = new FUNCTION htmlLayout
    VARIABLE getHtmlIndex = VARIABLE htmlIndex SET FUNCTION htmlIndex
    foreach VARIABLE listBarang as VARIABLE lsBarang
    BEGIN
      VARIABLE ls[] = VARIABLE getHtmlIndex[1]
      VARIABLE ls[].= VARIABLE lsBarang SET idBarang.VARIABLE lsBarang SET
namaBarang
      VARIABLE ls[].= VARIABLE getHtmlIndex[2]
      VARIABLE ls[].= VARIABLE lsBarang SET gambarBarang
      VARIABLE ls[].= "Rp ".FUNCTION number_format BASED ON VARIABLE lsBarang SET
hargaBarang,2,',','.'
      VARIABLE ls[].= VARIABLE getHtmlIndex[3]
      VARIABLE ls[].= VARIABLE cutTxtSET SET FUNCTION cut BASED ON VARIABLE
lsBarang SET deskripsiBarang,25
      VARIABLE ls[].= VARIABLE lsBarang SET idBarang
      VARIABLE ls[].= “masukkan cart”
      VARIABLE ls[].= VARIABLE getHtmlIndex[4];
    END

    VARIABLE setHtml= VARIABLE getHtmlIndex[0];
    foreach VARIABLE ls as VARIABLE getDataBarang
    BEGIN
      VARIABLE setHtml.= VARIABLE getDataBarang
    END
    VARIABLE setHtml.= "Produk Lainnya... ";
    VARIABLE dataBarang = VARIABLE setHtml;
  ENDIF

  IF FUNCTION isset BASED ON VARIABLE _GET['go'] THEN
    VARIABLE limit = 4
    VARIABLE start = VARIABLE paging SET FUNCTION getStart BASED ON VARIABLE limit
    VARIABLE cutTxt = new FUNCTION textCutting
    VARIABLE listBarang = VARIABLE dbBarang SET FUNCTION listBarang BASED ON
VARIABLE start,VARIABLE limit
    VARIABLE numBarang = VARIABLE dbBarangSET numBarang
    VARIABLE htmlIndex = new FUNCTION  htmlLayout
    VARIABLE getHtmlNonIndex = VARIABLE htmlIndex SET FUNCTION htmlNonIndex BASED
ON 'Produk','Musik'
    foreach VARIABLE listBarang as VARIABLE lsBarang
    BEGIN
      VARIABLE ls[] = VARIABLE lsBarang SET namaBarang
      VARIABLE ls[].= VARIABLE lsBarang SET gambarBarang
      VARIABLE ls[].= "Harga : Rp ".FUNCTION number_format BASED ON VARIABLE
lsBarang SET hargaBarang,2,',','.'
      VARIABLE ls[].= VARIABLE cutTxt SET FUNCTION cut BASED ON VARIABLE lsBarang
SET deskripsiBarang,25
```

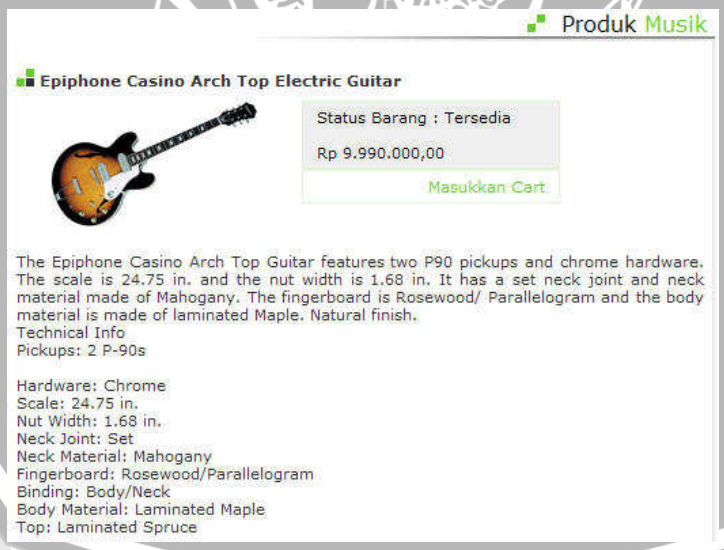
```
      VARIABLE ls[].= "Selengkapnya|Masukkan Cart
    END

    VARIABLE query = "index.php?page= Barang&amp;amp;go= listBarang&amp;amp;"
    VARIABLE setHtml = VARIABLE getHtmlNonIndex[0]

    IF VARIABLE numBarang>VARIABLE limit THEN
      VARIABLE setHtml.= VARIABLE paging SET FUNCTION setPaging BASED ON VARIABLE
numBarang,VARIABLE limit,VARIABLE query
    ELSE
      VARIABLE setHtml.= ''
    ENDIF

    foreach VARIABLE ls as VARIABLE getDataBarang
    BEGIN
      VARIABLE setHtml.= VARIABLE getDataBarang
    END
    VARIABLE setHtml.= VARIABLE getHtmlNonIndex[1]
    VARIABLE dataBarang['isi'] = VARIABLE setHtml
  ENDIF
ELSE
  VARIABLE limit = 3
  VARIABLE start = VARIABLE paging SET getStart FUNCTION VARIABLE limit
  VARIABLE cutTxt = new FUNCTION textCutting
  VARIABLE listBarang = VARIABLE dbBarang SET listBarang FUNCTION VARIABLE
start,VARIABLE limit
  VARIABLE numBarang = VARIABLE dbBarang SET FUNCTION numBarang

  foreach VARIABLE listBarang as VARIABLE lsBarang
  BEGIN
    VARIABLE idBarang = VARIABLE lsBarang SET idBarang
    VARIABLE ls[].= VARIABLE lsBarang SET namaBarang
    VARIABLE ls[].= VARIABLE lsBarang SET gambarBarang
    VARIABLE ls[].= VARIABLE lsBarang SET kategoriBarang
    VARIABLE ls[].= VARIABLE lsBarang SET stokBarang
    VARIABLE ls[].= VARIABLE lsBarang SET jenisBeratKirim
    VARIABLE ls[].= "Rp ".FUNCTION number_format BASED ON VARIABLE lsBarangSET
hargaBarang,2,',','.'
    VARIABLE ls[].= VARIABLE cutTxt SET FUNCTION cut BASED ON VARIABLE lsBarang
SET deskripsiBarang,25
  END
  VARIABLE query = "index.php?admin= cpanel&amp;amp;page= Barang&amp;amp;";

  IF VARIABLE numBarang>VARIABLE limit THEN
    VARIABLE setHtml.= VARIABLE paging SET FUNCTION setPaging BASED ON VARIABLE
numBarang,VARIABLE limit,VARIABLE query
  ELSE
    VARIABLE setHtml.= ''
  ENDIF

  foreach VARIABLE ls as VARIABLE barang
  BEGIN
    VARIABLE setHtml.= VARIABLE barang
  END
  VARIABLE dataBarang = VARIABLE setHtml
ENDIF

RETURN VARIABLE dataBarang
```

5.4.1.11. Implementasi Antarmuka Kebutuhan Fungsional Melihat Data Detail Barang

Aplikasi *Web Service E-Commerce* memiliki fasilitas melihat data detail barang. Setiap aktor dapat mengakses fasilitas ini setelah melakukan proses melihat data katalog barang. Gambar 5.25 menunjukkan implementasi antarmuka untuk melihat data detail barang.

Gambar 5. 25 : Implementasi Antarmuka untuk Melihat Data Detail Barang
Sumber : [Implementasi]

User harus menekan tombol '`selengkapnya`' untuk melihat data detail barang. Gambar 5.24 menunjukkan hasil implementasi antarmuka untuk melihat data detail barang.



Gambar 5. 26 : Hasil Implementasi Antarmuka untuk Melihat Data Detail Barang
Sumber : [Implementasi]

Implementasi algoritma untuk Melihat Data Detail Barang ditunjukkan dalam tabel berikut :

Algoritma 5. 11 : Algoritma Melihat Data Detail Barang

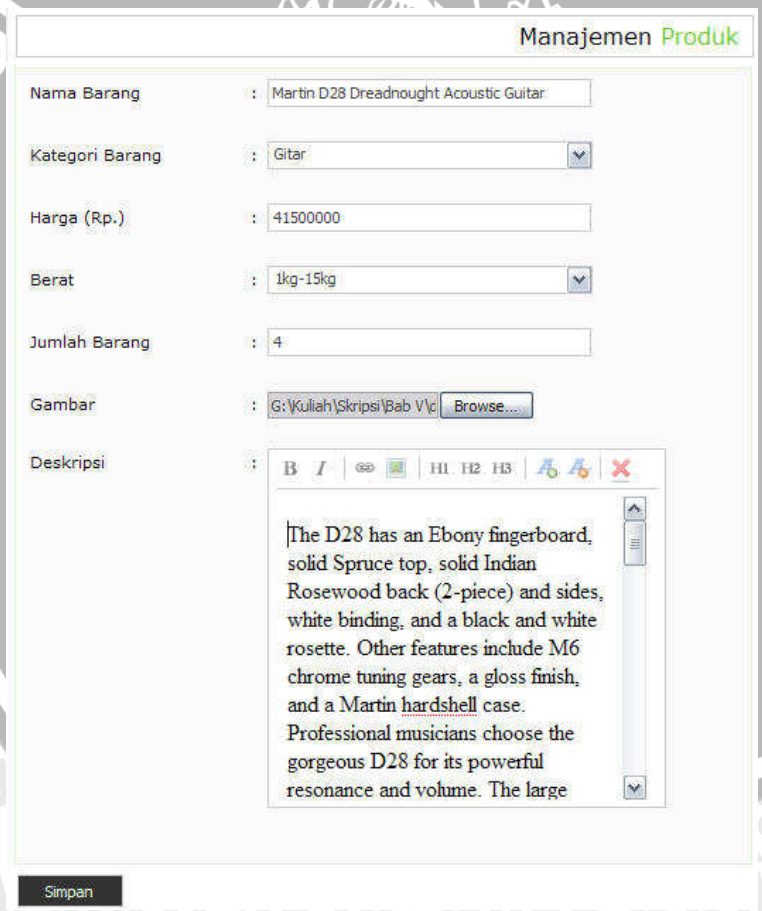
```
FUNCTION detBarang

VARIABLE htmlIndex = new FUNCTION htmlLayout
VARIABLE getHtmlNonIndex = VARIABLE htmlIndex SET FUNCTION htmlNonIndex BASED ON
'Produk','Musik'
VARIABLE dbBarang = new FUNCTION dbBarang
VARIABLE id = VARIABLE _GET['Id']
VARIABLE isiBarang = VARIABLE dbBarang SET FUNCTION detBarang BASED VARIABLE id
VARIABLE numBarang = VARIABLE dbBarang SET FUNCTION numBarang BASED ON VARIABLE id

IF VARIABLE _GET['admin'] THEN
  VARIABLE idBarang = VARIABLE isiBarang SET idBarang
  VARIABLE ls = VARIABLE isiBarang SET namaBarang
  VARIABLE ls.= VARIABLE isiBarang SET gambarBarang
  VARIABLE ls.= VARIABLE isiBarang SET stokBarang
  VARIABLE ls.= VARIABLE isiBarang SET jenisBeratKirim
  VARIABLE ls.= "Rp ".FUNCTION number_format BASED ON VARIABLE isiBarang SET
hargaBarang,2,',','.'
  VARIABLE ls.= VARIABLE isiBarang SET deskripsiBarang
  VARIABLE dataBarang['isi'] = VARIABLE ls
ELSE
  VARIABLE ls = VARIABLE getHtmlNonIndex[0]
  VARIABLE ls.= VARIABLE isiBarang SET namaBarang
  VARIABLE ls.= VARIABLE isiBarang SET gambarBarang
  VARIABLE ls.="Status Barang : ";
  IF VARIABLE numBarang = 0 THEN
    VARIABLE ls.= "Kosong"
  ELSE
    VARIABLE ls.="Tersedia"
    VARIABLE ls.= "Rp ".FUNCTION number_format BASED ON VARIABLE isiBarang SET
hargaBarang,2,',','.'
    VARIABLE ls.= "Masukkan Cart"
    VARIABLE ls.= VARIABLE isiBarang SET deskripsiBarang
    VARIABLE ls.= VARIABLE getHtmlNonIndex[1]
    VARIABLE dataBarang['isi'] = VARIABLE ls
  ENDIF
ENDIF

RETURN VARIABLE dataBarang
```

5.4.1.12. Implementasi Antarmuka Kebutuhan Fungsional Menambah Barang

Fasilitas menambah barang hanya dapat diakses oleh *administrator*. Fasilitas ini dapat diakses melalui halaman administrasi *website*. *Administrator* harus memilih menu '`tambah produk`' untuk menampilkan *form* tambah produk. Gambar 5.27 menunjukkan implementasi antarmuka untuk menambah barang.

Gambar 5. 27 : Implementasi Antarmuka untuk Menambah Barang
Sumber : [Implementasi]

Setelah sistem menampilkan *form* tambah produk, *administrator* dapat mengisi *field* yang tersedia dengan data produk. Gambar 5.28 menunjukkan hasil implementasi antarmuka untuk menambah barang.



Gambar 5. 28 : Hasil Implementasi Antarmuka untuk Menambah Barang
Sumber : [Implementasi]

Implementasi algoritma untuk Menambah Barang ditunjukkan dalam tabel berikut :

Algoritma 5. 12 : Algoritma Menambah Barang

```
FUNCTION cekFormTambahBarang

IF FUNCTION isset VARIABLE _POST['tambahBarang'] THEN
  VARIABLE fieldBarang = array()
  foreach VARIABLE fieldBarang as VARIABLE keyFieldBarang=>VARIABLE valFieldBarang
  BEGIN
  IF FUNCTION empty BASED ON VARIABLE _POST[VARIABLE keyFieldBarang] THEN
    VARIABLE error[VARIABLE keyFieldBarang] = VARIABLE valFieldBarang
    VARIABLE post = ''
    VARIABLE dataBarang = ''
  ELSE
    VARIABLE error[VARIABLE keyFieldBarang] = " "
    IF VARIABLE keyFieldBarang='katBarang' or VARIABLE
keyFieldBarang='beratBarang' THEN
      VARIABLE brg['katBarang'] = VARIABLE _POST['katBarang']
      VARIABLE brg['beratBarang'] = VARIABLE _POST['beratBarang']
      VARIABLE detBarang = VARIABLE this SET dbKatBarang SET FUNCTION
detDataPendukungBarang BASED ON VARIABLE brg
      VARIABLE detBerat = VARIABLE this SET dbBeratBarang SET FUNCTION
detDataPendukungBarang BASED ON VARIABLE brg
      VARIABLE post['katBarang'] = VARIABLE detBarang SET idKategoriBarang
      VARIABLE post['beratBarang'] = VARIABLE detBerat SET idBeratKirim
    ELSE
      VARIABLE post[VARIABLE keyFieldBarang] = VARIABLE _POST[VARIABLE
keyFieldBarang]
    ENDIF
  ENDIF
  END

  IF FUNCTION empty BASED ON VARIABLE _FILES['gbBarang']['name'] THEN
    VARIABLE error['gbBarang'] = “Gambar harus diisi.”
    VARIABLE post = ''
  ELSE
    VARIABLE error['gbBarang'] = “ ”
    VARIABLE post['gbBarang'] = VARIABLE _FILES['gbBarang']['name']
  ENDIF

  IF not FUNCTION empty BASED ON VARIABLE _POST['hargaBarang'] and not FUNCTION
is_numeric BASED ON VARIABLE _POST['hargaBarang'] THEN
    VARIABLE error['hargaBarang'] = “Harga barang salah.”
  ENDIF
  IF not FUNCTION empty BASED ON VARIABLE _POST['jmlBarang'] and not FUNCTION
is_numeric BASED ON VARIABLE _POST['jmlBarang'] THEN
    VARIABLE error['jmlBarang']=”Jumlah barang salah.”
  ENDIF
  IF VARIABLE error[] = “ ” THEN
    VARIABLE msgBarang = “Data Barang berhasil disimpan.”
    VARIABLE this SET FUNCTION tambahBarang BASED ON VARIABLE post
  ELSE
    VARIABLE msgBarang = ''
  ENDIF

  VARIABLE data = VARIABLE this SET FUNCTION formTambahBarang BASED ON VARIABLE
error,VARIABLE msgBarang
ELSE
  VARIABLE data['isi'] = ''
```

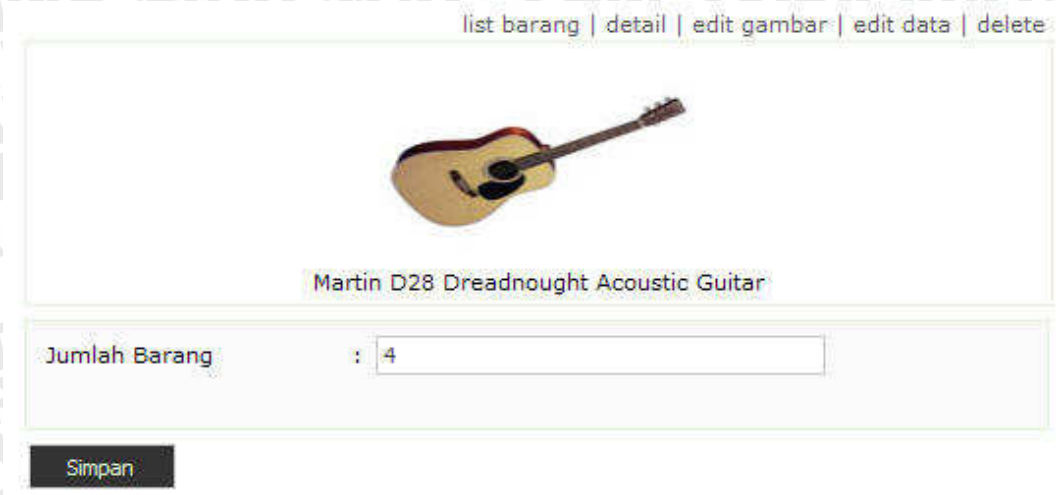
```
ENDIF

RETURN VARIABLE data
```

5.4.1.13. Implementasi Antarmuka Kebutuhan Fungsional Mengedit Stok Barang

Fasilitas mengedit stok barang hanya dapat diakses oleh *administrator*. Fasilitas ini dapat diakses melalui halaman administrasi *website*. *Administrator* harus memilih menu '`edit stok`' untuk menampilkan *form* edit stok barang. Gambar 5.29 menunjukkan implementasi antarmuka untuk edit stok barang.

Gambar 5. 29 : Implementasi Antarmuka untuk Mengedit Stok Barang
Sumber : [Implementasi]

Setelah sistem menampilkan *form* edit stok varang, *administrator* dapat mengisi *field* yang tersedia dengan jumlah stok produk. Gambar 5.27 menunjukkan hasil implementasi antarmuka untuk menambah barang.



Gambar 5. 30 : Hasil Implementasi Antarmuka untuk Mengedit Stok Barang
Sumber : [Implementasi]

Implementasi algoritma untuk Mengedit Stok Barang ditunjukkan dalam tabel berikut :

Algoritma 5. 13 : Algoritma Mengedit Stok Barang

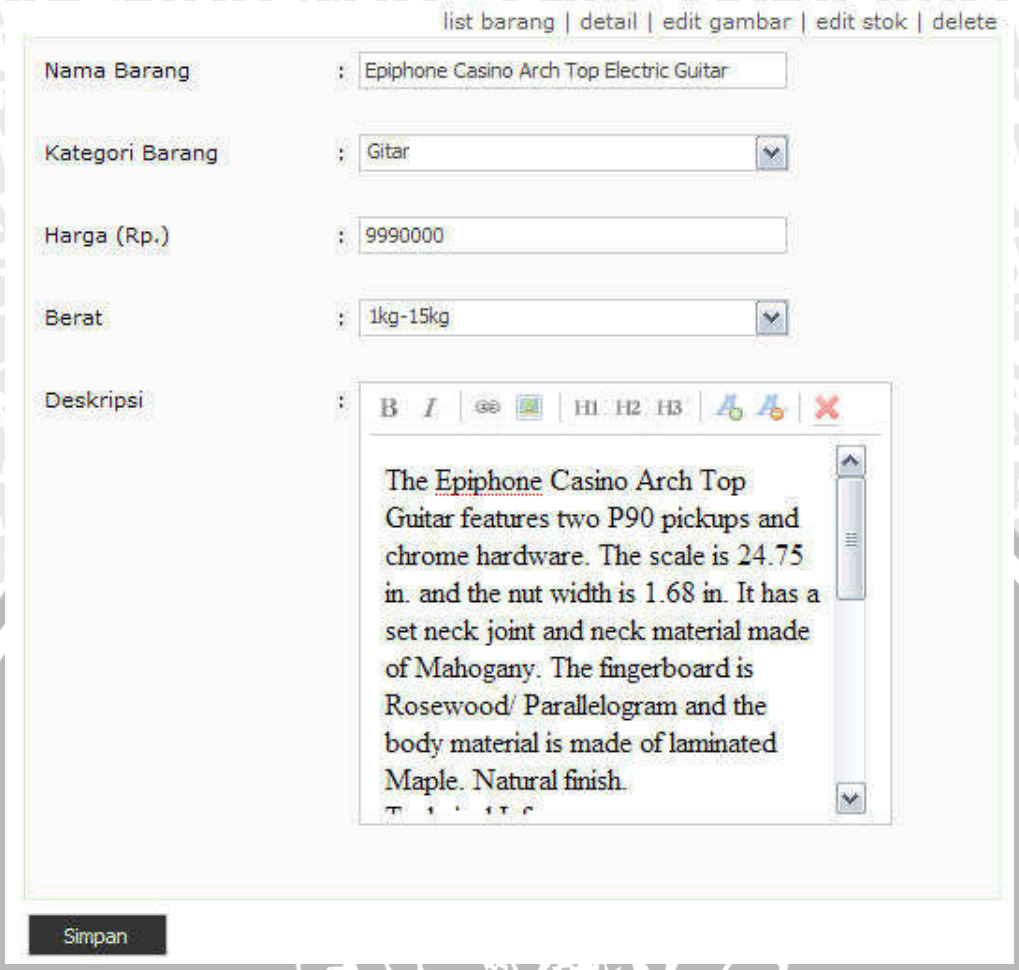
```
FUNCTION cekFormEditStokBarang

VARIABLE id = VARIABLE _GET['Id']
IF FUNCTION isset BASED ON VARIABLE _POST['tambahStokBarang'] THEN
  IF FUNCTION empty BASED ON VARIABLE _POST['jmlBarang'] THEN
    VARIABLE error['jmlBarang'] = "Jumlah Barang harus diisi. "
    VARIABLE post = ''
  ELSE
    VARIABLE error['jmlBarang'] = " "
    VARIABLE post['jmlBarang'] = VARIABLE _POST['jmlBarang']
  ENDIF
  IF NOT FUNCTION empty BASED ON VARIABLE _POST['jmlBarang'] and not FUNCTION
is_numeric BASED ON VARIABLE _POST['jmlBarang'] THEN
    VARIABLE error['jmlBarang'] = "Jumlah barang salah."
  ENDIF
  IF(VARIABLE error['jmlBarang'] = " " THEN
    VARIABLE msgBarang = "Data Barang berhasil disimpan."
    VARIABLE this SET FUNCTION editStokBarang BASED ON VARIABLE post,VARIABLE id
  ELSE
    VARIABLE msgBarang = ''
  ENDIF
    VARIABLE data = VARIABLE this SET FUNCTION formEditStokBarang BASED ON
VARIABLE error,VARIABLE msgBarang
ELSE
  VARIABLE data['isi'] = ""
ENDIF

RETURN VARIABLE data
```

5.4.1.14. Implementasi Antarmuka Kebutuhan Fungsional Mengedit Data Barang

Fasilitas mengedit data barang hanya dapat diakses oleh *administrator*. Fasilitas ini dapat diakses melalui halaman administrasi *website*. *Administrator* harus memilih menu '`edit data`' untuk menampilkan *form* edit data barang. Gambar 5.31 menunjukkan implementasi antarmuka untuk mengedit data barang.

Gambar 5. 31 : Implementasi Antarmuka untuk Mengedit Data Barang
Sumber : [Implementasi]

Setelah sistem menampilkan *form* edit data barang, *administrator* dapat mengisi *field* yang tersedia dengan data produk baru. Gambar 5.32 menunjukkan hasil implementasi antarmuka untuk menambah barang.



Gambar 5. 32 : Hasil Implementasi Antarmuka untuk Mengedit Data Barang
Sumber : [Implementasi]

Implementasi algoritma untuk Mengedit Data Barang ditunjukkan dalam tabel berikut :

Algoritma 5. 14 : Algoritma Mengedit Data Barang

```
FUNCTION cekformEditBarang

IF FUNCTION isset BASED ON VARIABLE _POST['editBarang'] THEN
  VARIABLE id=VARIABLE _GET['Id']
  VARIABLE fieldBarang = FUNCTION array()
  foreach VARIABLE fieldBarang as VARIABLE keyFieldBarang=>VARIABLE valFieldBarang
  BEGIN
  IF(FUNCTION empty VARIABLE _POST[VARIABLE keyFieldBarang]THEN
    VARIABLE error[VARIABLE keyFieldBarang] = VARIABLE valFieldBarang." harus
diisi."
    VARIABLE post = ''
    VARIABLE dataBarang = ''
  ELSE
    VARIABLE error[VARIABLE keyFieldBarang] = " "
    IF VARIABLE keyFieldBarang = 'editKatBrg' or VARIABLE keyFieldBarang =
'editBeratBrg' THEN
      VARIABLE brg['katBarang'] = VARIABLE _POST['editKatBrg']
      VARIABLE brg['beratBarang'] = VARIABLE _POST['editBeratBrg']
```

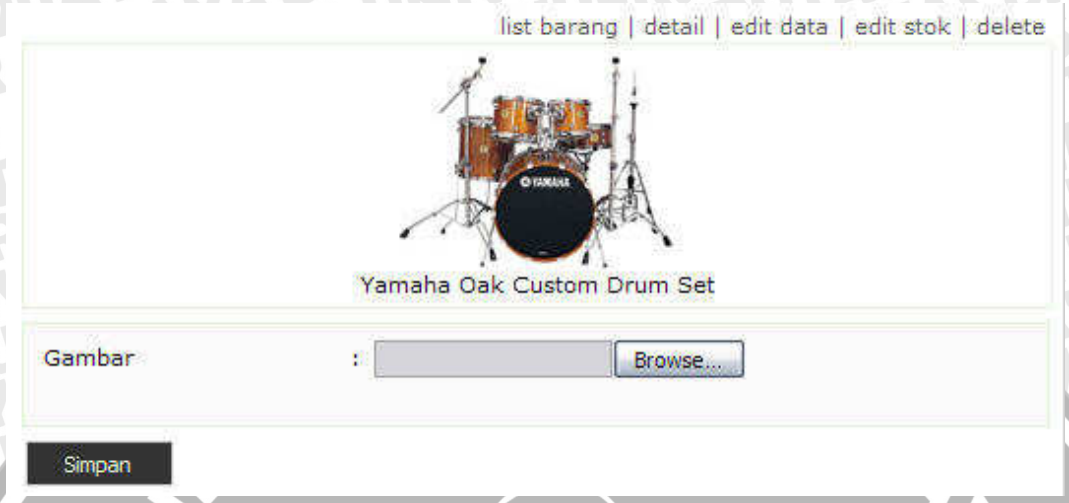
```
      VARIABLE detBarang = VARIABLE this SET dbKatBarang SET FUNCTION
detDataPendukungBarang BASED ON VARIABLE brg
      VARIABLE detBerat = VARIABLE this SET dbBeratBarang SET FUNCTION
detDataPendukungBarang BASED ON VARIABLE brg
      VARIABLE post['editKatBrg'] = VARIABLE detBarang SET idKategoriBarang
      VARIABLE post['editBeratBrg'] = VARIABLE detBerat SET idBeratKirim
    ENDIF
    VARIABLE post[VARIABLE keyFieldBarang] = VARIABLE _POST[VARIABLE
keyFieldBarang]
  ENDIF
  END

  IF not FUNCTION empty BASED ON VARIABLE _POST['editHargaBrg'] and not FUNCTION
is_numeric BASED ON VARIABLE _POST['editHargaBrg'] THEN
    VARIABLE error['editHargaBrg'] = "Harga barang salah."
  ENDIF
  IF VARIABLE error[] = " " THEN
    VARIABLE msgBarang = "Data Barang berhasil disimpan."
    VARIABLE this SET FUNCTION editBarang BASED ON VARIABLE post,VARIABLE id
  ELSE
    VARIABLE msgBarang = ''
  ENDIF
  VARIABLE data = VARIABLE this SET formEditBarang BASED ON VARIABLE
error,VARIABLE msgBarang
ELSE
  VARIABLE data['inc'] = ""
  VARIABLE data['isi'] = ""
ENDIF

RETURN VARIABLE data
```

5.4.1.15. Implementasi Antarmuka Kebutuhan Fungsional Mengedit Gambar Barang

Fasilitas mengedit gambart barang hanya dapat diakses oleh *administrator*. Fasilitas ini dapat diakses melalui halaman administrasi *website*. *Administrator* harus memilih menu '`edit gambar`' untuk menampilkan *form* edit gambar barang. Gambar 5.33 menunjukkan implementasi antarmuka untuk mengedit gambar barang.

Gambar 5. 33 : Implementasi Antarmuka untuk Mengedit Gambar Barang
Sumber : [Implementasi]

Setelah sistem menampilkan *form* edit gambar barang, *administrator* dapat mengisi *field* dengan gambar barang baru. Gambar 5.34 menunjukkan hasil implementasi antarmuka untuk menambah barang.



Gambar 5. 34 : Hasil Implementasi Antarmuka untuk Mengedit Gambar Barang
Sumber : [Implementasi]

Implementasi algoritma untuk Mengedit Gambar Barang ditunjukkan dalam tabel berikut :

Algoritma 5. 15 : Algoritma Mengedit Gambar Barang

```
FUNCTION cekFormEditGbBarang

VARIABLE id = VARIABLE _GET['Id'];
IF FUNCTION isset BASED ON VARIABLE _POST['editGbBarang'] THEN
  IF FUNCTION empty BASED ON VARIABLE _FILES['editGbBrg']['name'] THEN
    VARIABLE error['editGbBrg'] = "Gambar harus diisi."
    VARIABLE post = ''
  ELSE
    VARIABLE error['editGbBrg'] = " "
    VARIABLE post['editGbBrg'] = VARIABLE _FILES['editGbBrg']['name']
  ENDIF
  IF VARIABLE error['editGbBrg'] = " " THEN
    VARIABLE msgBarang = "Data Barang berhasil disimpan."
    VARIABLE this SET FUNTION editGbBarang BASED ON VARIABLE post,VARIABLE id
  ELSE
    VARIABLE msgBarang = ''
  ENDIF
  VARIABLE data = VARIABLE this SET FUNCTION formEditGbBarang BASED ON VARIABLE
error,VARIABLE msgBarang
ELSE
  VARIABLE data['isi'] = ""
ENDIF

RETURN VARIABLE data
```

5.4.1.16. Implementasi Antarmuka Kebutuhan Fungsional Menghapus Data Barang

Fasilitas menghapus data barang hanya dapat diakses oleh *administrator*. Fasilitas ini dapat diakses melalui halaman administrasi *website*. *Administrator* harus memilih menu ‘hapus’. Gambar 5.35 menunjukkan implementasi antarmuka untuk menghapus data barang.

Gambar 5. 35 : Implementasi Antarmuka untuk Menghapus Data Barang
Sumber : [Implementasi]

Setelah *administrator* menekan tombol ‘hapus’, maka sistem menampilkan pesan tentang penghapusan barang tersebut. Gambar 5.36 menunjukkan hasil implementasi antarmuka untuk menghapus data barang.

Gambar 5. 36 : Hasil Implementasi Antarmuka untuk Menghapus Data Barang
Sumber : [Implementasi]

Implementasi algoritma untuk Menghapus Data Barang ditunjukkan dalam tabel berikut :

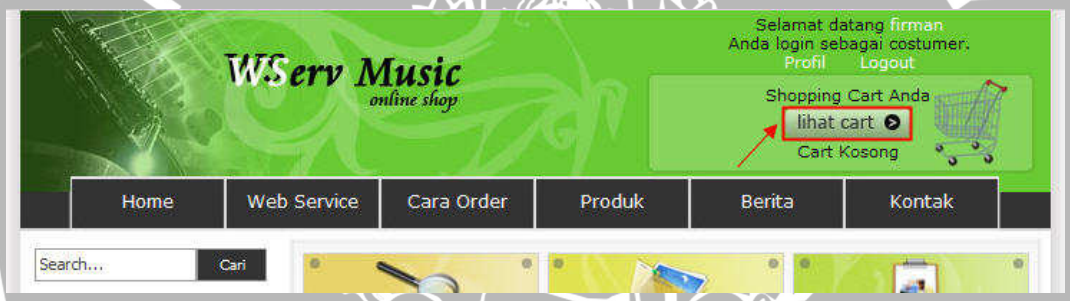
Algoritma 5. 16 : Algoritma Menghapus Data Barang

```
FUNCTION hapusBarang

IF FUNCTION isset BASED ON VARIABLE _SESSION['login']) and VARIABLE
_SESSION['login'] = true THEN
  VARIABLE dbBarang = new FUNCTION dbBarang
  VARIABLE id = VARIABLE _GET['Id']
  VARIABLE detBarang = VARIABLE dbBarang SET FUNCTION detBarang BASED ON VARIABLE
id
```

```
  VARIABLE setHtml = "Data produk untuk ".VARIABLE detBarang SET namaBarang."
telah berhasil dihapus."
  VARIABLE fileGb = VARIABLE detBarang SET FUNCTION gambarBarang
  FUNCTION unlink BASED ON VARIABLE fileGb
  VARIABLE dbBarang SET FUNCTION hapusBarang BASED ON VARIABLE id
  VARIABLE data['isi'] = VARIABLE setHtml
ELSE
  VARIABLE data['isi'] = ''
ENDIF

RETURN VARIABLE data
```

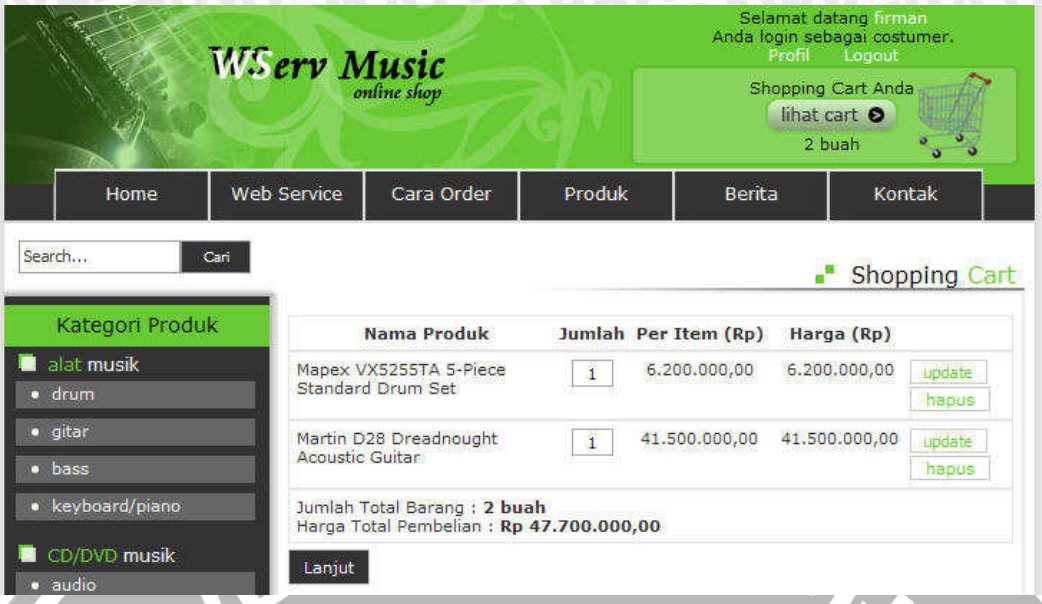
5.4.1.17. Implementasi Antarmuka Kebutuhan Fungsional Melihat *Shopping Cart*

Fasilitas melihat *shopping cart* dapat diakses oleh aktor yang terdaftar pada Aplikasi *Web Service E-Commerce*. Untuk melihat *shopping cart*, *user* dapan menekan tombol '`lihat cart`' di bagian *header website*. Gambar 5.37 menunjukkan implementasi antarmuka untuk melihat *shopping cart*.



Gambar 5. 37 : Implementasi Antarmuka untuk Melihat *Shopping Cart*
Sumber : [Implementasi]

Setelah *user* menekan tombol '`lihat cart`', sistem menampilkan daftar produk yang dipesan oleh *user*. Gambar 5.38 menunjukkan hasil implementasi antarmuka untuk menambah barang.



Gambar 5. 38 : Hasil Implementasi Antarmuka untuk Melihat *Shopping Cart*
Sumber : [Implementasi]

Implementasi algoritma untuk Melihat *Shopping Cart* ditunjukkan dalam tabel berikut :

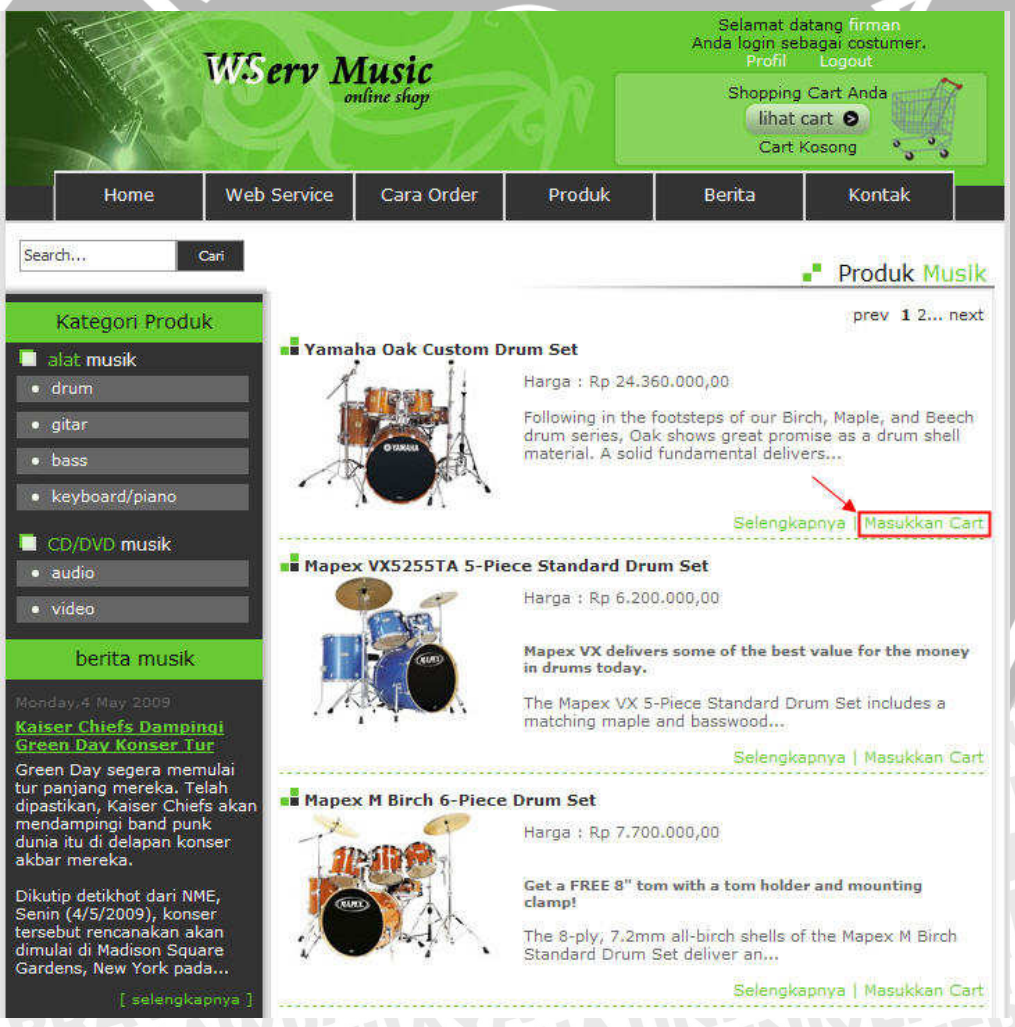
Algoritma 5. 17 : Algoritma Melihat *Shopping Cart*

```
FUNCTION listCart

VARIABLE dbCart = new FUNCTION dbShoppingCart
VARIABLE idUser = VARIABLE _SESSION['idUser']
VARIABLE tglPesan = FUNCTION date BASED ON "Y-m-d"
VARIABLE listCart = VARIABLE dbCart SET FUNCTION listCart BASED ON VARIABLE
idUser,1
VARIABLE sumCart = VARIABLE dbCart SET FUNCTION sumCart BASED ON VARIABLE idUser

foreach VARIABLE listCart as VARIABLE lsCart
BEGIN
  VARIABLE setHtml = VARIABLE lsCart SET namaBarang
  VARIABLE setHtml.= VARIABLE lsCart SET jmlPesan
  VARIABLE totHarga = VARIABLE lsCart SET jmlPesan*VARIABLE lsCart SET hargaBarang
  VARIABLE setHtml.= FUNCTION number_format BASED ON VARIABLE lsCart SET
hargaBarang,2,',','.'
  VARIABLE setHtml.= FUNCTION number_format BASED ON VARIABLE totHarga,2,',','.'
  VARIABLE arrayTotal[ ] = VARIABLE totHarga
  VARIABLE idStatusBayar[] = VARIABLE lsCart SET idStatusBayar
END

VARIABLE totalPembelian = FUNCTION array_sum BASED ON VARIABLE arrayTotal
```

```
VARIABLE setHtml.= VARIABLE sumCart SET total
VARIABLE setHtml.= " Rp ".FUNCTION number_format BASED ON VARIABLE
totalPembelian,2,',','.'

RETURN VARIABLE setHtml
```

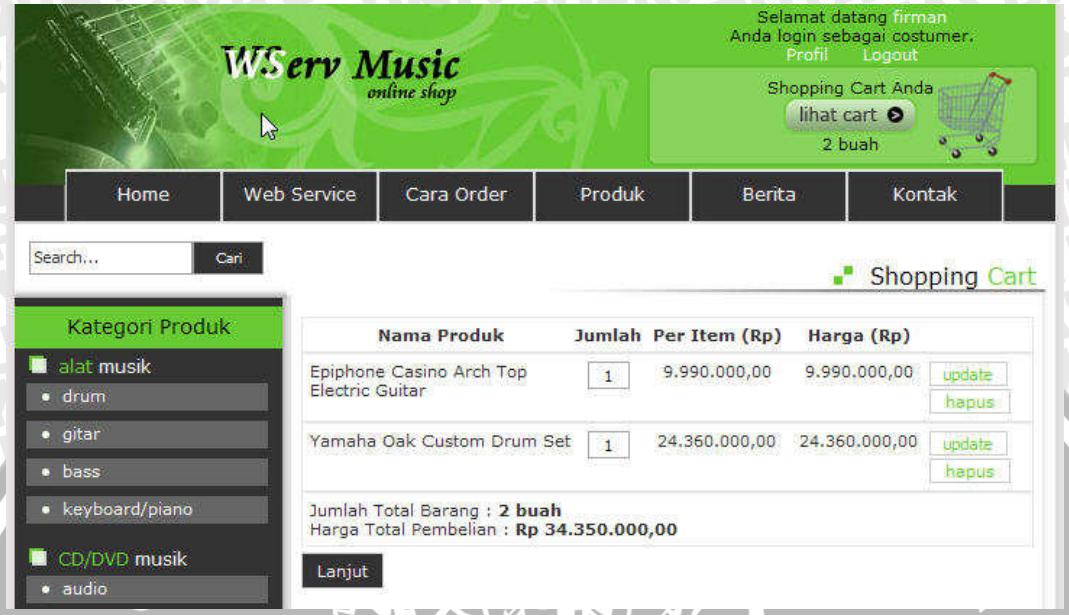
5.4.1.18. Implementasi Antarmuka Kebutuhan Fungsional Menambah *Item Shopping Cart*

Fasilitas menambah *item shopping cart* hanya dapat diakses oleh *user* yang telah terdaftar pada sistem. Fasilitas ini dapat diakses melalui halaman utama *website*. Untuk menambah *item shopping cart*, *user* menekan tombol '`masukkan cart`' pada halaman produk. Gambar 5.39 menunjukkan implementasi antarmuka untuk menambah *item shopping cart*.



Gambar 5. 39 : Implementasi Antarmuka untuk Menambah Item *Shopping Cart*
Sumber : [Implementasi]

Setelah *user* menekan tombol '`masukkan cart`' pada halaman produk, sistem menampilkan halaman daftar *shopping cart*. Gambar 5.40 menunjukkan hasil implementasi antarmuka untuk menambah *item shopping cart*.



Gambar 5. 40 : Hasil Implementasi Antarmuka untuk Menambah Item *Shopping Cart*
Sumber : [Implementasi]

Implementasi algoritma untuk Menambah *Item Shopping Cart* ditunjukkan dalam tabel berikut :

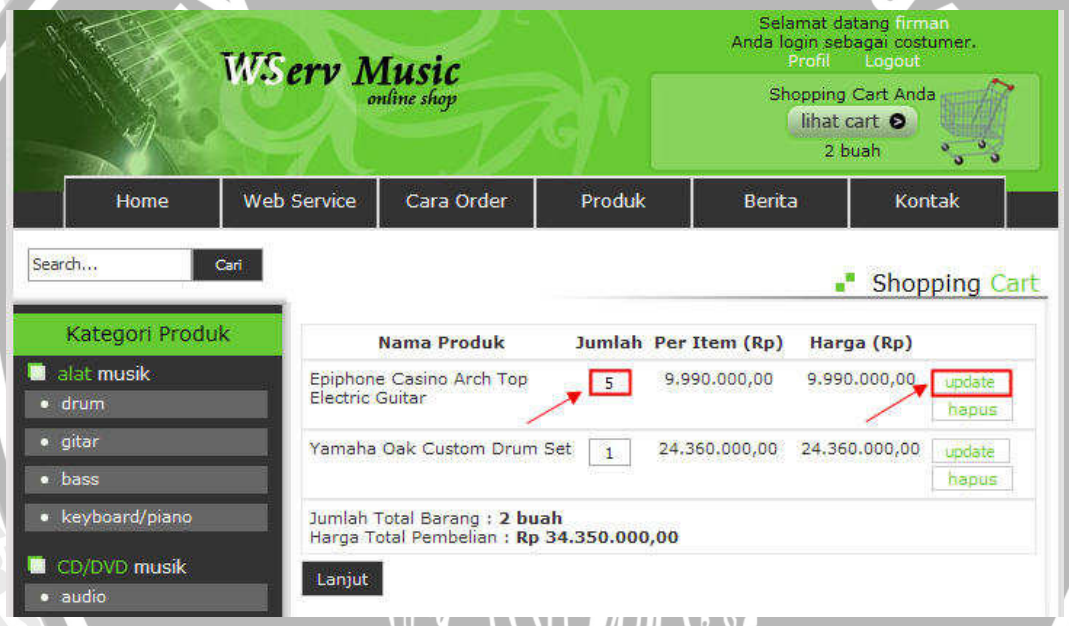
Algoritma 5. 18 : Algoritma Menambah *Item Shopping Cart*

```
FUNCTION tambahCart

VARIABLE dbCart = new FUNCTION dbShoppingCart
VARIABLE hari = FUNCTION date BASED ON "j"
VARIABLE bln = FUNTION date BASED ON "n"
VARIABLE thn = FUNCTION date BASED ON "Y"
VARIABLE tglRemove = FUNCTION date BASED ON "Y-m-d",FUNCTION mktime BASED ON
0,0,0,VARIABLE bln,VARIABLE hari+1,VARIABLE thn

IF FUNCTION isset BASED ON VARIABLE _SESSION['login']) and VARIABLE
_SESSION['login'] = true THEN
  VARIABLE getCart['idUser'] = VARIABLE _SESSION['idUser']
  VARIABLE getCart['idBarang'] = VARIABLE _GET['order']
  VARIABLE getCart['sessionId'] = FUNCTION session_id
  VARIABLE getCart['tglShop'] = FUNCTION date BASED ON"Y-m-d"
  VARIABLE getCart['tglRemove'] = VARIABLE tglRemove
  VARIABLE numCart = VARIABLE dbCart SET FUNCTION numCart BASED ON VARIABLE
getCart['idUser'],VARIABLE getCart['idBarang']

  IF VARIABLE numCart=1 THEN
    VARIABLE det = VARIABLE dbCart SET FUNCTION detCart BASED ON VARIABLE
getCart['idUser'],VARIABLE getCart['idBarang'
    VARIABLE idCart = VARIABLE det SET idShoppingCart
    VARIABLE jml = VARIABLE det SET FUNCTION jmlPesan+1
    VARIABLE dbCart SET FUNCTION editCart BASED ON VARIABLE jml,VARIABLE idCart
```

```
  ELSE
    VARIABLE dbCart SET FUNCTION tambahCart BASED ON VARIABLE getCart
  ENDIF
ELSE
  VARIABLE cart = "Silahkan login terlebih dahulu."
ENDIF

FUNCTION header BASED ON "location:index.php?page=ShoppingCart"
```

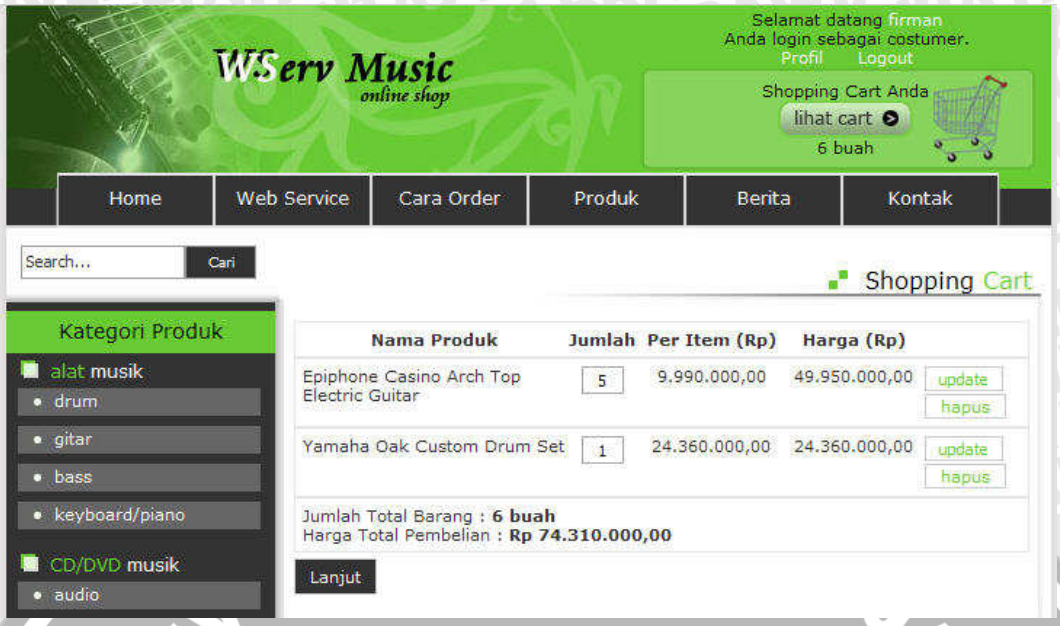
5.4.1.19. Implementasi Antarmuka Kebutuhan Fungsional Edit *Item Shopping Cart*

Fasilitas edit *item shopping cart* dapat diakses oleh *user* yang terdaftar pada Aplikasi *Web Service E-Commerce* di halaman *shopping cart*. Data yang dapat diedit oleh *user* berupa jumlah pemesanan barang. Gambar 5.41 menunjukkan implementasi antarmuka untuk edit *item shopping cart*.



Gambar 5. 41 : Implementasi Antarmuka untuk *Edit Item Shopping Cart*
Sumber : [Implementasi]

Setelah menekan tombol '`update`', sistem akan merubah jumlah pemesanan barang sesuai dengan jumlah yang diedit oleh *user* Gambar 5.42 menunjukkan hasil implementasi antarmuka untuk edit *item shopping cart*.



Gambar 5. 42 : Hasil Implementasi Antarmuka untuk *Edit Item Shopping Cart*
Sumber : [Implementasi]

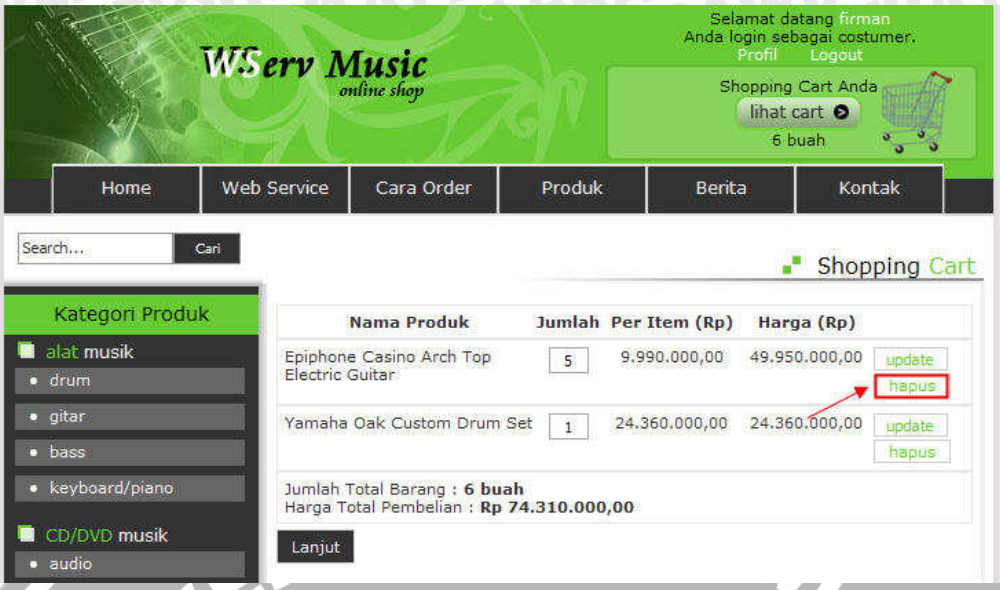
Implementasi algoritma untuk Edit *Item Shopping Cart* ditunjukkan dalam tabel berikut :

Algoritma 5. 19 : Algoritma Edit *Shopping Cart*

```
FUNCTION editCart

VARIABLE dbCart = new FUNCTION dbShoppingCart
IF FUNCTION isset BASED VARIABLE _POST['updateCart'] THEN
  VARIABLE jml = VARIABLE _POST['jmlPesan']
  IF not FUNCTION empty BASED ON VARIABLE _POST['jmlPesan'] and FUNCTION
is_numeric BASED ON VARIABLE _POST['jmlPesan'] and VARIABLE _POST['jmlPesan']!=0
THEN
    VARIABLE idCart = VARIABLE _GET['order']
    VARIABLE dbCart SET FUNCTION editCart BASED ON VARIABLE jml,VARIABLE idCart
    FUNCTION header BASED ON "location:index.php?page=ShoppingCart"
  ELSE
    FUNCTION header BASED ON "location:index.php?page=ShoppingCart"
  ENDIF
ENDIF
```

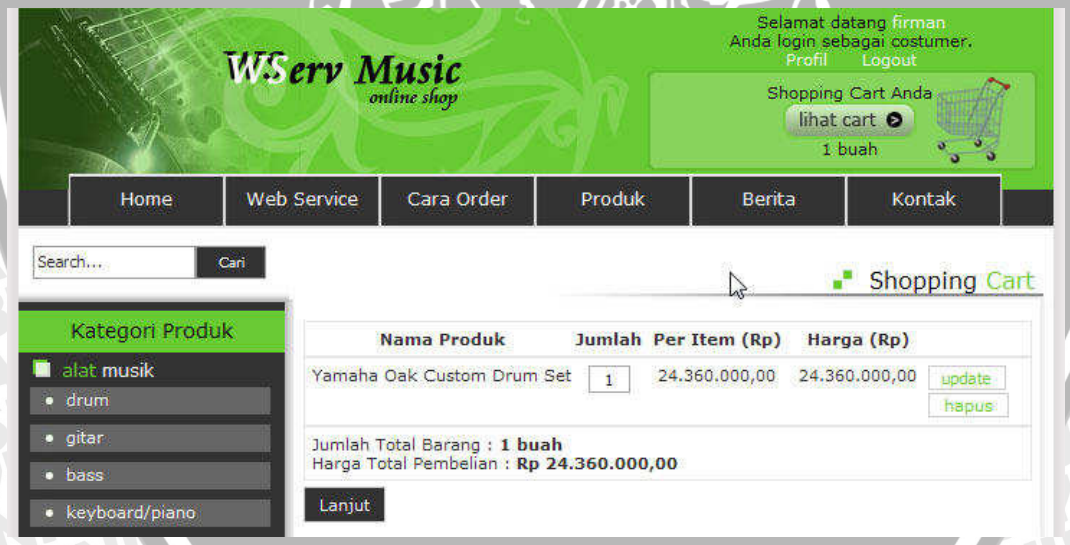
5.4.1.20. Implementasi Antarmuka Kebutuhan Fungsional Menghapus *Item Shopping Cart*

Fasilitas menghapus *item shopping cart* dapat diakses di halaman *shopping cart*. Gambar 5.43 menunjukkan implementasi antarmuka untuk menghapus *item shopping cart*.



Gambar 5. 43 : Implementasi Antarmuka untuk Menghapus *Item Shopping Cart*
Sumber : [Implementasi]

Setelah menekan tombol 'hapus', sistem akan menghapus salah satu data pemesanan barang *user*. Gambar 5.44 menunjukkan hasil implementasi antarmuka untuk menghapus *item shopping cart*.



Gambar 5. 44 : Hasil Implementasi Antarmuka untuk Menghapus *Item Shopping Cart*
Sumber : [Implementasi]

Implementasi algoritma untuk Menghapus *Item Shopping Cart* ditunjukkan dalam tabel berikut :

Algoritma 5. 20 : Algoritma Menghapus *Item Shopping Cart*

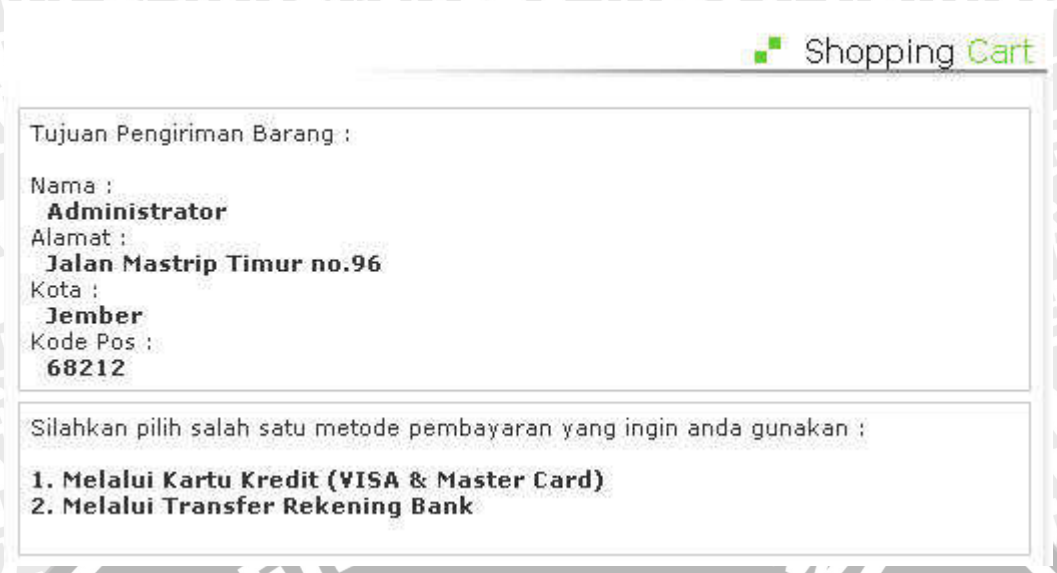
```
FUNCTION hapusCart

VARIABLE dbCart = new FUNCTION dbShoppingCart
IF FUNCTION isset BASED ON VARIABLE _SESSION['login'] and VARIABLE
_SESSION['login']=true THEN
  VARIABLE idCart = VARIABLE _GET['order']
  VARIABLE cart = VARIABLE this SET FUNCTION listCart
  VARIABLE dbCart SET FUNCTION hapusCart BASED ON VARIABLE idCart
ELSE
  VARIABLE cart = "Silahkan login terlebih dahulu."
ENDIF

FUNCTION header BASED ON "location:index.php?page=ShoppingCart"
```

5.4.1.21. Implementasi Antarmuka Kebutuhan Fungsional Memilih Pembayaran

Fasilitas memilih pembayaran dapat diakses setelah *user* melakukan proses penambahan *item shopping cart*. Gambar 5.45 menunjukkan implementasi antarmuka untuk memilih jasa kirim.

Gambar 5. 45 : Implementasi Antarmuka untuk Memilih Pembayaran
Sumber : [Implementasi]

Setelah *user* menekan tombol '`Lanjut`' pada halaman *shopping cart*, sistem menampilkan *form* untuk memilih pembayaran yang akan digunakan oleh *user*. Gambar 5.46 menunjukkan hasil implementasi antarmuka untuk memilih pembayaran.



Gambar 5. 46 : Hasil Implementasi Antarmuka untuk Memilih Jasa Kirim
Sumber : [Implementasi]

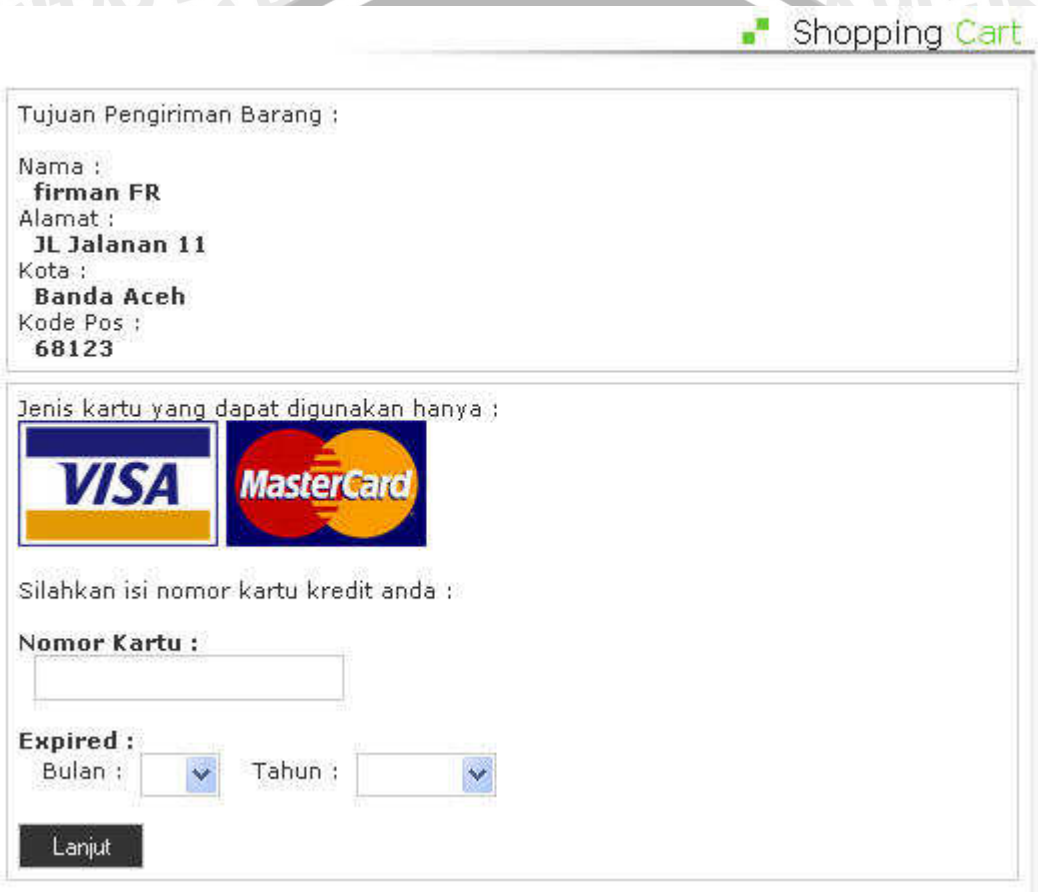
Implementasi algoritma untuk Memilih Pembayaran ditunjukkan dalam tabel berikut :

Algoritma 5. 21 : Algoritma Memilih Pembayaran

```
FUNCTION setPayment

VARIABLE htmlIndex=new FUNTION htmlLayout
VARIABLE getHtmlIndex=VARIABLE htmlIndex SET FUNCTION htmlNonIndex BASED ON
Shopping,Cart
VARIABLE kota=new FUNCTION dbKota
VARIABLE isiKota=VARIABLE kota SET FUNCTION listLokasi
VARIABLE idUser=VARIABLE _SESSION['idUser']
VARIABLE dbCart=new FUNTION dbShoppingCart
VARIABLE listCart=VARIABLE dbCart SET FUNCTION listCart BASED ON VARIABLE idUser,1

foreach VARIABLE listCart as VARIABLE lsCart
BEGIN
  VARIABLE namaUser=VARIABLE lsCart SET namaUser
  VARIABLE alamatUser=VARIABLE lsCart SET alamatUser
  VARIABLE kotaUser=VARIABLE lsCart SET kotaUser
  VARIABLE kodePos=VARIABLE lsCart SET kodePos
END

VARIABLE setHtml=VARIABLE getHtmlIndex[0]
VARIABLE setHtml.= VARIABLE namaUser
VARIABLE setHtml.= VARIABLE alamatUser
VARIABLE setHtml.= VARIABLE kotaUser
VARIABLE setHtml.= VARIABLE kodePos
VARIABLE setHtml.= VARIABLE getHtmlIndex[1]
VARIABLE data['isi']=VARIABLE setHtml

RETURN VARIABLE data
```

5.4.1.22. Implementasi Antarmuka Kebutuhan Fungsional Mengisi Data Kartu Kredit

Fasilitas mengisi data kartu kredit dapat diakses setelah *user* memilih metode pembayaran kartu kredit. Gambar 5.47 menunjukkan implementasi antarmuka untuk mengisi data kartu kredit.



Gambar 5. 47 : Implementasi Antarmuka untuk Mengisi Data Kartu Kredit
Sumber : [Implementasi]

Setelah *user* memilih pembayaran '`Melalui Kartu Kredit`' pada halaman *shopping cart*, sistem menampilkan *form* untuk mengisi data kartu kredit. Gambar 5.48 menunjukkan hasil implementasi antarmuka untuk mengisi data kartu kredit.

Gambar 5. 48 : Hasil Implementasi Antarmuka untuk Mengisi Data Kartu Kredit
Sumber : [Implementasi]

Implementasi algoritma untuk Mengisi Data Kartu Kredit ditunjukkan dalam tabel berikut :

Algoritma 5. 22 : Algoritma Mengisi Data Kartu Kredit

```
FUNCTION getPayment

VARIABLE cc=new Validate_Credit_Card
VARIABLE htmlIndex=new FUNCTION htmlLayout
VARIABLE getHtmlIndex=VARIABLE htmlIndex SET FUNCTION htmlNonIndex BASED ON
Shopping,Cart
VARIABLE dbCart=new FUNCTION dbShoppingCart
VARIABLE idUser=VARIABLE _SESSION['idUser']
VARIABLE listCart=VARIABLE dbCart SET listCart BASED ON VARIABLE idUser,1

Foreach VARIABLE listCart as VARIABLE lsCart
BEGIN
  VARIABLE namaUser=VARIABLE lsCart SET namaUser
  VARIABLE alamatUser=VARIABLE lsCart SET alamatUser
  VARIABLE kotaUser=VARIABLE lsCart SET kotaUser
  VARIABLE kodePos=VARIABLE lsCart SET kodePos
END

IF FUNCTION issetBASED ON VARIABLE _POST['skp_Submit'] THEN
  VARIABLE card_number=FUNCTION trim BASED ON VARIABLE _POST["cc_number"]
  VARIABLE expiration_month=FUNCTION trim BASED ON VARIABLE
_POST["expiration_month"]
  VARIABLE expiration_year=FUNCTION trim BASED ON VARIABLE
_POST["expiration_year"]
  VARIABLE ret=VARIABLE cc SET FUNCTION is_valid_number BASED ON VARIABLE
card_number
  VARIABLE ret2=VARIABLE cc SET FUNCTION is_valid_expiration BASED ON VARIABLE
expiration_month,VARIABLE expiration_year
```

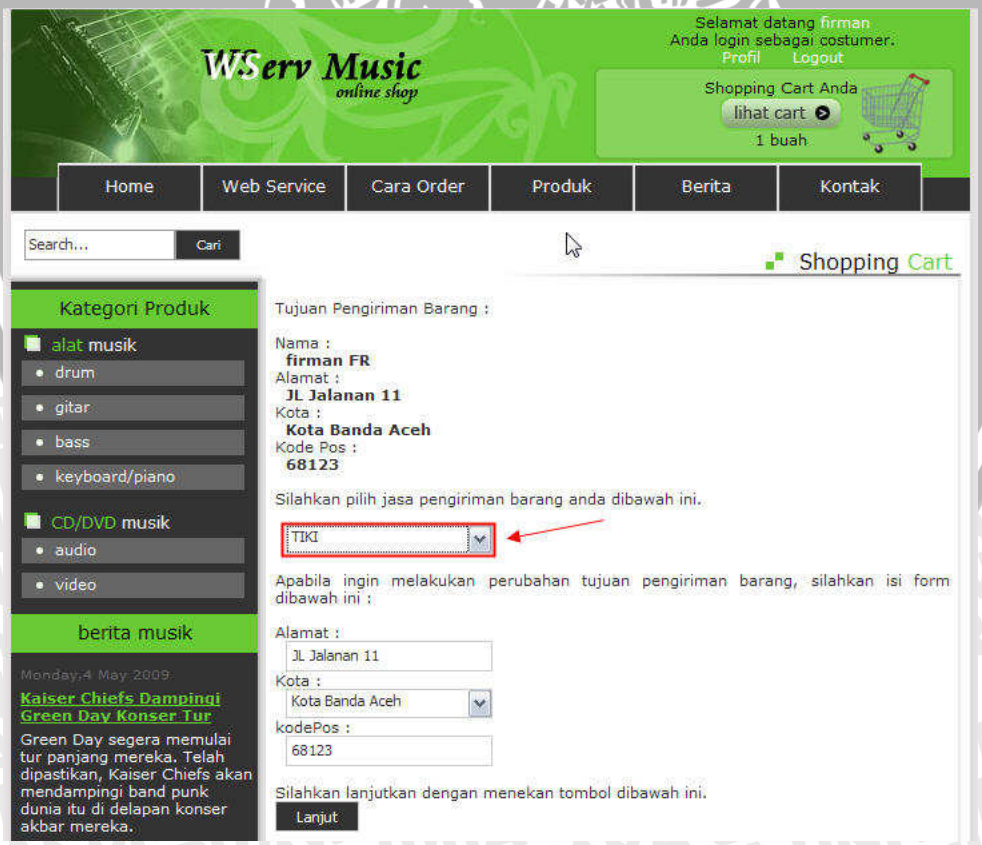
```
  IF FUNCTION empty BASED ON VARIABLE card_number OR not FUNCTION is_numeric BASED
ON VARIABLE card_number OR not VARIABLE ret=CC_SUCCESS THEN
    VARIABLE error['no']="Isi nomor kartu anda dengan benar."
  ELSE
    VARIABLE error['no']=""
  ENDIF

  IF FUNCTION empty BASED ON VARIABLE expiration_month OR FUNCTION empty BASED ON
VARIABLE expiration_year OR not VARIABLE ret2=CC_SUCCESS THEN
    VARIABLE error['exp']="Isi waktu expired kartu anda dengan benar."
  ELSE
    VARIABLE error['exp']=""
  ENDIF

  VARIABLE data=VARIABLE this SET FUNCTION formGetPayment BASED ON VARIABLE error

  IF FUNCTION empty BASED ON VARIABLE error['no'] AND FUNCTION empty BASED ON
VARIABLE error['exp'] THEN
    VARIABLE setHtml=VARIABLE getHtmlIndex[0]
    VARIABLE link="index.php?page=ShoppingCart&amp;go=setJasaKirim"
    VARIABLE setHtml.= VARIABLE namaUser
    VARIABLE setHtml.= VARIABLE alamatUser
    VARIABLE setHtml.= VARIABLE kotaUser
    VARIABLE setHtml.= VARIABLE kodePos
    VARIABLE setHtml."Jenis Kartu Kredit anda : ".VARIABLE cc SET FUNCTION
get_credit_card_name
    VARIABLE pot= FUNCTION substr_replace BASED ON VARIABLE card_number,'xxx',-3
    VARIABLE setHtml."Nomor Kartu Kredit anda :".VARIABLE pot
    VARIABLE bln=FUNCTION date BASED ON "F",FUNCTION strtotime BASED ON VARIABLE
expiration_month
    VARIABLE thn=FUNCTION date BASED ON "Y",FUNCTION strtotime BASED ON VARIABLE
expiration_month
    VARIABLE setHtml."Waktu Expired :".VARIABLE bln.VARIABLE expiration_year
    VARIABLE setHtml.= VARIABLE getHtmlIndex[1]
    VARIABLE data['isi']=VARIABLE setHtml
  ENDIF
ENDIF
RETURN VARIABLE data
```

5.4.1.23. Implementasi Antarmuka Kebutuhan Fungsional Memilih Jasa Kirim

Fasilitas memilih jasa kirim dapat diakses setelah *user* melakukan proses penambahan *item shopping cart*. Gambar 5.49 menunjukkan implementasi antarmuka untuk memilih jasa kirim.

Gambar 5. 49 : Implementasi Antarmuka untuk Memilih Jasa Kirim
Sumber : [Implementasi]

Setelah *user* menekan tombol '`Lanjut`' pada halaman *shopping cart*, sistem menampilkan *form* untuk memilih jasa kirim yang akan digunakan oleh *user*. Gambar 5.50 menunjukkan hasil implementasi antarmuka untuk memilih jasa kirim.



Gambar 5. 50 : Hasil Implementasi Antarmuka untuk Memilih Jasa Kirim
Sumber : [Implementasi]

Implementasi algoritma untuk Memilih jasa kirim ditunjukkan dalam tabel berikut :

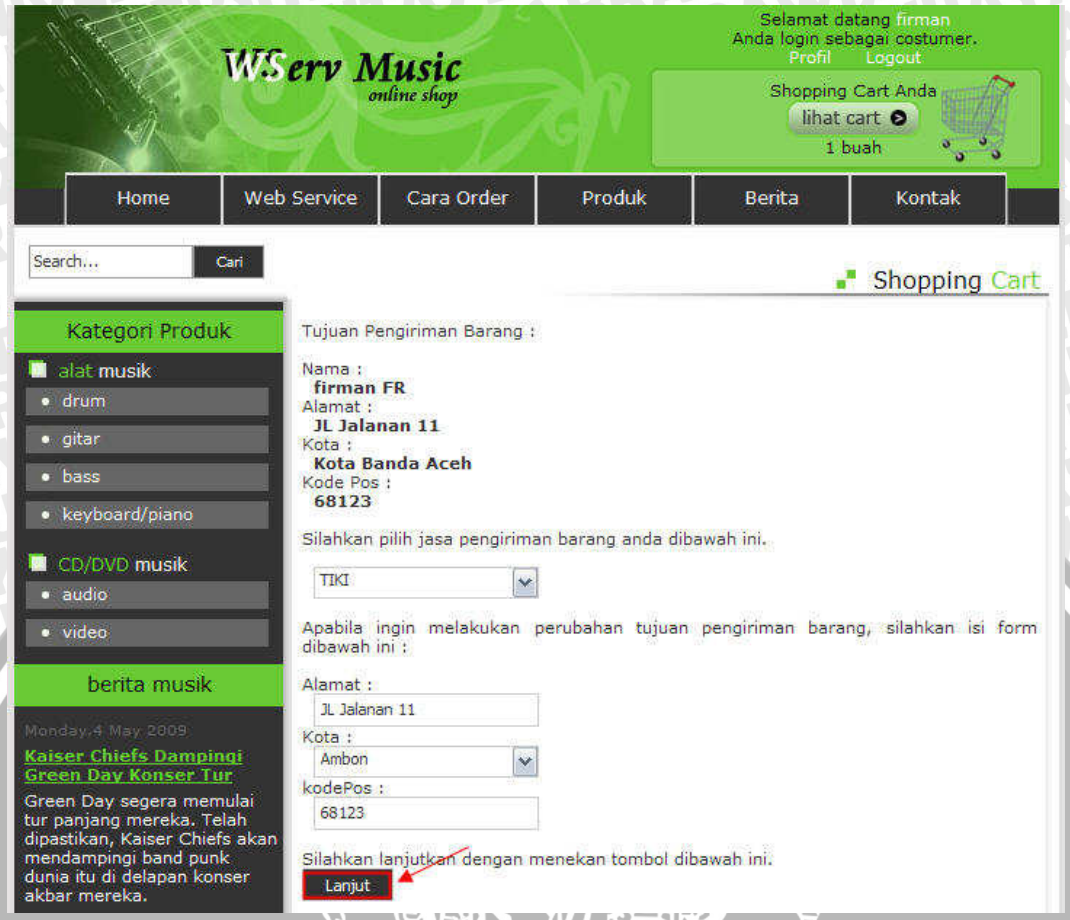
Algoritma 5. 23 : Algoritma Memilih Jasa Kirim

```
FUNCTION setJasaKirim

VARIABLE htmlIndex = new FUNCTION htmlLayout
VARIABLE getHtmlIndex = VARIABLE htmlIndex SET FUNCTION htmlNonIndex BASED ON
'Shopping','Cart'
VARIABLE dbCart = new FUNCTION dbShoppingCart
VARIABLE idUser = VARIABLE _SESSION['idUser']
VARIABLE listCart = VARIABLE dbCart SET FUNCTION listCart BASED ON VARIABLE
idUser,1

foreach VARIABLE listCart as VARIABLE lsCart
BEGIN
VARIABLE dataBarang[] = VARIABLE lsCart SET namaBarang."(@Rp ".FUNCTION
number_format BASED ON VARIABLE lsCart SET hargaBarang,2,',','.'
VARIABLE dataBarang[].= VARIABLE lsCart SET jmlPesan
VARIABLE totHarga[] = VARIABLE lsCart SET jmlPesan*VARIABLE lsCart SET hargaBarang
VARIABLE namaUser = VARIABLE lsCart SET namaUser
VARIABLE alamatUser = VARIABLE lsCart SET alamatUser
VARIABLE emailUser = VARIABLE lsCart SET emailUser
VARIABLE kotaUser = VARIABLE lsCart SET kotaUser
VARIABLE kodePosUser = VARIABLE lsCart SET kodePos
END

IF FUNCTION isset BASED ON VARIABLE _POST['next'] THEN
  IF FUNCTION empty BASED ON VARIABLE _POST['jasaKirim'] THEN
    VARIABLE error = "Pilih salah satu jasa kirim."
    VARIABLE data = VARIABLE this SET FUNCTION getJasaKirim BASED ON VARIABLE
error
  ELSE

  VARIABLE dbDataKirim = new FUNCTION dbDataKirim
  VARIABLE detJasaKirim = VARIABLE dbDataKirim SET FUNCTION detJasaKirim BASED ON
VARIABLE _POST['jasaKirim']
  VARIABLE det = VARIABLE dbDataKirim SET detDataKirim BASED ON VARIABLE
detJasaKirim SET idJasaKirim,VARIABLE _POST['kotaUser']
  VARIABLE setHtml=VARIABLE getHtmlIndex[0]
  VARIABLE setHtml.="Nama : ".VARIABLE namaUser
  VARIABLE setHtml.="Alamat :"
  IF FUNCTION empty BASED ON VARIABLE _POST['alamatUser'] THEN
    VARIABLE alamat = VARIABLE alamatUser
  ELSE
    VARIABLE alamat = VARIABLE _POST['alamatUser']
  ENDIF
  VARIABLE setHtml.= VARIABLE alamat
  VARIABLE setHtml.= "Kota :".VARIABLE _POST['kotaUser']
  VARIABLE setHtml.= "Kode Pos :"
  IF FUNCTION empty BASED ON VARIABLE _POST['kodePos'] or not FUNCTION is_numeric
BASED ON VARIABLE _POST['kodePos']THEN
    VARIABLE kodePos = VARIABLE kodePosUser
  ELSE
    VARIABLE kodePos = VARIABLE _POST['kodePos']
  ENDIF
  VARIABLE setHtml.= VARIABLE kodePos
  VARIABLE setHtml.= "Informasi Barang Pesanan Anda :"
  VARIABLE setHtml.= "Nama Barang :"
```

```
  foreach VARIABLE dataBarang as VARIABLE dBarang
  BEGIN
    VARIABLE setHtml.= VARIABLE dBarang
  END
  VARIABLE totalHarga = FUNCTION array_sum BASED ON VARIABLE totHarga
  VARIABLE setHtml.= "Rp ".FUNCTION number_format BASED ON VARIABLE
totalHarga,2,',','.'
  VARIABLE setHtml.= "Jasa Kirim :".VARIABLE _POST['jasaKirim']
  VARIABLE setHtml.= "Biaya Pengiriman : Rp ".FUNCTION number_format BASED ON
VARIABLE det SET hargaKirim,2,',','.'
  VARIABLE totalPembayaran = VARIABLE totalHarga+VARIABLE det SET hargaKirim
  VARIABLE setHtml.= "Jumlah Total Biaya : Rp ".FUNCTION number_format BASED ON
VARIABLE totalPembayaran,2,',','.'
  VARIABLE setHtml.= VARIABLE getHtmlIndex[1]
  FUNCTION setcookie BASED ON "namaUser",VARIABLE namaUser
  FUNCTION setcookie BASED ON "barang",VARIABLE dBarang
  FUNCTION setcookie BASED ON "email",VARIABLE emailUser
  FUNCTION setcookie BASED ON "idJasaKirim",VARIABLE detJasaKirim SET idJasaKirim
  FUNCTION setcookie BASED ON "jasaKirim",VARIABLE _POST['jasaKirim']
  FUNCTION setcookie BASED ON "biayaKirim",VARIABLE det SET hargaKirim
  FUNCTION setcookie BASED ON "totalBiaya",VARIABLE totalPembayaran
  FUNCTION setcookie BASED ON "alamat",VARIABLE alamat
  FUNCTION setcookie BASED ON "kota",VARIABLE _POST['kotaUser']
  FUNCTION setcookie BASED ON "kodePos",VARIABLE kodePos
  FUNCTION setcookie BASED ON "note",VARIABLE note
  VARIABLE data['isi'] = VARIABLE setHtml
ELSE
  VARIABLE data = ''
ENDIF

return VARIABLE data
```

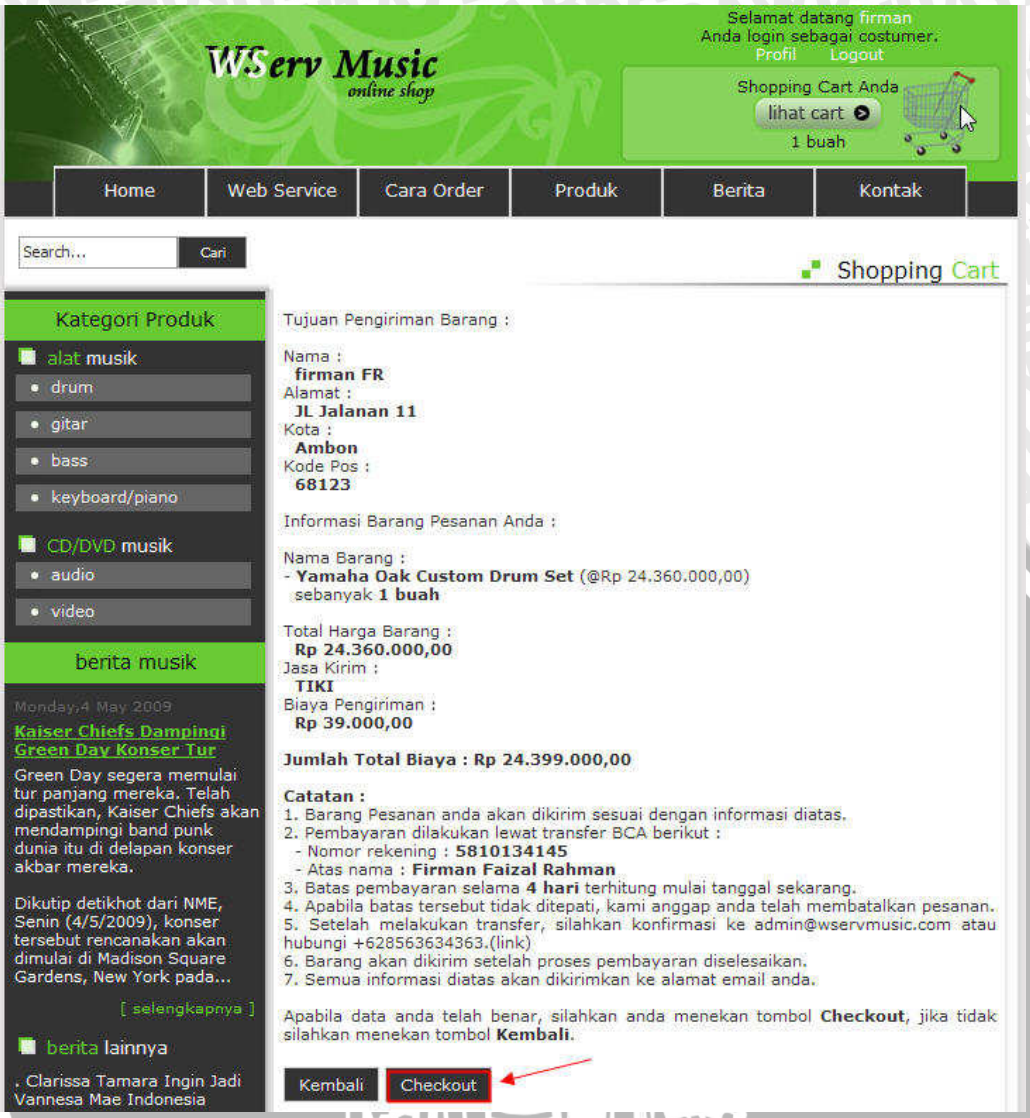
5.4.1.24. Implementasi Antarmuka Kebutuhan Fungsional *Checkout Shopping Cart*

Fasilitas *checkout shopping cart* dapat diakses setelah melakukan proses memilih jasa kirim. Gambar 5.51 menunjukkan implementasi antarmuka untuk menambah barang.



Gambar 5. 51 : Implementasi Antarmuka untuk *Checkout Shopping Cart*
Sumber : [Implementasi]

Setelah *user* menekan tombol '`Lanjut`' setelah *user* memilih jasa kirim, sistem menampilkan data informasi produk yang dipesan oleh *user*. Untuk melakukan *checkout*, *user* menekan tombol '`checkout`'. Gambar 5.47 menunjukkan hasil implementasi antarmuka untuk *checkout shopping cart*.



Gambar 5. 52 : Hasil Implementasi Antarmuka untuk *Checkout Shopping Cart*
Sumber : [Implementasi]

Implementasi algoritma untuk *Checkout Shopping Cart* ditunjukkan dalam tabel berikut :

Algoritma 5. 24 : Algoritma *Checkout Shopping Cart*

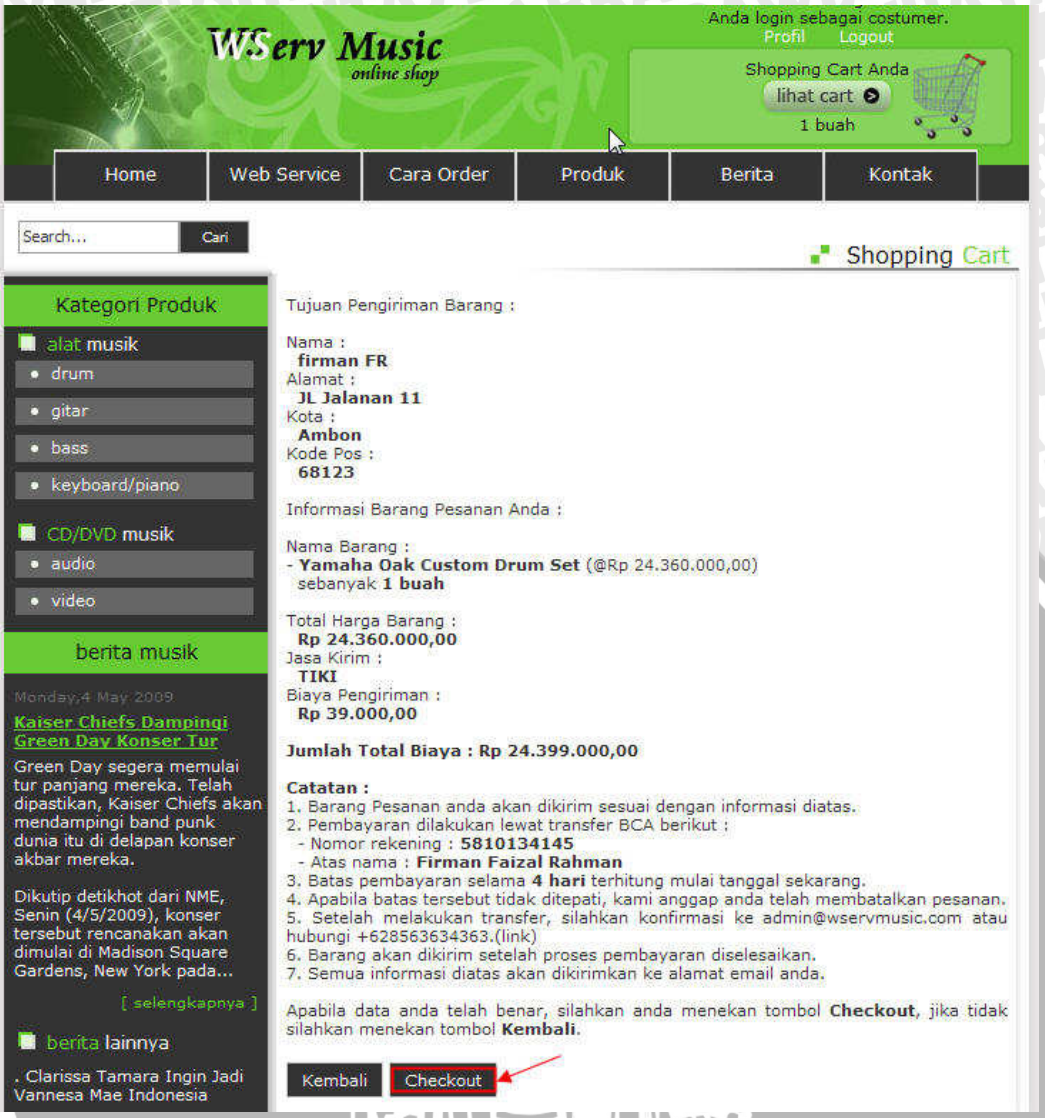
```
FUNCTION checkout

VARIABLE htmlIndex = new FUNCTION  htmlLayout
VARIABLE getHtmlIndex = VARIABLE htmlIndex SET FUNCTION htmlNonIndex BASED ON
'Shopping','Cart'
VARIABLE hari = FUNCTION date BASED ON "j"
VARIABLE bln = FUNCTION date BASED ON "n"
VARIABLE thn = FUNCTION date BASED ON "Y"
VARIABLE tglExpired = FUNCTION date BASED ON "Y-m-d",FUNCTION mktime BASED ON
0,0,0,VARIABLE bln,VARIABLE hari+4,VARIABLE thn
VARIABLE dbOrderUser = new FUNCTION  dbOrderUser
VARIABLE cart['idUser'] = VARIABLE _SESSION['idUser']
VARIABLE cart['idJasaKirim'] = VARIABLE _COOKIE['idJasaKirim']
```

```
VARIABLE cart['totalBiaya'] = VARIABLE _COOKIE['totalBiaya']
VARIABLE cart['alamat'] = VARIABLE _COOKIE['alamat']
VARIABLE cart['kota'] = VARIABLE _COOKIE['kota']
VARIABLE cart['kodePos'] = VARIABLE _COOKIE['kodePos']
VARIABLE cart['sessionId'] = FUNCTION session_id
VARIABLE cart['tglPesan'] = FUNCTION date BASED ON "Y-m-d"
VARIABLE cart['tglExpired'] = VARIABLE tglExpired
VARIABLE dbOrderUser SET FUNCTION tambahOrderUser BASED ON VARIABLE cart
VARIABLE detOrderUser = VARIABLE dbOrderUser SET FUNCTION detOrderUser BASED ON
VARIABLE cart['idUser'],VARIABLE cart['sessionId']
VARIABLE dbCart = new FUNCTION dbShoppingCart
VARIABLE dbCart SET FUNCTION checkout BASED ON VARIABLE detOrderUser SET
idOrder,VARIABLE cart['idUser'],VARIABLE cart['sessionId']
VARIABLE this SET FUNCTION kirimEmail
VARIABLE setHtml = VARIABLE getHtmlIndex[0]
VARIABLE setHtml.="Terima kasih telah berbelanja di situs kami."
VARIABLE setHtml.= VARIABLE getHtmlIndex[1]
VARIABLE dataCart['isi'] = VARIABLE setHtml
FUNCTION session_regenerate_id

RETURN VARIABLE dataCart
```

5.4.1.25. Implementasi Antarmuka Kebutuhan Fungsional Konfirmasi *Email*

Konfirmasi *email* merupakan proses dimana *user* telah melakukan pemesanan produk dan memilih jasa kirim. Fasilitas ini merupakan konfirmasi dari *administrator* Aplikasi *Web Service E-Commerce* untuk mengirimkan informasi data pesanan produk *customer* melalui *email*.

Proses konfirmasi *email* dilakukan bersamaan dengan *user* melakukan *checkout shopping cart*. Konfirmasi *email* diproses saat *user* menekan tombol 'checkout' pada halaman *shopping cart*. Gambar 5.48 menunjukkan implementasi antarmuka untuk konfirmasi *email*.



Gambar 5. 53 : Implementasi Antarmuka untuk Konfirmasi *Email*
Sumber : [Implementasi]

Setelah *user* menekan tombol '`checkout`', sistem menampilkan pesan pada halaman *website*. Gambar 5.49 menunjukkan hasil implementasi antarmuka untuk konfirmasi *email*.

Gambar 5. 54 : Hasil Implementasi Antarmuka untuk Konfirmasi *Email*
Sumber : [Implementasi]

Implementasi algoritma untuk Konfirmasi *Email* ditunjukkan dalam tabel berikut :

Algoritma 5. 25 : Algoritma Konfirmasi *Email*

```
FUNCTION kirimEmail

VARIABLE kepada = VARIABLE _COOKIE['email']
VARIABLE judul = "Informasi Pemesanan Barang"
VARIABLE isi = "Atas nama : ".VARIABLE _COOKIE['namaUser']."\n"
VARIABLE isi.= "Alamat : ".VARIABLE _COOKIE['alamat']."\n"
VARIABLE isi.= "Kota : ".VARIABLE _COOKIE['kota']."\n"
VARIABLE isi.= "Kode Pos : ".VARIABLE _COOKIE['kodePos']."\n"
VARIABLE isi.= "Barang Pesanan : ".VARIABLE _COOKIE['barang']."\n"
VARIABLE isi.= "Jasa Kirim : ".VARIABLE _COOKIE['jasaKirim']."\n"
VARIABLE isi.= "Biaya Kirim : Rp ".FUNCTION number_format BASED ON VARIABLE
_COOKIE['biayaKirim'],2,',','.'."\n"
VARIABLE isi.= "Total Biaya : Rp ".FUNCTION number_format BASED ON VARIABLE
_COOKIE['totalBiaya'],2,',','.'."\n\n"
VARIABLE isi.= VARIABLE _COOKIE['note']
VARIABLE msg = FUNCTION strip_tags BASED ON VARIABLE isi
IF FUNCTION mail BASED ON VARIABLE kepada,VARIABLE judul,VARIABLE msg THEN
  RETURN true
ELSE
  RETURN false
ENDIF
```

5.4.1.26. Implementasi Antarmuka Kebutuhan Fungsional Melihat Daftar *Order* Barang

Fasilitas melihat daftar *order* barang hanya dapat diakses oleh *administrator*. Fasilitas ini dapat diakses melalui halaman administrasi *website*. *Administrator* harus memilih menu ‘order’ untuk menampilkan daftar *order* barang. Gambar 5.50 menunjukkan implementasi antarmuka untuk melihat daftar *order* barang.

Gambar 5. 55 : Implementasi Antarmuka untuk Melihat *Order* Barang
Sumber : [Implementasi]

Setelah memilih menu 'order', sistem menampilkan daftar pemesan produk pada Aplikasi *Web Service E-Commerce*. Gambar 5.51 menunjukkan hasil implementasi antarmuka untuk melihat *order* barang.

Gambar 5. 56 : Hasil Implementasi Antarmuka untuk Melihat *Order* Barang
Sumber : [Implementasi]

Implementasi algoritma untuk Melihat *Order* Barang ditunjukkan dalam tabel berikut :

Algoritma 5. 26 : Algoritma Melihat *Order* Barang

```
FUNCTION listOrder

VARIABLE dbOrderUser = new FUNCTION dbOrderUser
VARIABLE paging = new FUNCTION paging
IF FUNCTION isset BASED ON VARIABLE _GET['admin'] THEN
  VARIABLE limit = 3
  VARIABLE start = VARIABLE paging SET FUNCTION getStart BASED ON VARIABLE limit
  VARIABLE numOrder = VARIABLE dbOrderUser SET FUNCTION numOrder
  IF VARIABLE numOrder=0 THEN
    VARIABLE setHtml="Order Produk Masih Kosong"
  ELSE
  VARIABLE orderUser = VARIABLE dbOrderUser SET FUNCTION listOrder BASED ON
VARIABLE start,VARIABLE limit
  foreach VARIABLE orderUser as VARIABLE lsOrderUser
  BEGIN
    VARIABLE idOrder = VARIABLE lsOrderUser SET idOrder
    VARIABLE detCart = VARIABLE dbOrderUser SET FUNCTION detCart BASED ON VARIABLE
idOrder
    foreach VARIABLE detCart as VARIABLE lsOrderCart
    VARIABLE idStatusBayar = VARIABLE lsOrderCart SET idStatusBayar
    VARIABLE statusBayar = VARIABLE lsOrderCart SET statusBayar
    END
  VARIABLE tglPesan = FUNCTION date BASED ON "j F Y",FUNCTION strtotime BASED ON
VARIABLE lsOrderUser SET tglPesan
  IF VARIABLE idStatusBayar=2 THEN
    VARIABLE style = "color:#DB9124"
  ELSE
    VARIABLE style = "color:#66cc33;”
  ENDIF
  IF VARIABLE idStatusBayar = 4 THEN VARIABLE style = "color:red"
  VARIABLE ls[].="Status pembayaran : ".VARIABLE style. FUNCTION ucwords BASED ON
VARIABLE statusBayar)
  VARIABLE ls[]= "Tanggal pemesanan : VARIABLE tglPesan
  VARIABLE ls[].= "Nama : ” VARIABLE lsOrderUser SET namaUser
  VARIABLE ls[].= "Total pembelian : ".FUNCTION number_format BASED ON VARIABLE
lsOrderUser SET totalBiaya,2,',','.'
  END

  VARIABLE query = "index.php?admin=cpanel&amp;page=OrderUser&amp;"
  IF VARIABLE numOrder>VARIABLE limit THEN
    VARIABLE setHtml.= VARIABLE paging SET setPaging BASED ON VARIABLE
numOrder,VARIABLE limit,VARIABLE query
  ELSE
    VARIABLE setHtml.=""
  ENDIF

  foreach VARIABLE ls as VARIABLE getDataOrderUser
  BEGIN
    VARIABLE setHtml = VARIABLE getDataOrderUser
  END
ELSE
  VARIABLE setHtml = ''
  FUNCTION header BASED ON "location:index.php"
ENDIF

RETURN VARIABLE setHtml
```

5.4.1.27. Implementasi Antarmuka Kebutuhan Fungsional Melihat Detail *Order* Barang

Fasilitas melihat detail *order* barang hanya dapat diakses oleh *administrator*. Fasilitas ini dapat diakses melalui halaman administrasi *website*. *Administrator* menekan tombol '`detail`' untuk informasi detail pemesanan produk. Gambar 5.52 menunjukkan implementasi antarmuka untuk melihat detail *order* barang.

Gambar 5. 57 : Implementasi Antarmuka untuk Melihat Detail *Order* Barang
Sumber : [Implementasi]

Setelah menekan tombol '`detail`', sistem menampilkan informasi detail pemesanan produk. Gambar 5.53 menunjukkan hasil implementasi antarmuka untuk menambah barang.

Gambar 5. 58 : Hasil Implementasi Antarmuka untuk Melihat Detail *Order* Barang
Sumber : [Implementasi]

Implementasi algoritma untuk Melihat Detail *Order* Barang ditunjukkan dalam tabel berikut :

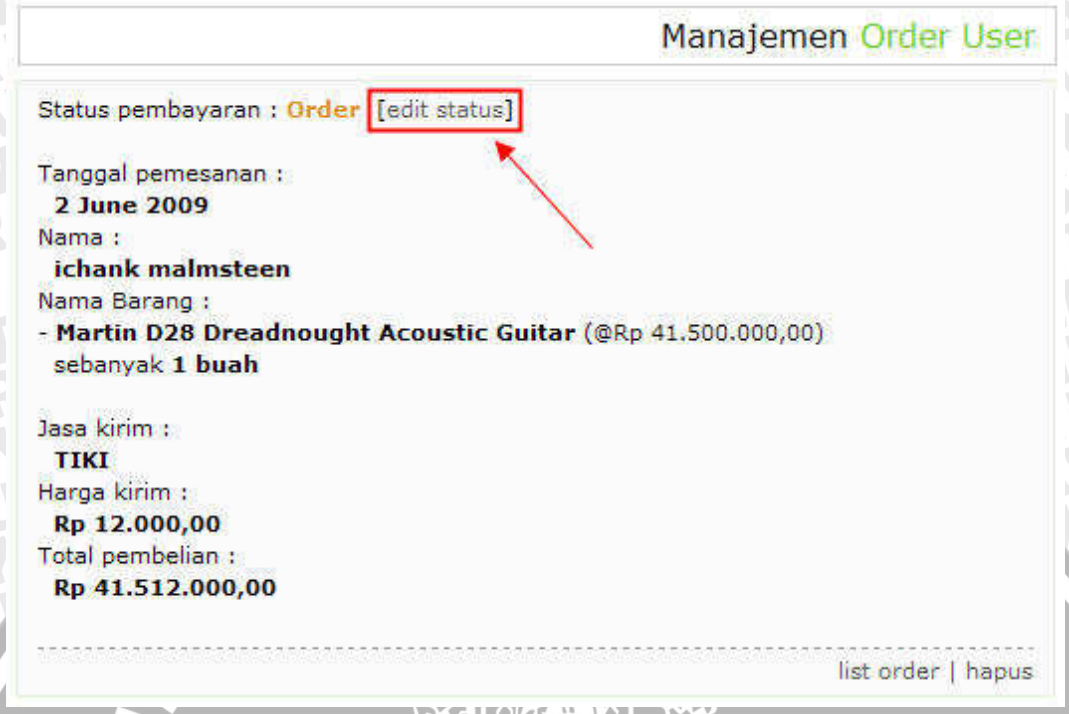
Algoritma 5. 27 : Algoritma Melihat Detail *Order* Barang

```
FUNCTION detOrder

VARIABLE dbOrderUser = new FUNCTION dbOrderUser
VARIABLE idOrder = VARIABLE _GET['Id']
VARIABLE detOrder = VARIABLE dbOrderUser SET FUNCTION detOrder BASED ON VARIABLE
idOrder
VARIABLE detCart = VARIABLE dbOrderUser SET FUNCTION detCart BASED ON VARIABLE
idOrder
VARIABLE tglPesan = FUNCTION date BASED ON "j F Y",FUNCTION strtotime BASED ON
VARIABLE detOrder SET tglPesan

foreach VARIABLE detCart as VARIABLE lsOrderCart
BEGIN
VARIABLE lsCart[]= VARIABLE lsOrderCart SET namaBarang."(@Rp ".FUNCTION
number_format BASED ON VARIABLE lsOrderCart SET hargaBarang,2,',','.')"
VARIABLE lsCart[].= "sebanyak ".VARIABLE lsOrderCart SET jmlPesan." buah"
VARIABLE idStatusBayar = VARIABLE lsOrderCart SET idStatusBayar
VARIABLE statusBayar = VARIABLE lsOrderCart SET statusBayar
END

IF VARIABLE idStatusBayar=2 THEN
  VARIABLE style = "color:#DB9124"
ELSE
  VARIABLE style = "color:#66cc33"
  VARIABLE link = ""
ENDIF
```

```
IF VARIABLE idStatusBayar=4 THEN
  VARIABLE style = "color:red";
  VARIABLE link = ""
ENDIF

VARIABLE ls = "Status pembayaran : ".FUNCTION ucwords BASED ON VARIABLE
statusBayar
VARIABLE ls.= VARIABLE link
VARIABLE ls.= "Tanggal pemesanan : ".VARIABLE tglPesan
VARIABLE ls.= "Nama : ".VARIABLE detOrder SET namaUser
VARIABLE ls.= "Nama Barang :"

foreach VARIABLE lsCart as VARIABLE cart
BEGIN
  VARIABLE ls.= VARIABLE cart
  VARIABLE ls.= "Jasa kirim : ".VARIABLE detOrder SET jasaKirim
  VARIABLE ls.= "Harga kirim : Rp ".FUNCTION number_format BASED ON VARIABLE
detOrder SET hargaKirim,2,',','.'
  VARIABLE ls.= "Total pembelian : Rp ".FUNCTION number_format BASED ON VARIABLE
detOrder SET totalBiaya,2,',','.'
END
VARIABLE setHtml.= VARIABLE ls

IF FUNCTION isset BASED ON VARIABLE _SESSION['login']) and VARIABLE
_SESSION['login'] = true THEN
  VARIABLE order['isi'] = VARIABLE setHtml
ELSE
  VARIABLE order = ''
  FUNCTION header BASED ON "location:index.php"
ENDIF

RETURN VARIABLE order
```

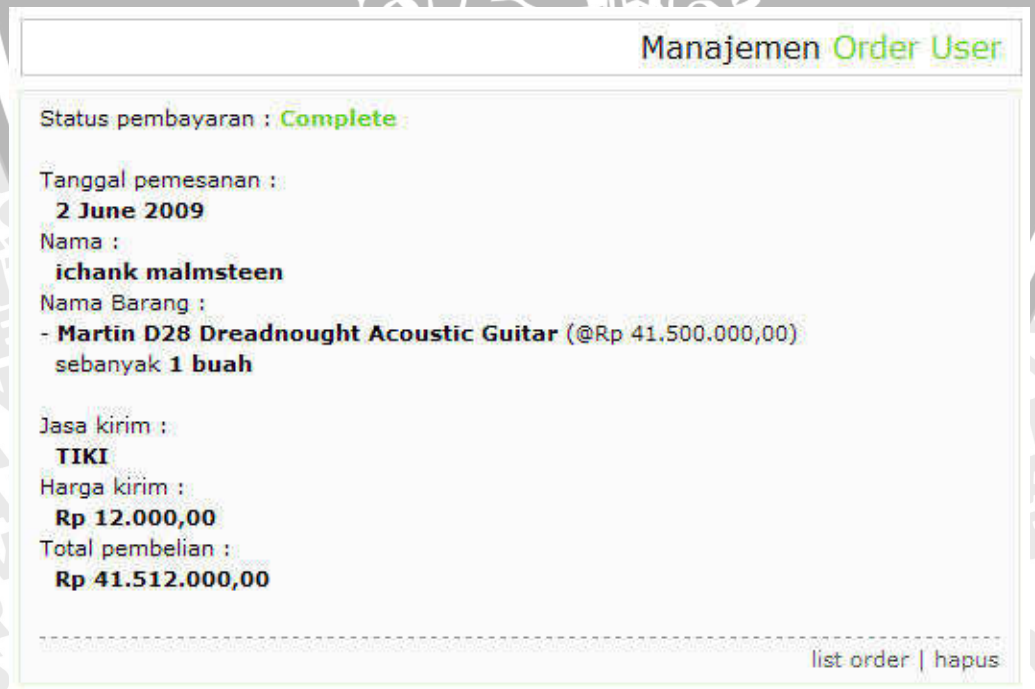
5.4.1.28. Implementasi Antarmuka Kebutuhan Fungsional Mengedit Status *Order* Barang

Fasilitas mengedit status *order* barang hanya dapat diakses oleh *administrator*. Fasilitas ini dapat diakses pada halaman detail *order*. *Administrator* menekan tombol '`edit status`' untuk merubah status *order*. Status *order* barang terdiri dari *complete*, *order*, dan *expired*. Fasilitas ini dapat digunakan hanya untuk *order* barang yang memiliki status *order*. Gambar 5.54 menunjukkan implementasi antarmuka untuk mengedit status *order* barang.



Gambar 5. 59 : Implementasi Antarmuka untuk Mengedit Status *Order* Barang
Sumber : [Implementasi]

Setelah *administrator* menekan tombol '`edit status`', sistem merubah status barang dari *order* menjadi *complete*. Gambar 5.55 menunjukkan hasil implementasi antarmuka untuk mengedit status *order* barang.



Gambar 5. 60 : Hasil Implementasi Antarmuka untuk Mengedit Status *Order* Barang
Sumber : [Implementasi]

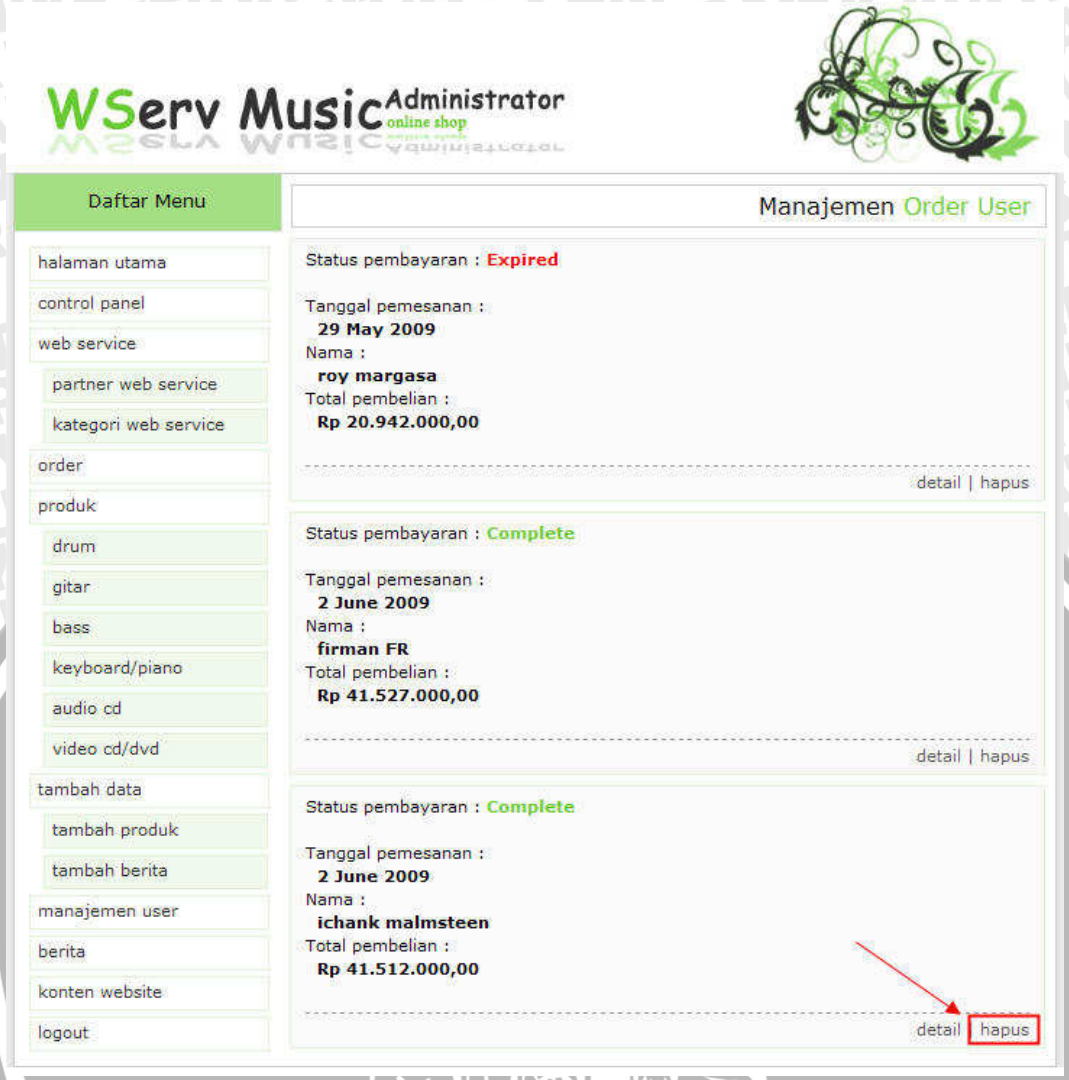
Implementasi algoritma untuk Mengedit Status *Order* Barang ditunjukkan dalam tabel berikut :

Algoritma 5. 28 : Algoritma Mengedit Status *Order* Barang

```
FUNCTION editOrderStatus

IF FUNCTION isset BASED VARIABLE _SESSION['login']) and VARIABLE _SESSION['login']
= true THEN
  VARIABLE idOrder = VARIABLE _GET['Id']
  VARIABLE dbOrderUser = new FUNCTION dbOrderUser
  VARIABLE dbOrderUser SET FUNCTION editOrderStatus BASED ON VARIABLE idOrder
ENDIF

FUNCTION header BASED ON
"location:index.php?admin=cpanel&page=OrderUser&go=detOrder&Id=".VARIABLE idOrder
```

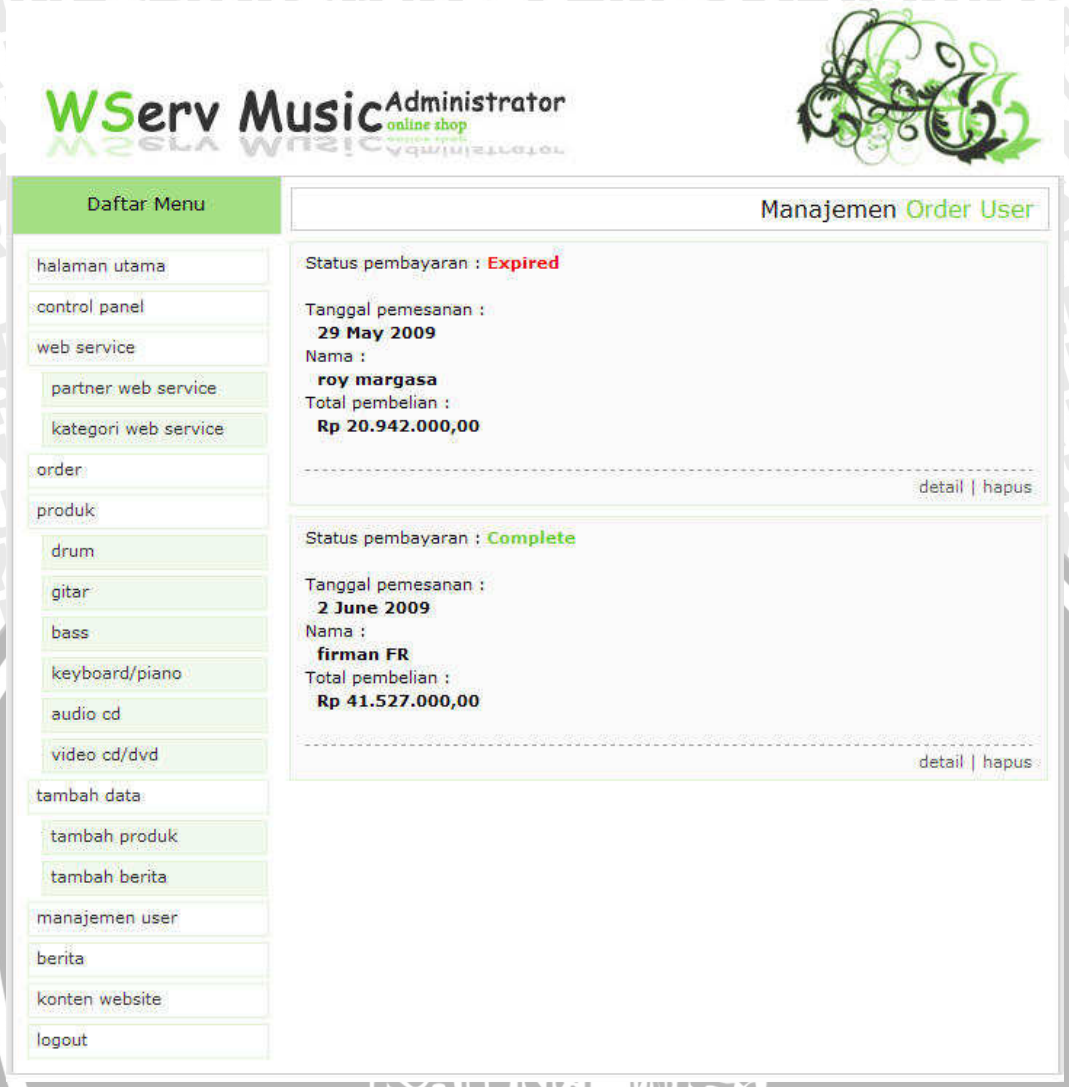
5.4.1.29. Implementasi Antarmuka Kebutuhan Fungsional Menghapus *Order* Barang

Fasilitas menghapus *order* barang hanya dapat diakses oleh *administrator*. Fasilitas ini dapat diakses melalui halaman administrasi *website*. *Administrator* menekan tombol ‘hapus’ untuk menghapus data *order* barang. Gambar 5.56 menunjukkan implementasi antarmuka untuk menghapus *order* barang.



Gambar 5. 61 : Implementasi Antarmuka untuk Menghapus *Order* Barang
Sumber : [Implementasi]

Setelah *administrator* menekan tombol 'hapus', sistem menghapus data *order* barang dari sistem. Gambar 5.57 menunjukkan hasil implementasi antarmuka untuk menghapus *order* barang.



Gambar 5. 62 : Hasil Implementasi Antarmuka untuk Menghapus *Order* Barang
Sumber : [Implementasi]

Implementasi algoritma untuk Menghapus *Order* Barang ditunjukkan dalam tabel berikut :

Algoritma 5. 29 : Algoritma Menghapus *Order* Barang

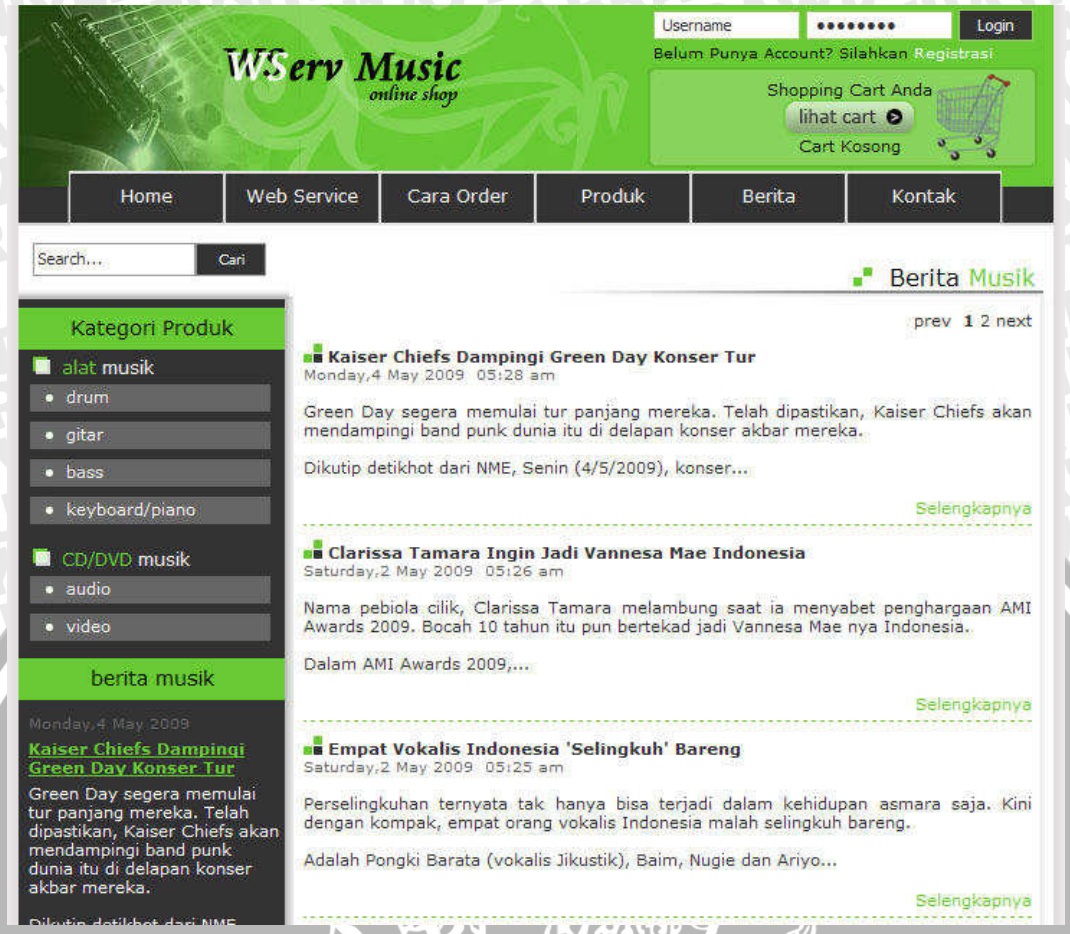
```
FUNCTION hapusOrder

IF FUNCTION isset BASED ON VARIABLE _SESSION['login'] and VARIABLE
_SESSION['login'] = true THEN
  VARIABLE idOrder = VARIABLE _GET['Id']
  VARIABLE dbOrderUser = new FUNCTION dbOrderUser
  VARIABLE dbOrderUser SET FUNCTION hapusOrder BASED ON VARIABLE idOrder
ENDIF

FUNCTION header BASED ON "location:index.php?admin=cpanel&page=OrderUser"
```

5.4.1.30. Implementasi Antarmuka Kebutuhan Fungsional Melihat Daftar Berita

Setiap *user* yang mengakses Aplikasi *Web Service E*-Commerce dapat melihat daftar Berita. Gambar 5.57 menunjukkan implementasi antarmuka untuk melihat daftar berita.

Gambar 5. 63 : Implementasi Antarmuka untuk Melihat Daftar Berita
Sumber : [Implementasi]

Halaman daftar berita dapat diakses dengan memilih menu 'berita' pada halaman utama *website*. Gambar 5.58 menunjukkan hasil implementasi antarmuka untuk melihat daftar berita.



Gambar 5. 64 : Hasil Implementasi Antarmuka untuk Melihat Daftar Berita
Sumber : [Implementasi]

Implementasi algoritma untuk Melihat Daftar Berita ditunjukkan dalam tabel berikut :

Algoritma 5. 30 : Algoritma Melihat Daftar Berita

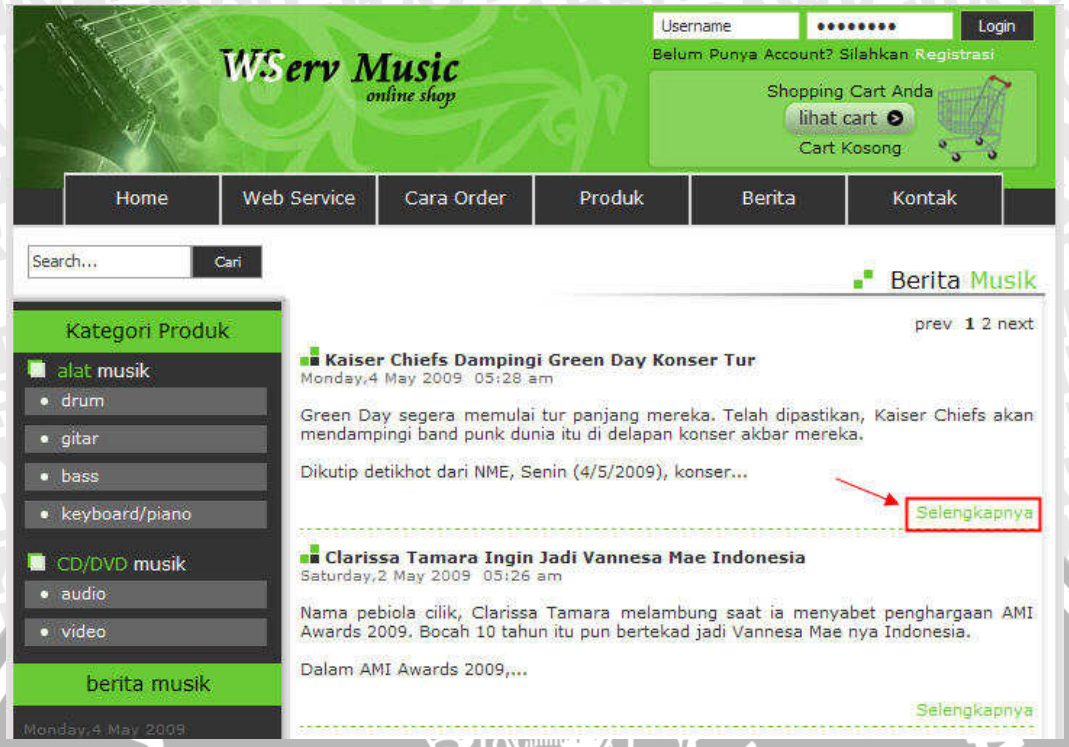
```
FUNCTION listBerita

VARIABLE cutTxt = new FUNCTION textCutting
VARIABLE htmlIndex = new FUNCTION htmlLayout
VARIABLE getHtmlIndex = VARIABLE htmlIndex SET FUNCTION htmlNonIndex BASED ON
'Berita','Musik'
VARIABLE paging = new FUNCTION paging
VARIABLE dataBerita = new FUNCTION dbBerita

IF FUNCTION isset BASED ON VARIABLE _GET['admin'] and FUNCTION isset BASED ON
VARIABLE _GET['page'] and VARIABLE _GET['page']="Berita" THEN
  VARIABLE limit = 5
  VARIABLE start = VARIABLE paging SET FUNCTION getStart BASED ON VARIABLE limit
  VARIABLE isiBerita = VARIABLE dataBerita SET FUNCTION listBerita BASED ON
VARIABLE start,VARIABLE limit
  VARIABLE numBerita = VARIABLE dataBerita SET FUNCTION numBerita
  foreach VARIABLE isiBerita as VARIABLE lsBerita
  BEGIN
  VARIABLE idBerita = VARIABLE lsBerita SET idDataWebsite
  VARIABLE tglBerita = FUNCTION date BASED ON "l,j F Y",FUNCTION strtotime BASED
```

```
ON VARIABLE lsBerita SET tgl
  VARIABLE jamBerita = FUNCTION date BASED ON "h:i a",FUNCTION strtotime BASED ON
VARIABLE lsBerita SET jam
  VARIABLE ls[].= VARIABLE lsBerita SET judulDataWebsite
  VARIABLE ls[].= VARIABLE tglBerita.VARIABLE jamBerita
  VARIABLE ls[].= VARIABLE cutTxt SET FUNCTION cut BASED ON VARIABLE lsBerita SET
isiDataWebsite,25
  END
  VARIABLE query = "index.php?admin=cpanel&amp;page=Berita&amp;"
  IF VARIABLE numBerita>VARIABLE limit THEN
    VARIABLE setHtml.= VARIABLE paging SET FUNCTION setPaging BASED ON VARIABLE
numBerita,VARIABLE limit,VARIABLE query
  ELSE
    VARIABLE setHtml.= ''
  ENDIF

  foreach VARIABLE ls as VARIABLE getDataBerita
  BEGIN
  VARIABLE setHtml.= VARIABLE getDataBerita
  END

  IF FUNCTION isset BASED ON VARIABLE _SESSION['login'] and VARIABLE
_SESSION['login']=true THEN
    VARIABLE data = VARIABLE setHtml
  ELSE
    VARIABLE data = ''
    FUNCTION header BASED ON "Location:index.php"
  ENDIF
ELSE
  IF FUNCTION empty BASED ON VARIABLE index THEN
    VARIABLE limit = 5
    VARIABLE start = VARIABLE paging SET FUNCTION getStart BASED ON VARIABLE limit
    VARIABLE isiBerita = VARIABLE dataBerita SET FUNCTION listBerita VARIABLE
start,VARIABLE limit
    VARIABLE numBerita = VARIABLE dataBerita SET FUNCTION numBerita
    foreach VARIABLE isiBerita as VARIABLE lsBerita
    BEGIN
      VARIABLE idBerita=VARIABLE lsBerita SET idDataWebsite
      VARIABLE tglBerita = FUNCTION date BASED ON "l,j F Y",FUNCTION strtotime
BASED ON VARIABLE lsBerita SET tgl
      VARIABLE jamBerita = FUNCTION date BASED ON "h:i a",FUNCTION strtotime BASED
ON VARIABLE lsBerita SET jam
      VARIABLE linkId = 'index.php?page=Berita&amp;go=detBerita&amp;Id='.VARIABLE
idBerita
      VARIABLE tambahId =
'index.php?page=Berita&amp;go=detBerita&amp;Id='.VARIABLE idBerita
      VARIABLE ls[] = VARIABLE lsBerita SET judulDataWebsite
      VARIABLE ls[].= VARIABLE tglBerita.VARIABLE jamBerita
      VARIABLE ls[].= VARIABLE cutTxt SET FUNCTION cut BASED ON VARIABLE lsBerita
SET isiDataWebsite,25
      VARIABLE ls[].= 'Selengkapnya'
    END

    VARIABLE query = "index.php?page=Berita&amp;"
    VARIABLE setHtml = VARIABLE getHtmlIndex[0]
    IF BASED VARIABLE numBerita>VARIABLE limit THEN
      VARIABLE setHtml.= VARIABLE paging SET FUNCTION setPaging BASED ON VARIABLE
numBerita,VARIABLE limit,VARIABLE query
    ELSE
      VARIABLE setHtml.= ''
    ENDIF
    foreach VARIABLE ls as VARIABLE getDataBerita
```

```
    BEGIN
      VARIABLE setHtml.= VARIABLE getDataBerita
    END
    VARIABLE setHtml.= VARIABLE getHtmlIndex[1]
    VARIABLE data = VARIABLE setHtml
  ENDIF
  IF VARIABLE index=1 THEN
    VARIABLE isiBerita = VARIABLE dataBerita SET FUNCTION listBerita BASED ON 0,1
    foreach VARIABLE isiBerita as VARIABLE lsBerita
    BEGIN
      VARIABLE idBerita = VARIABLE lsBerita SET idDataWebsite
      VARIABLE tglBerita = FUNCTION date BASED ON "l,j F Y",FUNCTION strtotime
BASED ON VARIABLE lsBerita SET tgl
      VARIABLE ls[]= VARIABLE tglBerita
      VARIABLE ls[].= VARIABLE lsBerita SET judulDataWebsite
      VARIABLE ls[].= VARIABLE cutTxt SET FUNCTION cut BASED ON VARIABLE lsBerita
SET isiDataWebsite,35
      VARIABLE ls[].= "[ selengkapnya ]"
    END
    foreach VARIABLE ls as VARIABLE getDataBerita
    BEGIN
      VARIABLE setHtml = VARIABLE getDataBerita
    END
    VARIABLE data['berita'] = VARIABLE setHtml
  ENDIF

  IF VARIABLE index=2 THEN
    VARIABLE isiBerita = VARIABLE dataBerita SET FUNCTION listBerita BASED ON 1,2
    foreach VARIABLE isiBerita as VARIABLE lsBerita
    BEGIN
    VARIABLE idBerita = VARIABLE lsBerita SET idDataWebsite
    VARIABLE ls[] = VARIABLE lsBerita SET judulDataWebsite
    END

    foreach VARIABLE ls as VARIABLE getDataBerita
    BEGIN
      VARIABLE setHtml.= VARIABLE getDataBerita
    END
    VARIABLE data['listBerita'] = VARIABLE setHtml
  ENDIF
ENDIF

RETURN VARIABLE data
```

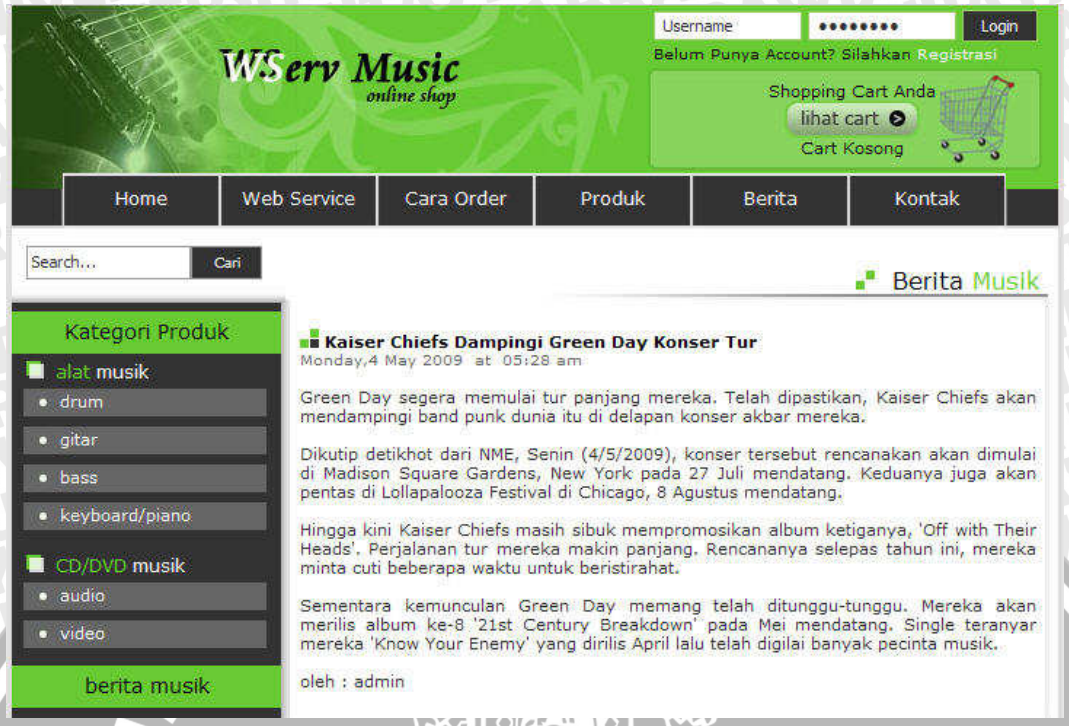
5.4.1.31. Implementasi Antarmuka Kebutuhan Fungsional Melihat Detail Berita

Setiap *user* dapat melihat detail berita pada Aplikasi *Web Service E-Commerce*. Fasilitas ini terdapat pada halaman *website*. Gambar 5.59 menunjukkan implementasi antarmuka untuk melihat detail berita.



Gambar 5. 65 : Implementasi Antarmuka untuk Melihat Detail Berita
Sumber : [Implementasi]

Untuk mengakses halaman detail berita, *user* harus menekan tombol ‘`selengkapnya`’ Gambar 5.60 menunjukkan hasil implementasi antarmuka untuk melihat detail berita.



Gambar 5. 66 : Hasil Implementasi Antarmuka untuk Melihat Detail Berita
Sumber : [Implementasi]

Implementasi algoritma untuk Melihat Detail Berita ditunjukkan dalam tabel berikut :

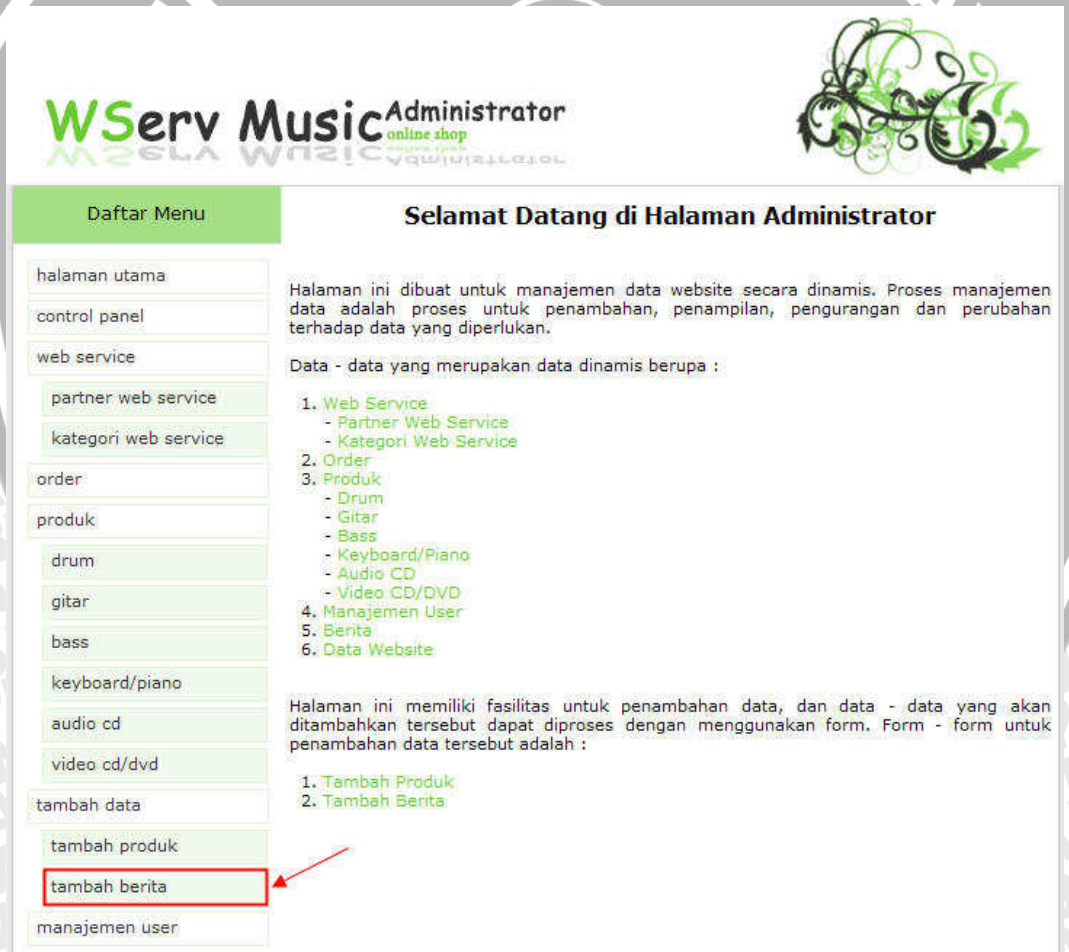
Algoritma 5. 31 : Algoritma Melihat Detail Berita

```
FUNCTION detBerita

VARIABLE htmlIndex = new FUNCTION htmlLayout
VARIABLE getHtmlIndex = VARIABLE htmlIndex SET FUNCTION htmlNonIndex BASED ON
'Berita','Musik'
VARIABLE this SET dataBerita = new FUNCTION dbBerita
VARIABLE id = VARIABLE _GET['Id']
VARIABLE isiBerita = VARIABLE this SET dataBerita SET FUNCTION detBerita BASED ON
VARIABLE id
VARIABLE tglBerita = FUNCTION date BASED "l,j F Y",FUNCTION strtotime BASED ON
VARIABLE isiBerita SET tgl
VARIABLE jamBerita = FUNCTION date BASED ON "h:i a",FUNCTION strtotime BASED ON
VARIABLE isiBerita SET jam
IF FUNCTION isset BASED ON VARIABLE _GET['admin'] THEN
  VARIABLE ls = VARIABLE isiBerita SET judulDataWebsite
  VARIABLE ls.= VARIABLE tglBerita.VARIABLE jamBerita
  VARIABLE ls.= VARIABLE isiBerita SET isiDataWebsite
  VARIABLE ls.= "oleh : ".VARIABLE isiBerita SET username

  IF FUNCTION isset BASED ON VARIABLE _SESSION['login'] and VARIABLE
_SESSION['login']=true THEN
    VARIABLE dataBerita['isi'] = VARIABLE ls
  ELSE
    VARIABLE dataBerita['isi'] = ""
    FUNCTION header BASED ON "Location:index.php"
  ENDIF
ELSE
```

```
  VARIABLE ls = VARIABLE getHtmlIndex[0]
  VARIABLE ls.= VARIABLE tglBerita.VARIABLE jamBerita
  VARIABLE ls.= VARIABLE isiBerita SET isiDataWebsite
  VARIABLE ls.= 'oleh : '.VARIABLE isiBerita SET username
  VARIABLE ls.= VARIABLE getHtmlIndex[1]
  VARIABLE dataBerita['isi'] = VARIABLE ls
ENDIF

RETURN VARIABLE dataBerita
```

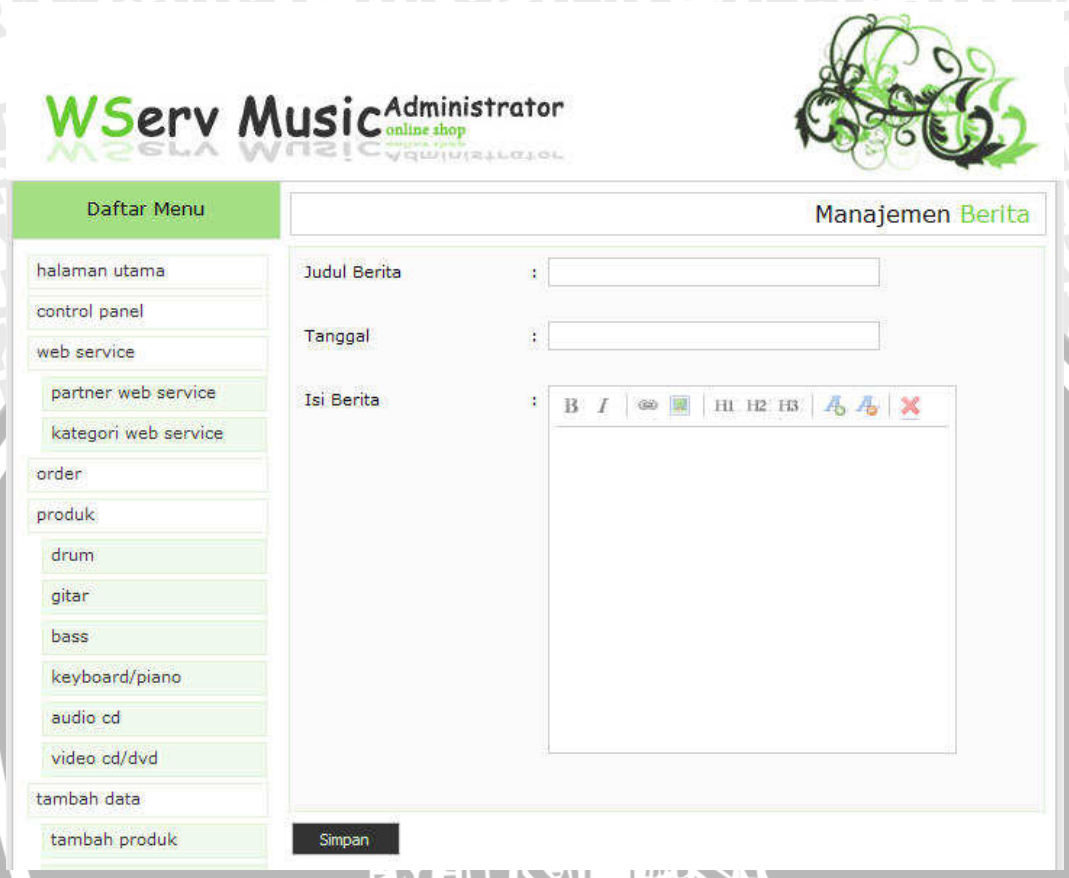
5.4.1.32. Implementasi Antarmuka Kebutuhan Fungsional Menambah Berita

Fasilitas menambah berita hanya dapat diakses oleh *administrator*. Fasilitas ini dapat diakses melalui halaman administrasi *website*. *Administrator* memilih menu '`tambah berita`' untuk membuka halaman *form* tambah berita. Gambar 5.63 menunjukkan implementasi antarmuka untuk menambah berita.



Gambar 5. 67 : Implementasi Antarmuka untuk Menambah Berita
Sumber : [Implementasi]

Setelah *administrator* memilih menu ‘tambah berita’, sistem menampilkan *form* tambah berita. *Administrator* dapat mengisi *form* tersebut dengan berita terbaru tentang seputar musik dan *entertainment*. Gambar 5.64 menunjukkan hasil implementasi antarmuka untuk menambah berita.



Gambar 5. 68 : Hasil Implementasi Antarmuka untuk Menambah Berita
Sumber : [Implementasi]

Implementasi algoritma untuk Menambah Berita ditunjukkan dalam tabel berikut :

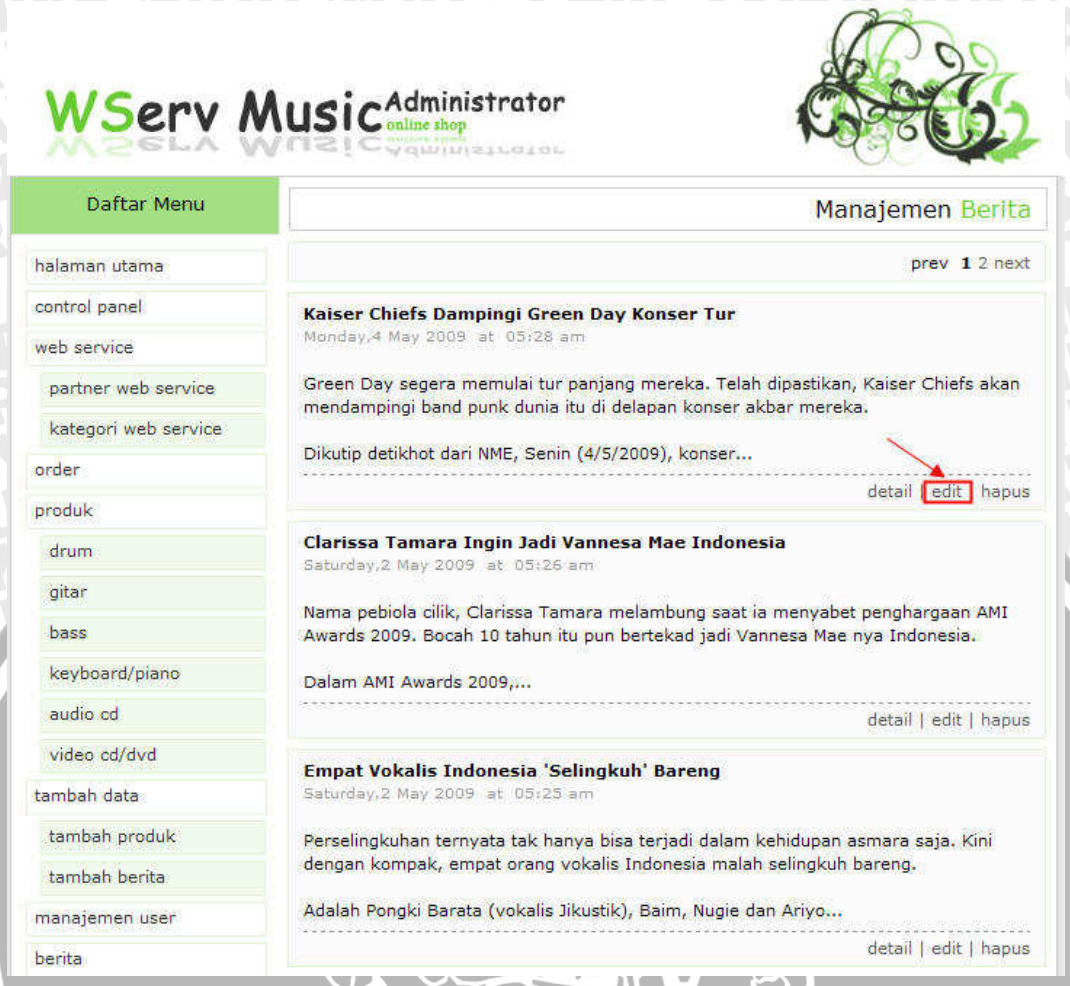
Algoritma 5. 32 : Algoritma Menambah Berita

```
FUNCTION cekFormTambahBerita

IF FUNCTION isset BASED ON VARIABLE _POST['tambahBerita'] THEN
  VARIABLE judulBerita = VARIABLE _POST['judulBerita']
  VARIABLE isiBerita = VARIABLE _POST['isiBerita']
  VARIABLE fieldBerita = array()
  foreach VARIABLE fieldBerita as VARIABLE keyFieldBerita=>VARIABLE valFieldBerita
  BEGIN
  IF FUNCTION empty BASED ON VARIABLE _POST[VARIABLE keyFieldBerita] THEN
    VARIABLE error[VARIABLE keyFieldBerita]= VARIABLE valFieldBerita
    VARIABLE post = ''
  ELSE
    VARIABLE error[VARIABLE keyFieldBerita] = " "
    VARIABLE post[VARIABLE keyFieldBerita] = VARIABLE _POST[VARIABLE
```

```
keyFieldBerita]
    VARIABLE post['jam'] = FUNCTION date BASED ON 'H:i:s'
  ENDIF
  END

  IF VARIABLE error[] = " " THEN
    VARIABLE msgBerita = "Data berita berhasil disimpan."
    VARIABLE this SET FUNCTION tambahBerita BASED ON VARIABLE post
  ELSE
    VARIABLE msgBerita = ''
  ENDIF

  VARIABLE data = VARIABLE this SET FUNCTION formTambahBerita BASED ON VARIABLE
error,VARIABLE msgBerita
ELSE
  VARIABLE data['inc'] = ""
  VARIABLE data['isi'] = ""
ENDIF

RETURN VARIABLE data
```

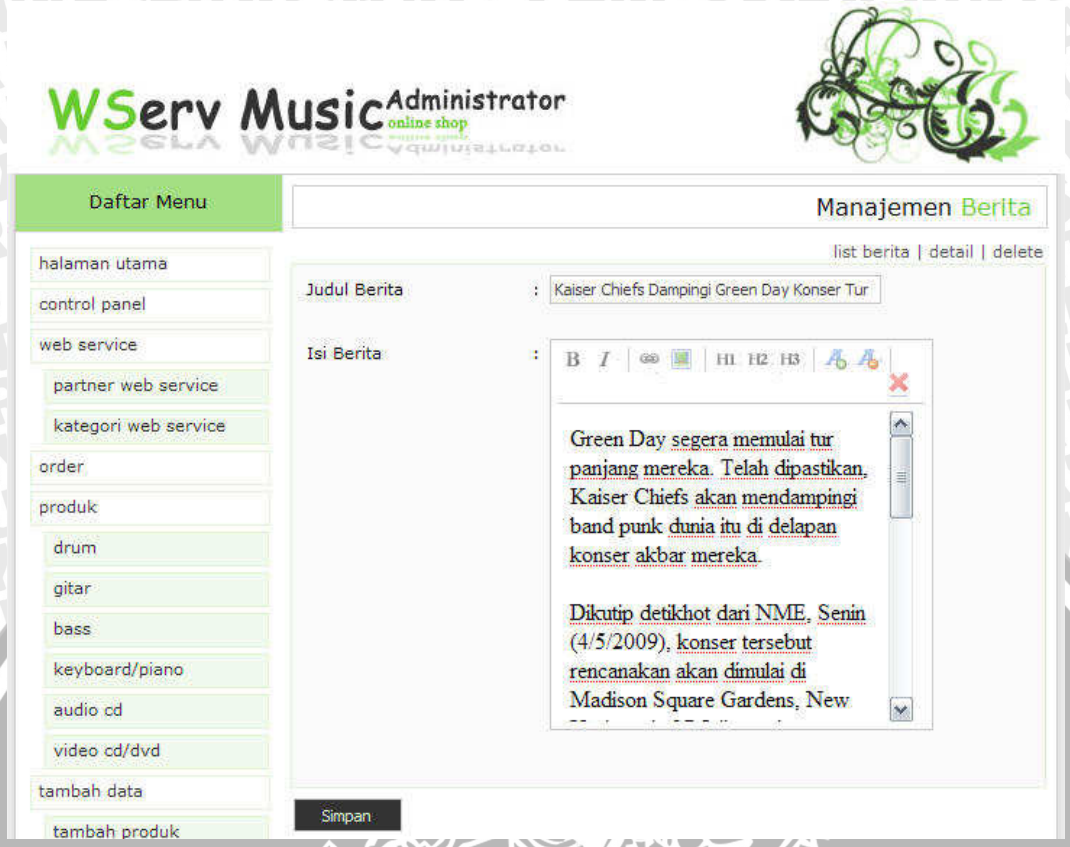
5.4.1.33. Implementasi Antarmuka Kebutuhan Fungsional Mengedit Berita

Fasilitas mengedit berita hanya dapat diakses oleh *administrator*. Fasilitas ini dapat diakses melalui halaman administrasi *website*. *Administrator* menekan tombol ‘edit’ untuk membuka halaman *form* edit berita. Gambar 5.65 menunjukkan implementasi antarmuka untuk mengedit berita.



Gambar 5. 69 : Implementasi Antarmuka untuk Mengedit Berita
Sumber : [Implementasi]

Setelah *administrator* menekan tombol '`edit`', sistem menampilkan halaman *form* berita. *Administrator* dapat mengedit data pada tiap – tiap *field form* edit berita. Gambar 5.66 menunjukkan hasil implementasi antarmuka untuk menghapus *order* barang.



Gambar 5. 70 : Hasil Implementasi Antarmuka untuk Mengedit Berita
Sumber : [Implementasi]

Implementasi algoritma untuk Mengedit Berita ditunjukkan dalam tabel berikut :

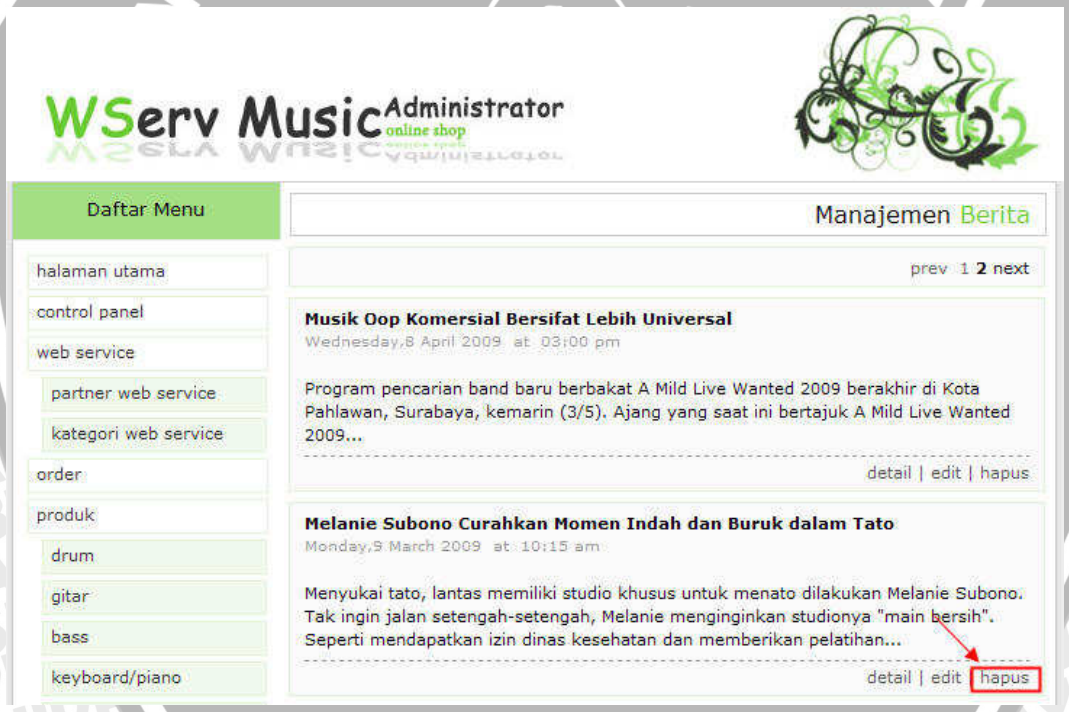
Algoritma 5. 33 : Algoritma Mengedit Berita

```
FUNCTION cekFormEditBerita

IF FUNCTION isset BASED ON VARIABLE _POST['editBerita'] THEN
  VARIABLE id = VARIABLE _GET['Id']
  VARIABLE fieldBerita = FUNCTION array()
  foreach VARIABLE fieldBerita as VARIABLE keyFieldBerita=>VARIABLE valFieldBerita
  BEGIN
  IF FUNCTION empty BASED ON VARIABLE _POST[VARIABLE keyFieldBerita] THEN
    VARIABLE error[VARIABLE keyFieldBerita] = VARIABLE valFieldBerita." harus
diisi."
    VARIABLE post = ''
  ELSE
    VARIABLE error[VARIABLE keyFieldBerita] = " "
    VARIABLE post[VARIABLE keyFieldBerita] = VARIABLE _POST[VARIABLE
keyFieldBerita]
  ENDIF
  END

  IF VARIABLE error[]= " " THEN
     VARIABLE msgBerita="Data berita berhasil disimpan."
     VARIABLE this SET FUNCTION editBerita BASED ON VARIABLE post,VARIABLE id
  ELSE
```

```
    VARIABLE msgBerita = ''
  ENDIF
  VARIABLE data = VARIABLE this SET FUNTION formEditBerita BASED ON VARIABLE
error,VARIABLE msgBerita
ELSE
  VARIABLE data['inc'] = ""
  VARIABLE data['isi'] = ""
ENDIF

RETURN VARIABLE data
```

5.4.1.34. Implementasi Antarmuka Kebutuhan Fungsional Menghapus Berita

Fasilitas menghapus berita hanya dapat diakses oleh *administrator*. Fasilitas ini dapat diakses melalui halaman administrasi *website*. *Administrator* menekan tombol 'hapus' untuk menghapus berita. Gambar 5.67 menunjukkan implementasi antarmuka untuk menghapus berita.



Gambar 5. 71 : Implementasi Antarmuka untuk Menghapus Berita
Sumber : [Implementasi]

Setelah *administrator* menekan tombol 'hapus', sistem menghapus data berita dari sistem. Gambar 5.68 menunjukkan hasil implementasi antarmuka untuk menghapus berita.

Gambar 5. 72 : Hasil Implementasi Antarmuka untuk Menghapus Berita
Sumber : [Implementasi]

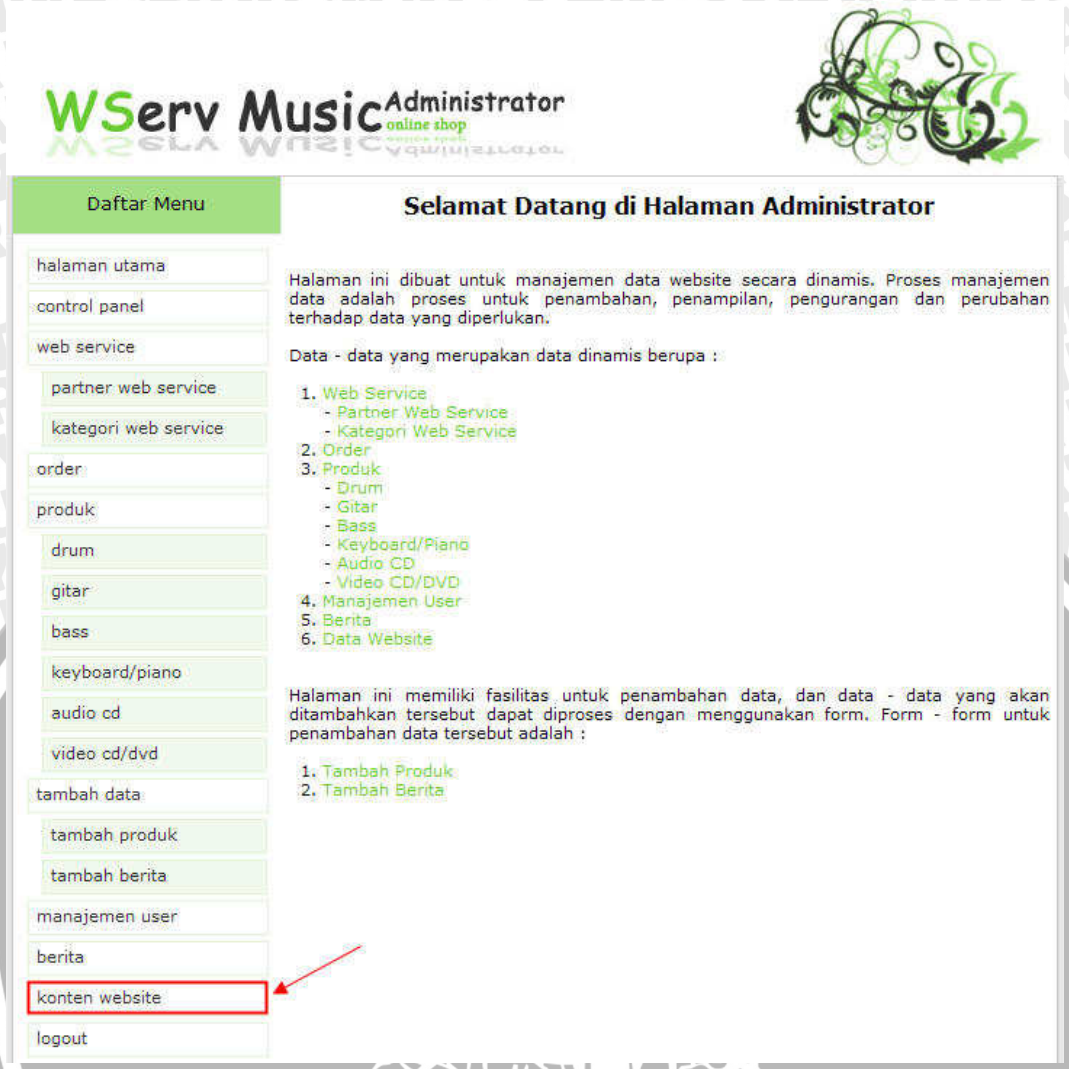
Implementasi algoritma untuk Menghapus Berita ditunjukkan dalam tabel berikut :

Algoritma 5. 34 : Algoritma Menghapus Berita

```
FUNCTION hapusBerita

IF FUCNTION isset BASED ON VARIABLE _SESSION['login'] AND VARIABLE
_SESSION['login'] = true THEN
  VARIABLE dbBerita = new FUNCTION dbBerita
  VARIABLE id = VARIABLE _GET['Id']
  VARIABLE detBerita = VARIABLE dbBerita SET detBerita BASED ON VARIABLE id
  VARIABLE setHtml= "Data berita tentang ".VARIABLE detBerita SET
judulDataWebsite." telah berhasil dihapus."
  VARIABLE dbBerita SET FUNCTION hapusBerita BASED ON VARIABLE id
  VARIABLE data['isi'] = VARIABLE setHtml
ELSE
  VARIABLE data['isi'] = ''
ENDIF

RETURN VARIABLE data
```

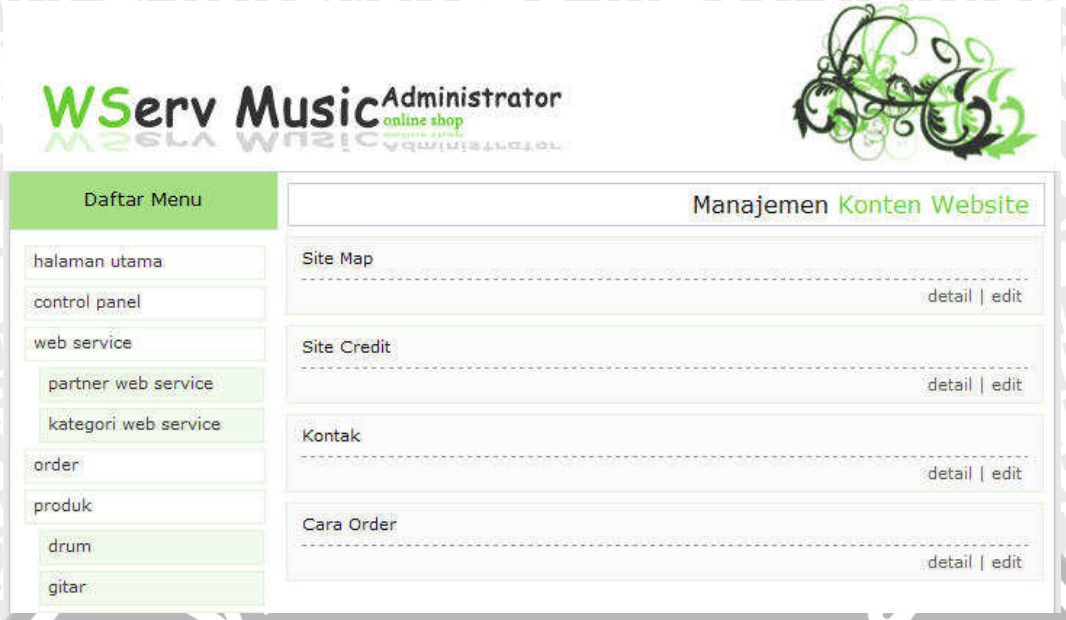
5.4.1.35. Implementasi Antarmuka Kebutuhan Fungsional Melihat Daftar Konten *Website*

Fasilitas melihat daftar konten *website* hanya dapat diakses oleh *administrator*. Fasilitas ini diakses melalui halaman administrasi *website*. *Administrator* memilih menu '`konten website`' untuk membuka halaman daftar konten *website*. Gambar 5.69 menunjukkan implementasi antarmuka untuk melihat daftar konten *website*.



Gambar 5. 73 : Implementasi Antarmuka untuk Melihat Daftar Konten *Website*
Sumber : [Implementasi]

Setelah *administrator* memilih menu ‘konten website’, sistem menampilkan halaman daftar konten *website*. Gambar 5.70 menunjukkan hasil implementasi antarmuka untuk melihat daftar konten *website*.



Gambar 5. 74 : Hasil Implementasi Antarmuka untuk Melihat Daftar Konten *Website*
Sumber : [Implementasi]

Implementasi algoritma untuk Melihat Daftar Konten *Website* ditunjukkan dalam tabel berikut :

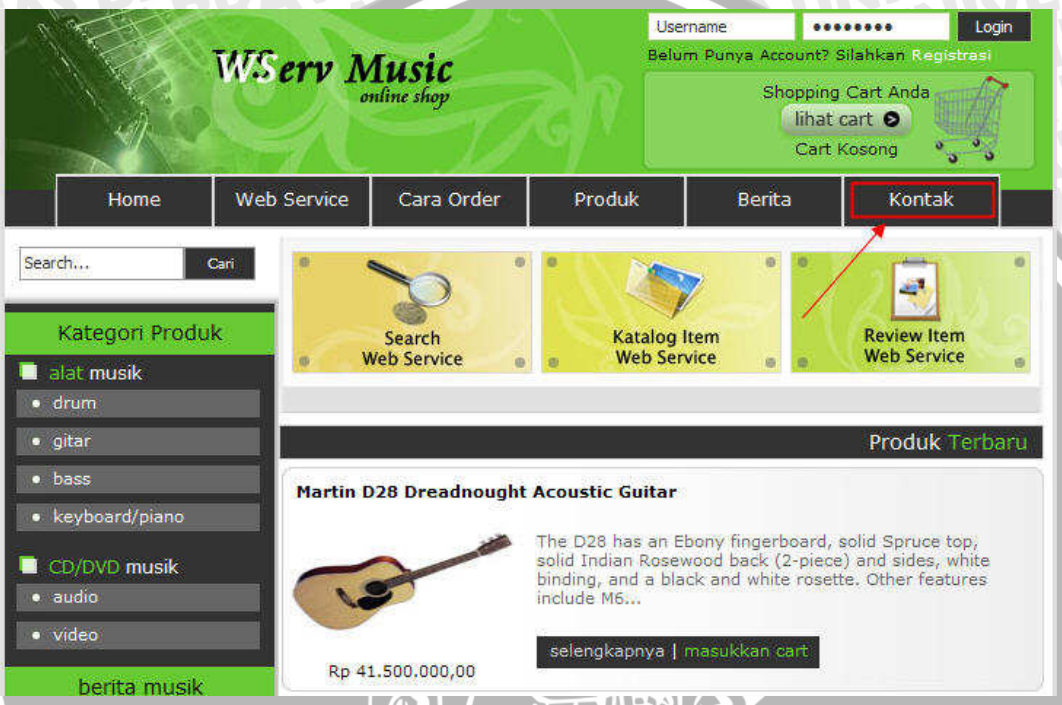
Algoritma 5. 35 : Algoritma Melihat Daftar Konten *Website*

```
FUNCTION listKonten

VARIABLE dbKonten = new FUNCTION dbKonten
IF FUNCTION isset BASED ON VARIABLE _GET['admin'] THEN
  VARIABLE isiKonten = VARIABLE dbKonten SET FUNCTION listKonten
  foreach VARIABLE isiKonten as VARIABLE lsKonten
  BEGIN
   VARIABLE idKonten = VARIABLE lsKonten SET idDataWebsite
   VARIABLE ls[] = VARIABLE lsKonten SET judulDataWebsite
  END

  foreach VARIABLE ls as VARIABLE getDataKonten
  BEGIN
   VARIABLE setHtml.=VARIABLE getDataKonten
  END
ELSE
  VARIABLE setHtml = ''
  FUNCTION header BASED ON "location:index.php"
ENDIF

RETURN VARIABLE setHtml
```

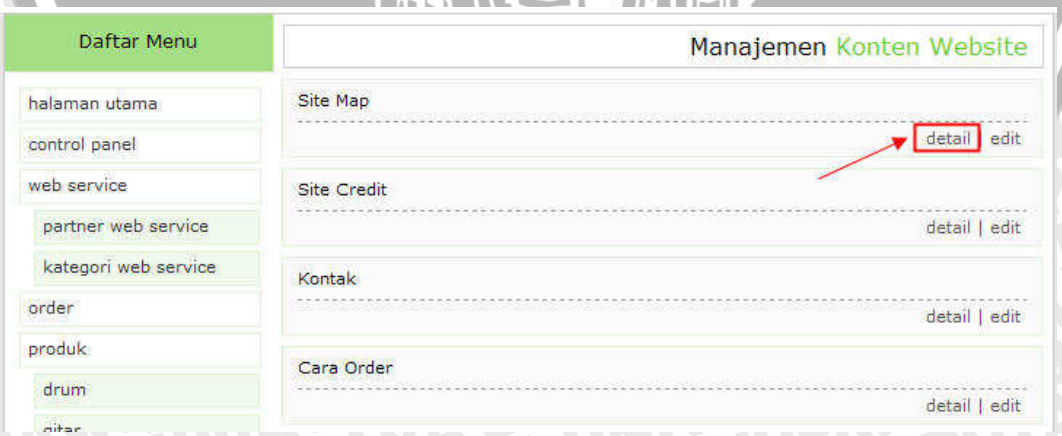
5.4.1.36. Implementasi Antarmuka Kebutuhan Fungsional Melihat Detail Konten *Website*

Fasilitas melihat detail konten *website* dapat diakses oleh *user* melalui halaman *website*. Fasilitas ini juga dapat diakses oleh *administrator* melalui halaman administrasi *website*.

Pada halaman *website*, fasilitas ini berupa menu utama dan *internal link*. Sedangkan pada halaman administrasi *website*, fasilitas ini dapat diakses dengan menekan tombol '`detail`'. Gambar 5.71 menunjukkan implementasi antarmuka untuk melihat detail konten *website* melalui halaman *website*. Dan Gambar 5.72 menunjukkan implementasi antarmuka untuk melihat detail konten *website* melalui halaman administrasi *website*.

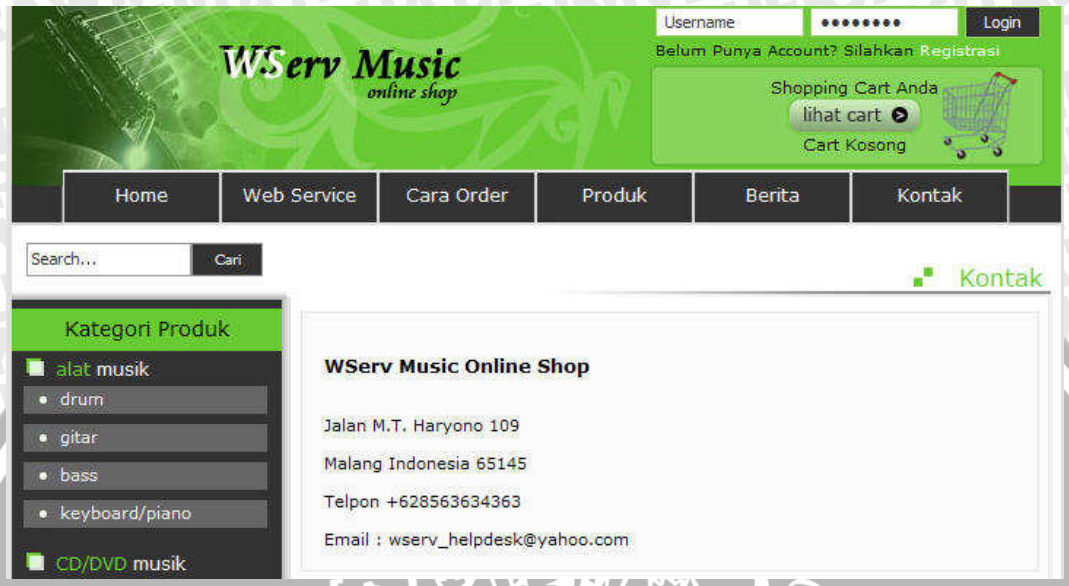


Gambar 5. 75 : Implementasi Antarmuka untuk Melihat Detail Konten *Website* melalui Halaman *Website*
Sumber : [Implementasi]



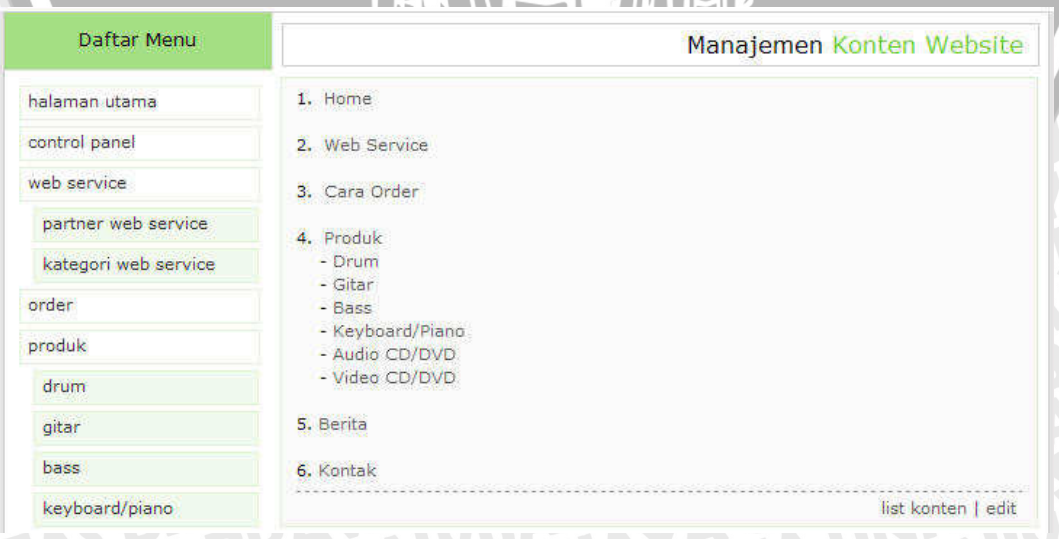
Gambar 5. 76 : Implementasi Antarmuka untuk Melihat Detail Konten *Website* melalui Halaman Administrasi *Website*
Sumber : [Implementasi]

Setelah *user* memilih salah satu menu utama *website* atau *internal link* pada halaman *website*, sistem menampilkan data detail *konten*. Gambar 5.73 menunjukkan hasil implementasi antarmuka untuk melihat detail konten *website* melalui halaman *website*.



Gambar 5. 77 : Hasil Implementasi Antarmuka untuk Melihat Detail Konten *Website* melalui Halaman *Website*
Sumber : [Implementasi]

Pada halaman administrasi website, setelah administrator menekan tombol 'detail', sistem menampilkan data detail konten. Gambar 5.74 menunjukkan hasil implementasi antarmuka untuk melihat detail konten website melalui halaman administrasi website.



Gambar 5. 78 : Hasil Implementasi Antarmuka untuk Melihat Detail Konten *Website* melalui Halaman Administrasi *Website*

Sumber : [Implementasi]

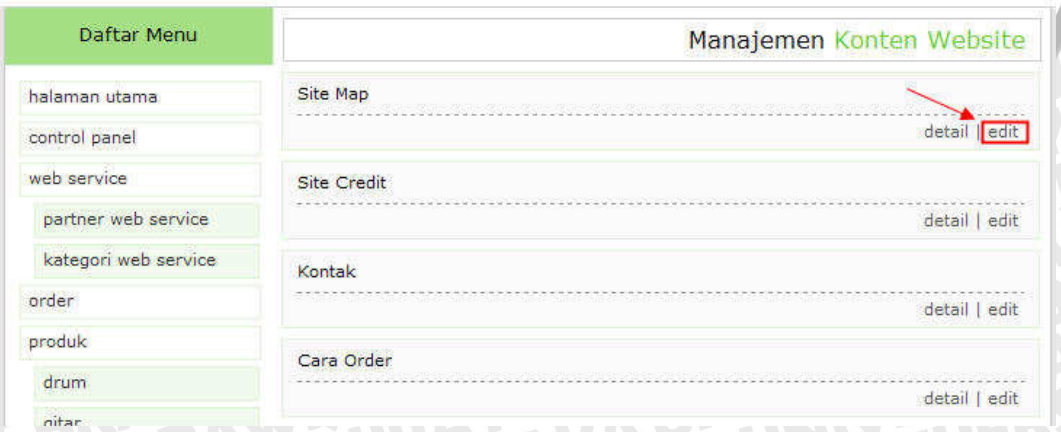
Implementasi algoritma untuk Melihat Detail Konten *Website* ditunjukkan dalam tabel berikut :

Algoritma 5. 36 : Algoritma Melihat Detail Konten *Website*

```
FUNCTION detKonten

VARIABLE dbKonten = new FUNCTION dbKonten
VARIABLE id = VARIABLE _GET['Id']
VARIABLE isiKonten = VARIABLE dbKonten SET FUNCTION detKonten BASED ON VARIABLE id
VARIABLE idKonten = VARIABLE isiKonten SET idDataWebsite

IF FUNCTION isset BASED ON VARIABLE _GET['admin'] THEN
  VARIABLE ls.= VARIABLE isiKonten SET isiDataWebsite
  VARIABLE dataKonten['isi'] = VARIABLE ls;
ELSE
  VARIABLE htmlIndex = new FUNCTION htmlLayout
  VARIABLE getHtmlIndex = VARIABLE htmlIndex SET FUNCTION htmlNonIndex BASED ON
'',FUNCTION ucwords BASED ON VARIABLE isiKonten SET judulDataWebsite
  VARIABLE ls = VARIABLE getHtmlIndex[0]
  VARIABLE ls.= VARIABLE isiKonten SET isiDataWebsite
  VARIABLE ls.= VARIABLE getHtmlIndex[1]
  VARIABLE dataKonten['isi'] = VARIABLE ls
ENDIF

RETURN VARIABLE dataKonten
```

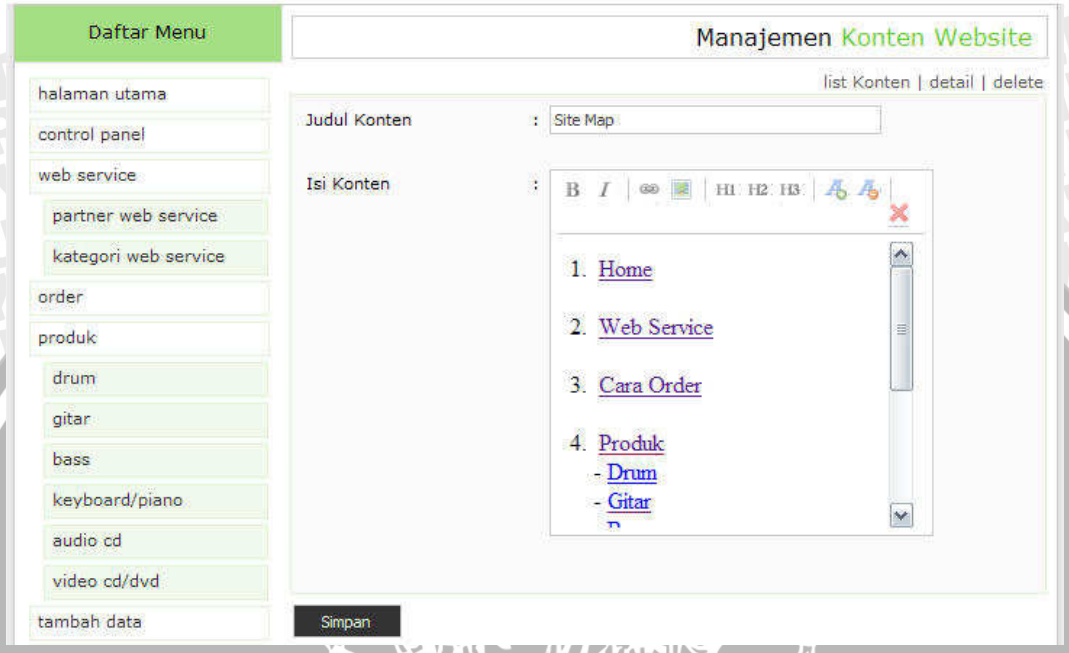
5.4.1.37. Implementasi Antarmuka Kebutuhan Fungsional Mengedit Konten *Website*

Fasilitas mengedit konten *website* hanya dapat diakses oleh *administrator*. Fasilitas ini dapat diakses melalui halaman administrasi *website*. *Administrator* menekan tombol '`edit`' untuk membuka halaman *form* edit konten. Gambar 5.75 menunjukkan implementasi antarmuka untuk mengedit konten *website*.



Gambar 5. 79 : Implementasi Antarmuka untuk Mengedit Konten *Website*
Sumber : [Implementasi]

Setelah *administrator* menekan tombol '`edit`', sistem menampilkan *form* edit konten *website*. *Administrator* dapat mengedit data yang terdapat pada *filed form* edit konten *website*. Gambar 5.76 menunjukkan hasil implementasi antarmuka untuk menghapus *order* barang.



Gambar 5. 80 : Hasil Implementasi Antarmuka untuk Mengedit Konten *Website*
Sumber : [Implementasi]

Implementasi algoritma untuk Mengedit Konten *Website* ditunjukkan dalam tabel berikut :

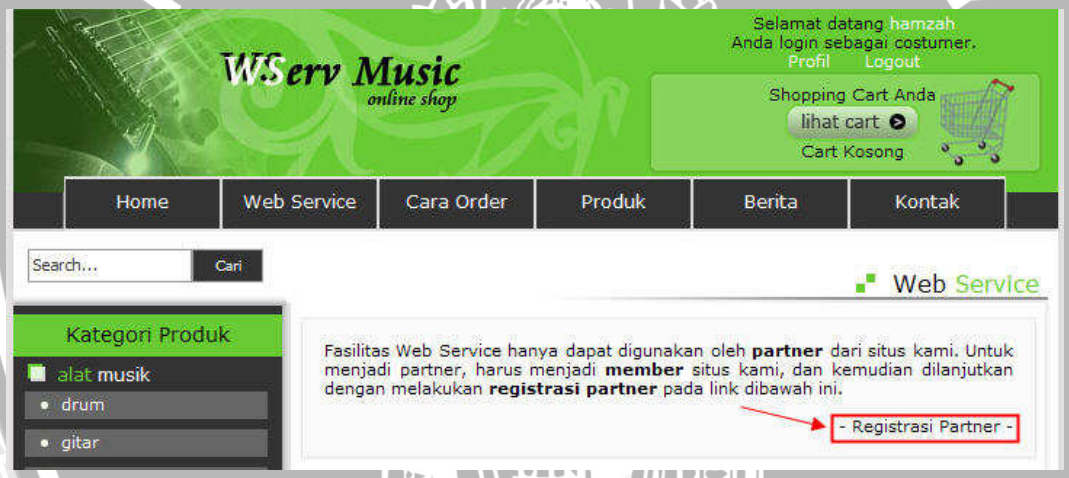
Algoritma 5. 37 : Algoritma Mengedit Konten *Website*

```
FUNCTION cekFormEditKonten

IF FUNCTION isset BASED ON VARIABLE _POST['editKonten'] THEN
  VARIABLE id = VARIABLE _GET['Id']
  VARIABLE fieldKonten = FUNCTION array()
  foreach VARIABLE fieldKonten as VARIABLE keyFieldKonten=>VARIABLE valFieldKonten
  BEGIN
  IF FUNCTION empty BASED ON VARIABLE _POST[VARIABLE keyFieldKonten] THEN
    VARIABLE error[VARIABLE keyFieldKonten]= VARIABLE valFieldKonten." harus
diisi."
    VARIABLE post = ''
  ELSE
    VARIABLE error[VARIABLE keyFieldKonten] = " "
    VARIABLE post[VARIABLE keyFieldKonten] = VARIABLE _POST[VARIABLE
keyFieldKonten]
  ENDIF
  END

  IF VARIABLE error[] = " " THEN
    VARIABLE msgKonten = "Data Konten berhasil disimpan.";
```

```
    VARIABLE this SET FUNCTION editKonten BASED ON VARIABLE post,VARIABLE id
  ELSE
    VARIABLE msgKonten = ''
  ENDIF
  VARIABLE data = VARIABLE this SET formEditKonten BASED ON VARIABLE
error,VARIABLE msgKonten
ELSE
  VARIABLE data = ""
ENDIF

RETURN VARIABLE data
```

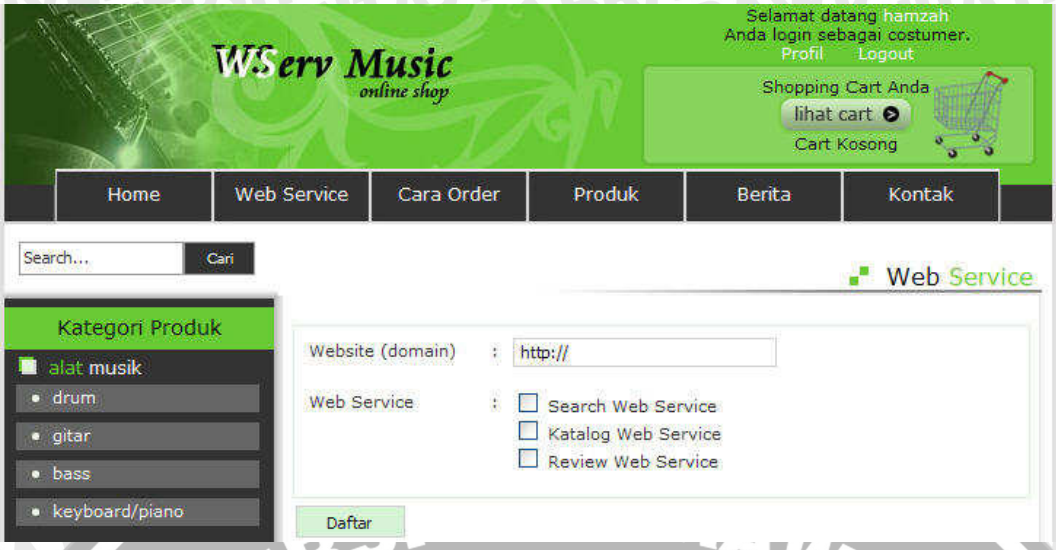
5.4.1.38. Implementasi Antarmuka Kebutuhan Fungsional Registrasi *Website Partner*

Aplikasi *Web Service E-Commerce* memiliki fasilitas registasi *website partner*. Fasilitas ini dapat diakses oleh aktor *customer* yang ingin menjadi *partner web service*. Gambar 5.77 menunjukkan implementasi antarmuka untuk registasi *website partner*.



Gambar 5. 81 : Implementasi Antarmuka untuk Registrasi *Website Partner*
Sumber : [Implementasi]

Untuk mengakses halaman registrasi *website partner*, *customer* harus menekan tombol '`Registrasi Partner`' pada halaman *web service*. Gambar 5.78 menunjukkan hasil implementasi antarmuka untuk registrasi *website partner*.



Gambar 5. 82 : Hasil Implementasi Antarmuka untuk Registrasi *Website Partner*
Sumber : [Implementasi]

Implementasi algoritma untuk Registrasi *Website Partner* ditunjukkan dalam tabel berikut :

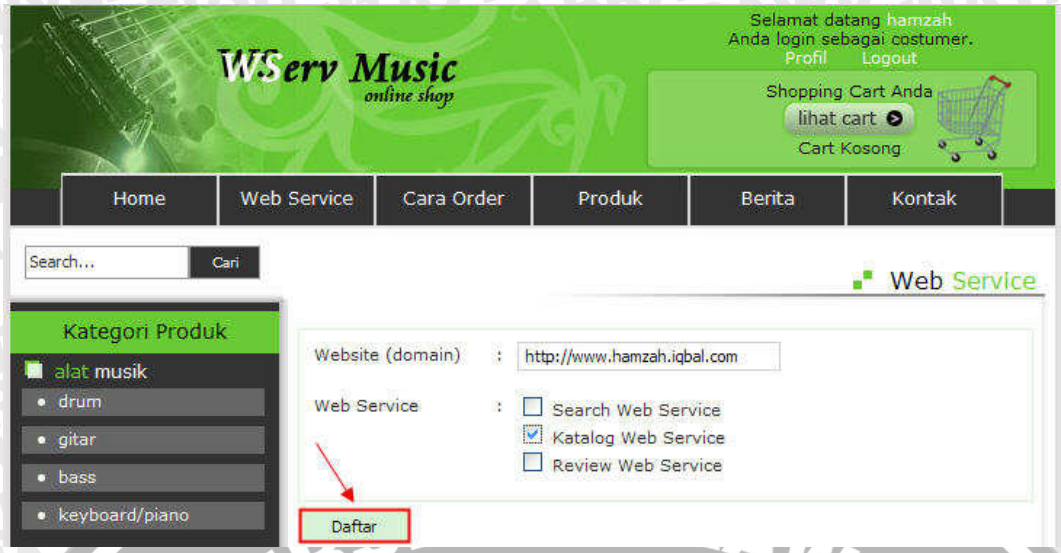
Algoritma 5. 38 : Algoritma Registrasi *Website Partner*

```
FUNCTION cekRegistrasiPartner

VARIABLE dbWs = new FUNCTION dbWebService
VARIABLE idUser = VARIABLE _SESSION['idUser']
IF FUNCTION empty BASED ON VARIABLE _POST['url'] or VARIABLE _POST['url'] =
“http://” THEN
  VARIABLE error['url'] = "Website tidak boleh kosong."
  VARIABLE post = ""
ELSE
  VARIABLE error['url'] = " "
  VARIABLE post['url'] = VARIABLE _POST['url']
ENDIF
IF FUNCTION empty BASED ON VARIABLE _POST['ws'] THEN
  VARIABLE error['WS'] = "Pilih salah satu web service."
  VARIABLE post = ""
ENDIF

IF not FUNCTION empty BASED ON VARIABLE _POST['ws'] THEN
  VARIABLE error['WS'] = " "
  foreach(VARIABLE _POST['ws'] as VARIABLE ws
  BEGIN
  IF VARIABLE ws = 'sWS' THEN
    VARIABLE wse['search'] = 1
    VARIABLE file['search'] = 'search'
  ENDIF
  IF VARIABLE ws = 'kWS' THEN
    VARIABLE wse['katalog'] = 2
    VARIABLE file['katalog'] = 'katalog'
  ENDIF
  IF VARIABLE ws = 'rWS' THEN
    VARIABLE wse['review'] = 3
    VARIABLE file['review'] = 'review'
```

```
  ENDIF
  END
ENDIF

IF VARIABLE error['url'] = " " THEN
  VARIABLE msg = "Terima kasih, proses registrasi telah berhasil."
  VARIABLE cutData = FUNCTION substr BASED ON FUNCTION md5 BASED ON VARIABLE
post['url'],7,18
  VARIABLE aktifasi = FUNCTION base64_encode BASED ON VARIABLE cutData
  VARIABLE tgl = date BASED ON "Y-m-d"
  VARIABLE kode = "Kode Aktifasi Anda : ".VARIABLE aktifasi"
  VARIABLE kode.= "Silahkan download file ini (contoh soap file):"

  foreach VARIABLE file as VARIABLE download
  BEGIN
  VARIABLE kode.= VARIABLE download
  END
  VARIABLE kode.= "-- klik disini --"
  VARIABLE this SET tambahWebServicePartner BASED ON VARIABLE wse,VARIABLE
idUser,VARIABLE tgl
  VARIABLE this SET regWebServicePartner BASED ON VARIABLE post['url'],VARIABLE
aktifasi,VARIABLE idUser
  VARIABLE _SESSION['idKategoriUser'] = 3
ELSE
  VARIABLE msg = ''
  VARIABLE kode = ''
ENDIF
VARIABLE data = VARIABLE this SET formRegistrasiPartner BASED ON VARIABLE
error,VARIABLE msg,VARIABLE kode

RETURN VARIABLE data
```

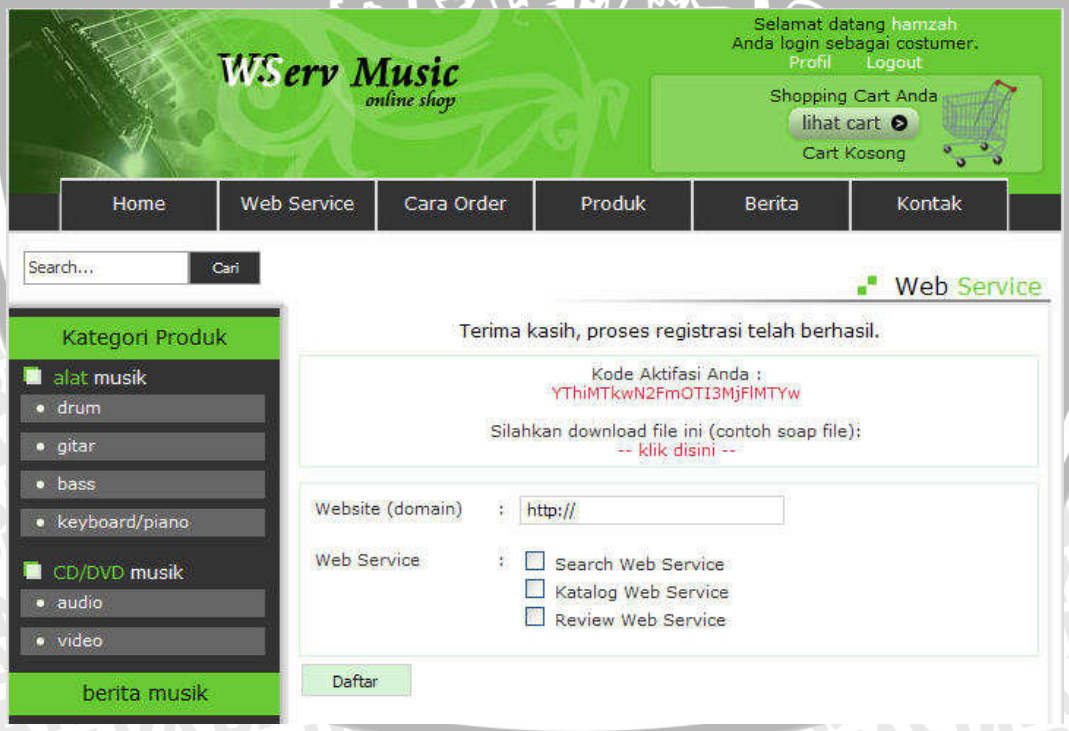
5.4.1.39. Implementasi Antarmuka Kebutuhan Fungsional *Download* SOAP *file*

Fasilitas download SOAP dapat diakses setelah *costumer* berhasil melakukan registrasi *website partner*. Gambar 5.79 menunjukkan implementasi antarmuka untuk *download* SOAP.



Gambar 5. 83 : Implementasi Antarmuka untuk *Download* SOAP
Sumber : [Implementasi]

Setelah *customer* berhasil mendaftarkan diri sebagai *partner*, sistem menampilkan halaman untuk men-*download* SOAP . Gambar 5.80 menunjukkan hasil implementasi antarmuka untuk *download* SOAP.



Gambar 5. 84 : Hasil Implementasi Antarmuka untuk *Download* SOAP
Sumber : [Implementasi]

Implementasi algoritma untuk *Download* SOAP ditunjukkan dalam tabel berikut :

Algoritma 5. 39 : Algoritma *Download* SOAP

```
FUNCTION cekRegistrasiPartner

VARIABLE dbWs = new FUNCTION dbWebService
VARIABLE idUser = VARIABLE _SESSION['idUser']
IF FUNCTION empty BASED ON VARIABLE _POST['url'] or VARIABLE _POST['url'] =
“http://” THEN
  VARIABLE error['url'] = "Website tidak boleh kosong."
  VARIABLE post = ""
ELSE
  VARIABLE error['url'] = " "
  VARIABLE post['url'] = VARIABLE _POST['url']
ENDIF
IF FUNCTION empty BASED ON VARIABLE _POST['ws'] THEN
  VARIABLE error['WS'] = "Pilih salah satu web service."
  VARIABLE post = ""
ENDIF

IF not FUNCTION empty BASED ON VARIABLE _POST['ws'] THEN
  VARIABLE error['WS'] = " "
  foreach(VARIABLE _POST['ws'] as VARIABLE ws
  BEGIN
  IF VARIABLE ws = 'sWS' THEN
    VARIABLE wse['search'] = 1
    VARIABLE file['search'] = 'search'
  ENDIF
  IF VARIABLE ws = 'kWS' THEN
    VARIABLE wse['katalog'] = 2
    VARIABLE file['katalog'] = 'katalog'
  ENDIF
  IF VARIABLE ws = 'rWS' THEN
    VARIABLE wse['review'] = 3
    VARIABLE file['review'] = 'review'
  ENDIF
  END
ENDIF

IF VARIABLE error['url'] = " " THEN
  VARIABLE msg = "Terima kasih, proses registrasi telah berhasil."
  VARIABLE cutData = FUNCTION substr BASED ON FUNCTION md5 BASED ON VARIABLE
post['url'],7,18
  VARIABLE aktifasi = FUNCTION base64_encode BASED ON VARIABLE cutData
  VARIABLE tgl = date BASED ON "Y-m-d"
  VARIABLE kode = "Kode Aktifasi Anda : ".VARIABLE aktifasi"
  VARIABLE kode.= "Silahkan download file ini (contoh soap file):"

  foreach VARIABLE file as VARIABLE download
  BEGIN
  VARIABLE kode.= VARIABLE download
  END
  VARIABLE kode.= "-- klik disini --"
  VARIABLE this SET tambahWebServicePartner BASED ON VARIABLE wse,VARIABLE
idUser,VARIABLE tgl
  VARIABLE this SET regWebServicePartner BASED ON VARIABLE post['url'],VARIABLE
aktifasi,VARIABLE idUser
  VARIABLE _SESSION['idKategoriUser'] = 3
ELSE
```

```
  VARIABLE msg = ''
  VARIABLE kode = ''
ENDIF
VARIABLE data = VARIABLE this SET formRegistrasiPartner BASED ON VARIABLE
error,VARIABLE msg,VARIABLE kode

RETURN VARIABLE data
```

5.4.1.40. Implementasi Antarmuka Kebutuhan Fungsional Melihat Daftar *Partner Web Service*

Fasilitas melihat daftar *partner web service* hanya dapat diakses oleh *administrator*. Fasilitas ini dapat diakses melalui halaman administrasi *website*. *Administrator* memilih menu '`partner web service`' untuk membuka halaman daftar *partner web service*. Gambar 5.81 menunjukkan implementasi antarmuka untuk melihat daftar *partner web service*.

Gambar 5. 85 : Implementasi Antarmuka untuk Melihat Daftar Partner *Web Service*
Sumber : [Implementasi]

Setelah *administrator* memilih menu '`partner web service`', sistem membuka halaman daftar *partner web service*. Gambar 5.82 menunjukkan hasil implementasi antarmuka untuk melihat daftar *partner web service*.

Gambar 5. 86 : Hasil Implementasi Antarmuka untuk Melihat Daftar *Partner Web Service*
Sumber : [Implementasi]

Implementasi algoritma untuk Melihat Daftar *Partner Web Service* ditunjukkan dalam tabel berikut :

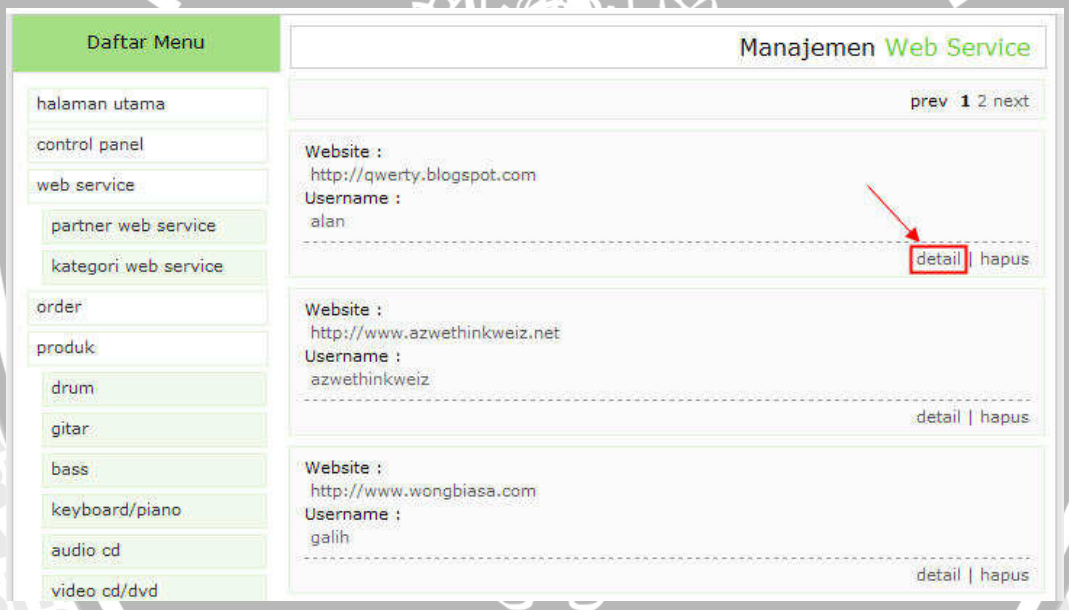
Algoritma 5. 40 : Algoritma Melihat Daftar *Partner Web Service*

```
FUNCTION listWebServicePartner

VARIABLE dbWs = new FUNCTION dbWebService
VARIABLE paging = new FUNCTION paging
VARIABLE htmlIndex = new FUNCTION htmlLayout
VARIABLE limit = 4
VARIABLE start = VARIABLE paging SET getStart BASED ON VARIABLE limit
VARIABLE listPartnerWS = VARIABLE dbWs SET FUNCTION listWebServicePartner BASED ON
VARIABLE start,VARIABLE limit
VARIABLE numWS = VARIABLE dbWs SET FUNCTION numWebService
foreach VARIABLE listPartnerWS as VARIABLE lsWS
BEGIN
  VARIABLE ls[] = "Website : "
  VARIABLE ls[].= VARIABLE lsWS SET website
  VARIABLE ls[].= "Username : "
  VARIABLE ls[].= VARIABLE lsWS SET username
END
VARIABLE query =
"index.php?admin=cpanel&amp;page=WebService&amp;go=listWebServicePartner&amp;"
VARIABLE getHtmlIndex = VARIABLE htmlIndex SET htmlNonIndex BASED ON
'Web','Service'
IF VARIABLE numWS>VARIABLE limit THEN
  VARIABLE setHtml.= VARIABLE paging SET FUNCTION setPaging BASED ON VARIABLE
numWS,VARIABLE limit,VARIABLE query
ELSE
```

```
  VARIABLE setHtml.= ''
ENDIF
foreach VARIABLE ls as VARIABLE lsWebService
BEGIN
  VARIABLE setHtml.= VARIABLE lsWebService
END
VARIABLE data['isi'] = VARIABLE setHtml

RETURN VARIABLE data
```

5.4.1.41. Implementasi Antarmuka Kebutuhan Fungsional Melihat Detail *Partner Web Service*

Fasilitas melihat detail *partner web service* hanya dapat diakses oleh *administrator*. Fasilitas ini dapat diakses melalui halaman administrasi *website*. *Administrator* menekan tombol '`detail`' untuk membuka halaman detail *partner web service*. Gambar 5.83 menunjukkan implementasi antarmuka untuk melihat detail *partner web service*.



Gambar 5. 87 : Implementasi Antarmuka untuk Melihat Detail *Partner Web Service*
Sumber : [Implementasi]

Setelah *administrator* menekan tombol '`detail`', sistem membuka halaman detail *partner web service*. Gambar 5.84 menunjukkan hasil implementasi antarmuka untuk melihat detail *partner web service*.

Gambar 5. 88 : Hasil Implementasi Antarmuka untuk Melihat Detail *Partner Web Service*
Sumber : [Implementasi]

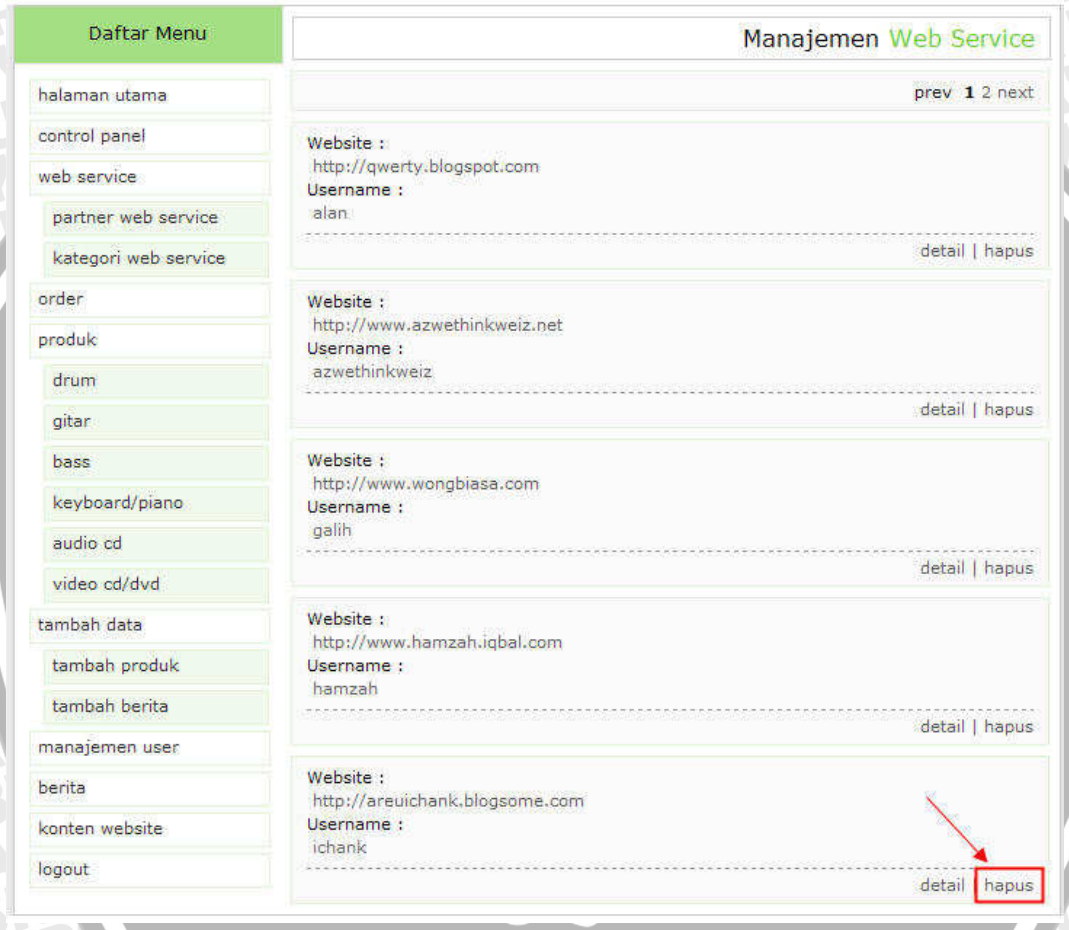
Implementasi algoritma untuk Melihat Detail *Partner Web Service* ditunjukkan dalam tabel berikut :

Algoritma 5. 41 : Algoritma Melihat Detail *Partner Web Service*

```
FUNCTION detWebServicePartner

VARIABLE dbWs = new FUNCTION dbWebService
VARIABLE htmlIndex = new FUNCTION htmlLayout
VARIABLE id = VARIABLE _GET['Id']
VARIABLE detPartnerWS = VARIABLE dbWs SET FUNCTION detWebServicePartner BASED ON
VARIABLE id

foreach VARIABLE detPartnerWS as VARIABLE detWS
BEGIN
  VARIABLE ls[] = VARIABLE detWS SET kategoriWebService
END
VARIABLE getHtmlIndex = VARIABLE htmlIndex SET FUNCTION htmlNonIndex BASED ON
'Web','Service'
VARIABLE setHtml.= "Username : "
VARIABLE setHtml.= VARIABLE detWS SET username
VARIABLE setHtml.= "Website : "
VARIABLE setHtml.= VARIABLE detWS SET website
VARIABLE setHtml.= "Aktif Mulai : "
VARIABLE setHtml.= FUNCTION date BASED ON "j F Y",FUNCTION strtotime BASED ON
VARIABLE detWS SET tgl
VARIABLE setHtml.= Web Service : "
foreach VARIABLE ls as VARIABLE lsWebService
BEGIN
  VARIABLE setHtml.= VARIABLE lsWebService
END
VARIABLE data['isi'] = VARIABLE setHtml

RETURN VARIABLE data;
```

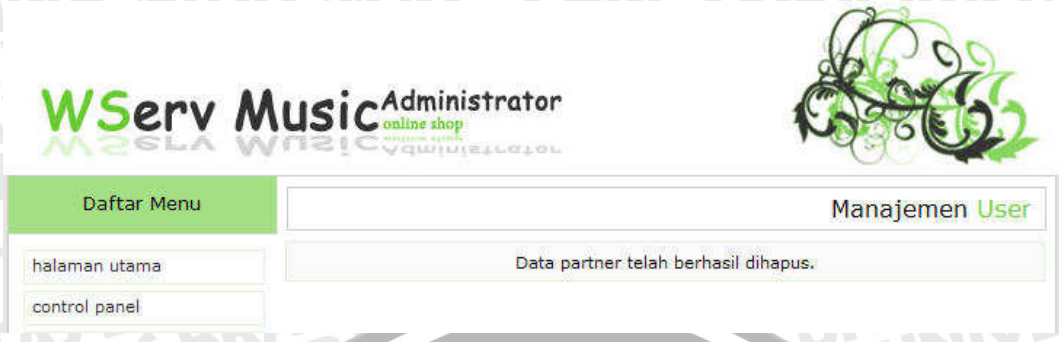
5.4.1.42. Implementasi Antarmuka Kebutuhan Fungsional Menghapus *Partner Web Service*

Fasilitas menghapus *partner web service* hanya dapat diakses oleh *administrator*. Fasilitas ini dapat diakses melalui halaman administrasi *website*. *Administrator* menekan tombol 'hapus' untuk menghapus *partner web service*. Gambar 5.85 menunjukkan implementasi antarmuka untuk menghapus *partner web service*.



Gambar 5. 89 : Implementasi Antarmuka untuk Menghapus *Partner Web Service*
Sumber : [Implementasi]

Setelah *administrator* menekan tombol 'hapus', sistem menghapus *partner web service* dari sistem. Gambar 5.86 menunjukkan hasil implementasi antarmuka untuk menghapus *partner web service*.



Gambar 5. 90 : Hasil Implementasi Antarmuka untuk Menghapus *Partner Web Service*
Sumber : [Implementasi]

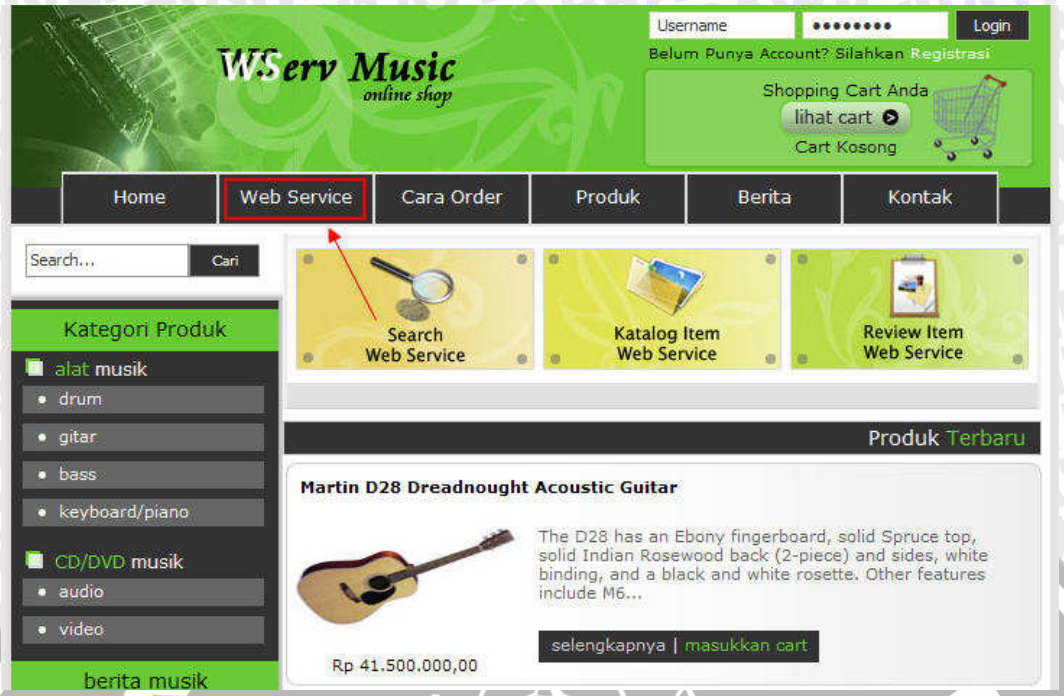
Implementasi algoritma untuk Menghapus *Partner Web Service* ditunjukkan dalam tabel berikut :

Algoritma 5. 42 : Algoritma Menghapus *Partner Web Service*

```
FUNCTION hapusWebServicePartner

VARIABLE dbWs = new FUNCTION dbWebService
VARIABLE id = VARIABLE _GET['Id']
VARIABLE setHtml = "Data partner telah berhasil dihapus."
VARIABLE dbWs SET FUNCTION hapusWebServicePartner BASED ON VARIABLE id
VARIABLE data['isi'] = VARIABLE setHtml

RETURN VARIABLE data
```

5.4.1.43. Implementasi Antarmuka Kebutuhan Fungsional Melihat Daftar Kategori *Web Service*

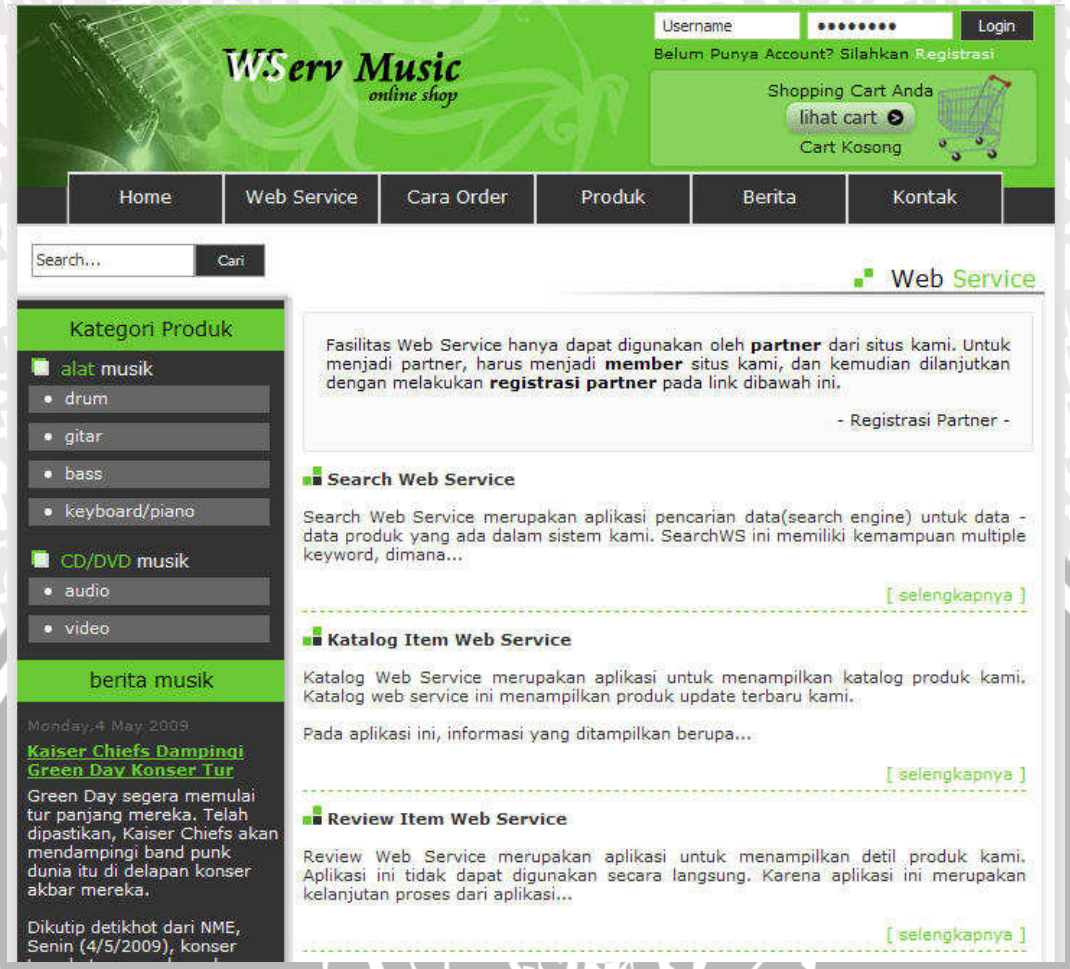
Fasilitas melihat daftar kategori *web service* dapat diakses oleh user di halaman website. Fasilitas ini juga dapat diakses oleh *administrator* di halaman administrasi *website*. Gambar 5.87 menunjukkan implementasi antarmuka untuk melihat kategori daftar *web service* melalui halaman *website*. Dan gambar 5.88 menunjukkan implementasi antarmuka untuk melihat kategori daftar *web service* melalui halaman administrasi *website*.



Gambar 5. 91 : Implementasi Antarmuka untuk Melihat Daftar Kategori *Web Service* melalui Halaman *Website*
Sumber : [Implementasi]

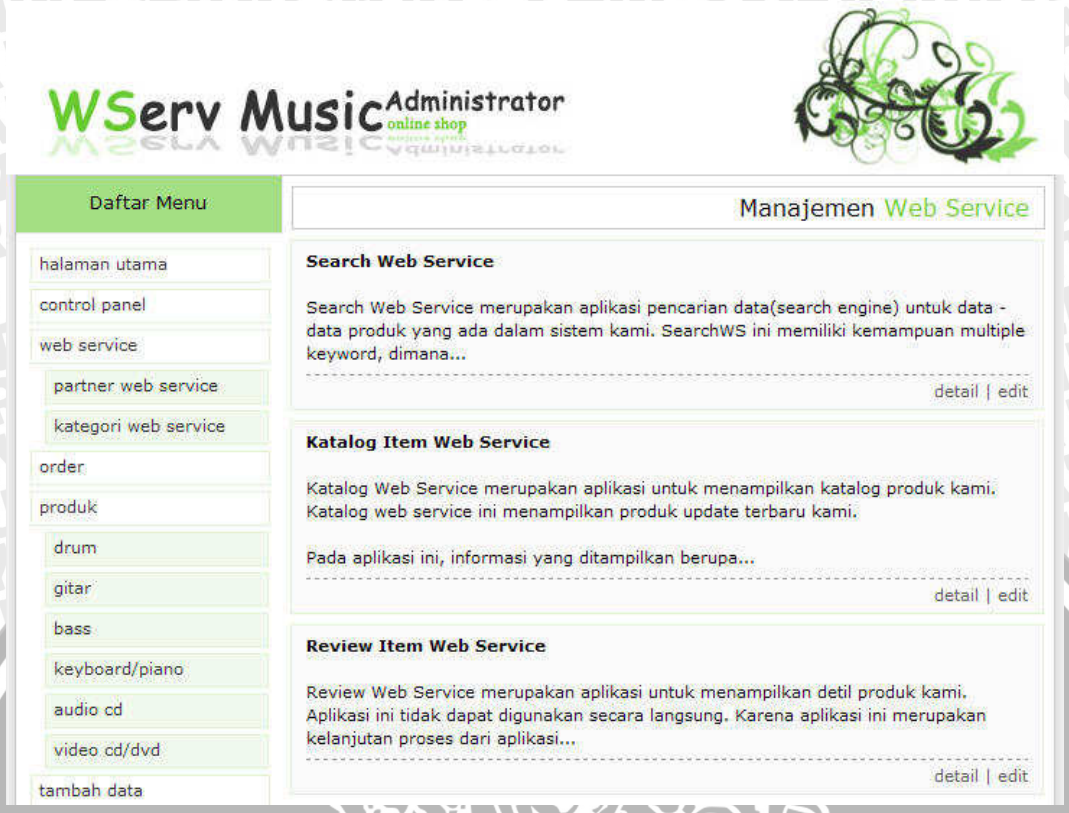
Gambar 5. 92 : Implementasi Antarmuka untuk Melihat Daftar Kategori *Web Service* melalui Halaman Administrasi *Website*
Sumber : [Implementasi]

Pada halaman *website*, *user* mengakses fasilitas melihat daftar kategori *web service* dengan memilih menu '`web service`' di menu utama. Gambar 5.89 menunjukkan hasil implementasi antarmuka untuk melihat daftar kategori *web service* melalui halaman *website*.



Gambar 5. 93 : Hasil Implementasi Antarmuka untuk Melihat Daftar Kategori *Web Service* melalui Halaman *Website*
Sumber : [Implementasi]

Pada halaman administrasi *website*, *administrator* mengakses fasilitas melihat daftar kategori *web service* dengan memilih menu '`kategori web service`'. Gambar 5.90 menunjukkan hasil implementasi antarmuka untuk melihat daftar kategori *web service* melalui halaman administrasi *website*.



Gambar 5. 94 : Hasil Implementasi Antarmuka untuk Melihat Daftar Kategori *Web Service* melalui Halaman Administrasi *Website*
Sumber : [Implementasi]

Implementasi algoritma untuk Melihat Daftar Kategori *Web Service* ditunjukkan dalam tabel berikut :

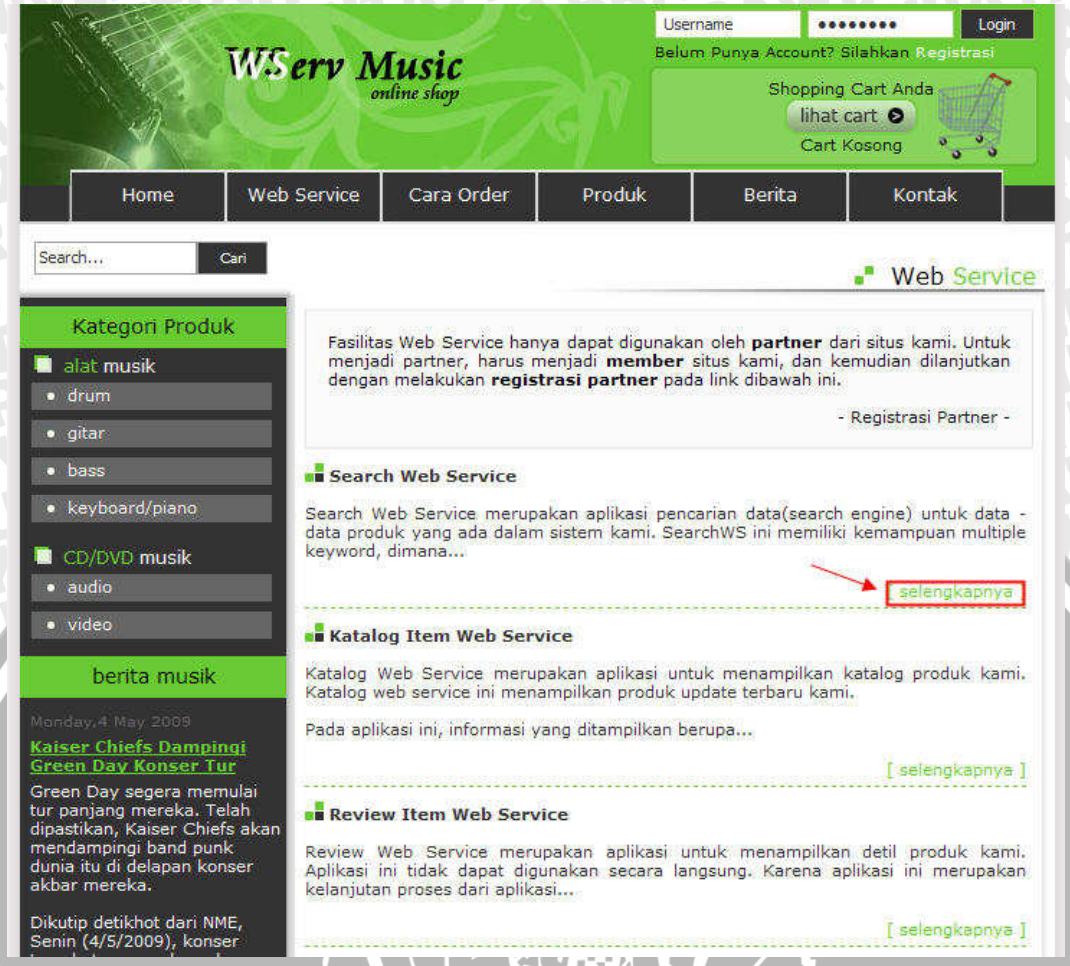
Algoritma 5. 43 : Algoritma Melihat Daftar Kategori *Web Service*

```
FUNCTION listKatWebService

VARIABLE cutTxt = new FUNCTION textCutting
VARIABLE htmlIndex = new FUNCTION htmlLayout
VARIABLE getHtmlIndex = VARIABLE htmlIndex SET FUNCTION htmlNonIndex BASED ON
'Web','Service'
VARIABLE dbWs = new FUNCTION dbWebService
VARIABLE listWS = VARIABLE dbWs SET listKatWebService

IF FUNCTION isset BASED ON VARIABLE _GET['admin'] THEN
  foreach VARIABLE listWS as VARIABLE lsWS
  BEGIN
    VARIABLE idKategoriWS = VARIABLE lsWS SET idKategoriWebService
    VARIABLE ls[] = VARIABLE lsWS SET kategoriWebService
    VARIABLE ls[].= VARIABLE cutTxt SET FUNCTION cut BASED ON VARIABLE lsWS SET
deskripsi,25
  END

  foreach VARIABLE ls as VARIABLE lsWebService
  BEGIN
    VARIABLE setHtml.= VARIABLE lsWebService
  END
ELSE
  foreach VARIABLE listWS as VARIABLE lsWS
```

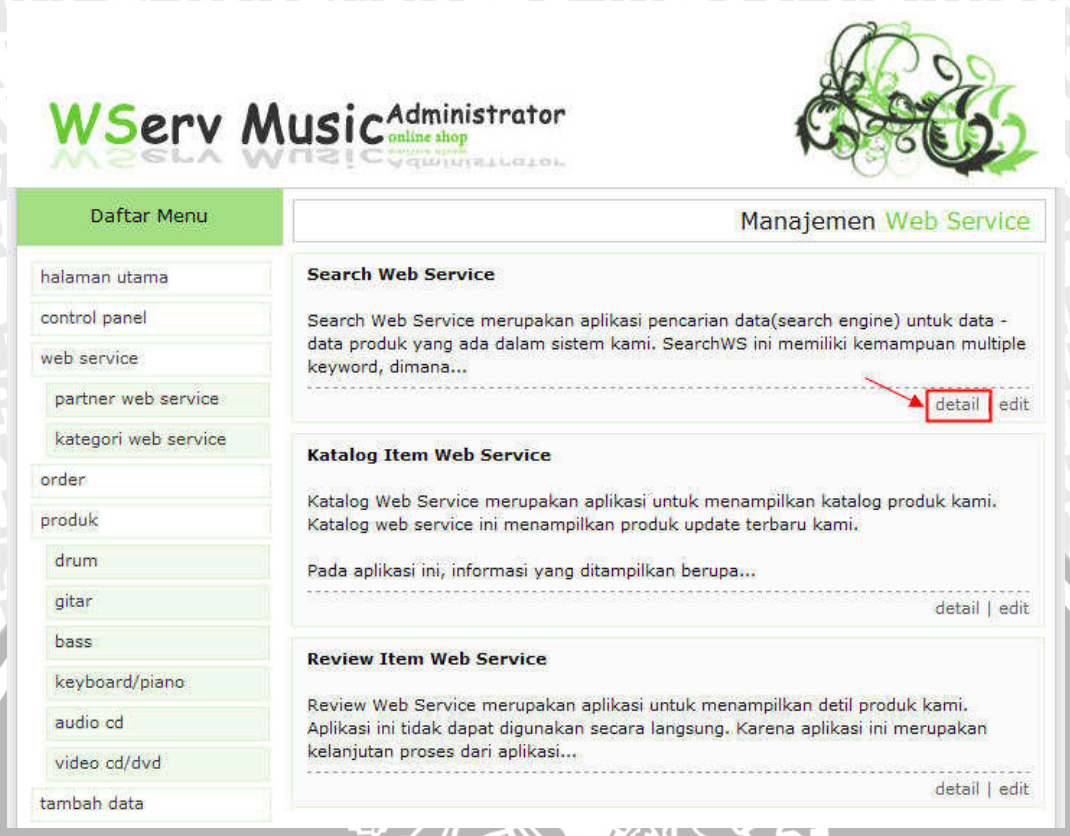
```
  BEGIN
    VARIABLE idKategoriWS = VARIABLE lsWS SET idKategoriWebService
    VARIABLE ls[] = VARIABLE lsWS SET kategoriWebService
    VARIABLE ls[].= VARIABLE cutTxt SET FUNCTION cut VARIABLE lsWS SET
deskripsi,25
  END
  VARIABLE setHtml = VARIABLE getHtmlIndex[0]
  IF FUNCTION isset BASED ON VARIABLE _SESSION['login'] and VARIABLE
_SESSION['idKategoriUser']!=3 or not FUNCTION isset BASED ON VARIABLE
_SESSION['login'] THEN
    VARIABLE setHtml.= FUNCTION text()
  ELSE
    VARIABLE setHtml.= ""
  ENDIF
  foreach VARIABLE ls as VARIABLE lsWebService
  BEGIN
    VARIABLE setHtml.= VARIABLE lsWebService
  END
  VARIABLE setHtml.= VARIABLE getHtmlIndex[1]
ENDIF

RETURN VARIABLE setHtml
```

5.4.1.44. Implementasi Antarmuka Kebutuhan Fungsional Melihat Detail Kategori *Web Service*

Fasilitas melihat detail kategori *web service* dapat diakses oleh *user* di halaman *website*. Fasilitas ini juga dapat diakses oleh *administrator* di halaman administrasi *website*. Gambar 5.91 menunjukkan implementasi antarmuka untuk melihat detail kategori *web service* melalui halaman *website*. Dan gambar 5.92 menunjukkan implementasi antarmuka untuk melihat detail kategori *web service*.



Gambar 5. 95 : Implementasi Antarmuka untuk Melihat Detail Kategori *Web Service* melalui Halaman *Website*
Sumber : [Implementasi]



Gambar 5. 96 : Implementasi Antarmuka untuk Melihat Detail Kategori *Web Service* melalui Halaman Administrasi *Website*
Sumber : [Implementasi]

Pada halaman *website*, *user* mengakses halaman detail kategori *web service* dengan menekan tombol '`selengkapnya`'. Gambar 5.93 menunjukkan hasil implementasi antarmuka untuk melihat detail kategori *web service* melalui halaman *website*.

Gambar 5. 97 : Hasil Implementasi Antarmuka untuk Menghapus *Partner Web Service* melalui Halaman *Website*
Sumber : [Implementasi]

Pada halaman administrasi *website*, *administrator* mengakses halaman detail kategori *web service* dengan menekan tombol '`detail`'. Gambar 5.94 menunjukkan hasil implementasi antarmuka untuk melihat detail kategori *web service* melalui halaman *website* melalui halaman administrasi *website*.

Gambar 5. 98 : Hasil Implementasi Antarmuka untuk Melihat Detail Kategori *Web Service* melalui Halaman Administrasi *Website*
Sumber : [Implementasi]

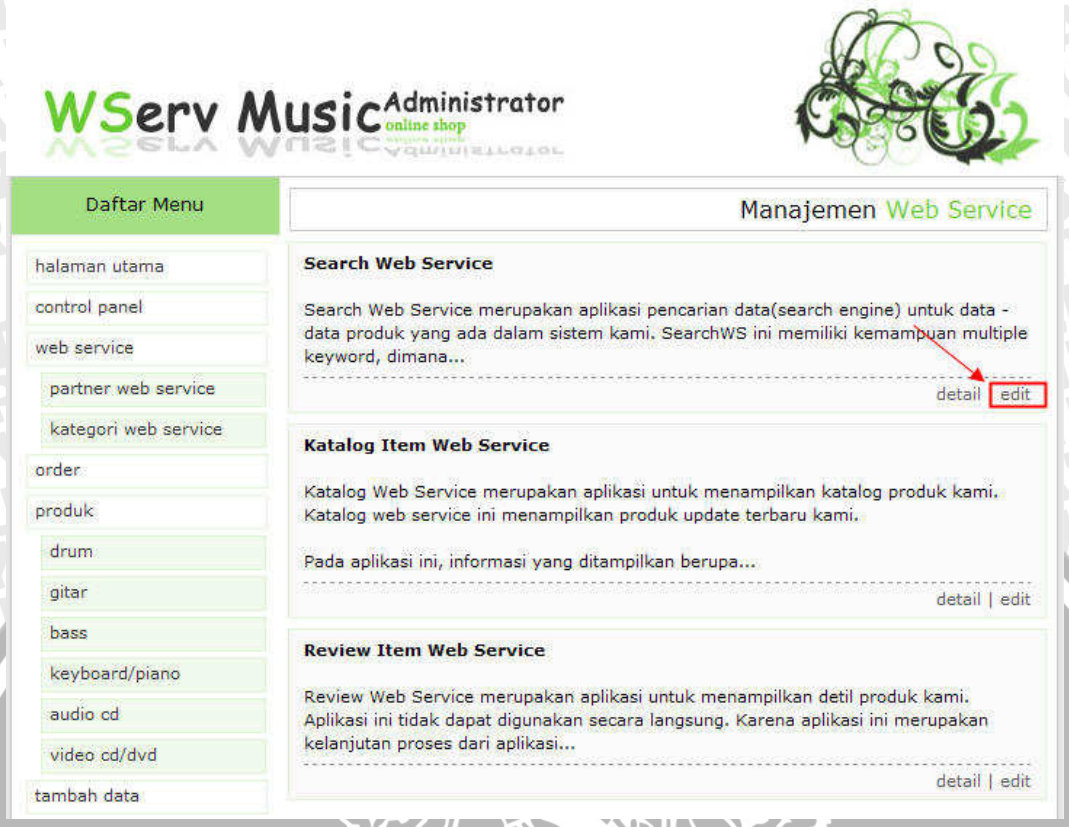
Implementasi algoritma untuk Melihat Detail Kategori *Web Service* ditunjukkan dalam tabel berikut :

Algoritma 5. 44 : Algoritma Melihat Detail Kategori *Web Service*

```
FUNCTION detKatWebService

VARIABLE htmlIndex = new FUNCTION htmlLayout
VARIABLE getHtmlIndex = VARIABLE htmlIndex SET FUNCTION htmlNonIndex BASED ON
'Web','Service'
VARIABLE dbWs = new FUNCTION dbWebService
VARIABLE id = VARIABLE _GET['Id']
VARIABLE detWS = VARIABLE dbWs SET FUNCTION detKatWebService BASED ON VARIABLE id
IF FUNCTION isset BASED ON VARIABLE _GET['admin'] THEN
  VARIABLE ls = VARIABLE detWS SET kategoriWebService
  VARIABLE ls.= VARIABLE detWS SET deskripsi
  VARIABLE dataBerita['isi'] = VARIABLE ls
ELSE
  VARIABLE ls = VARIABLE getHtmlIndex[0]
  VARIABLE ls.= VARIABLE detWS SET kategoriWebService
  VARIABLE ls.= VARIABLE detWS SET deskripsi
  VARIABLE ls.= VARIABLE getHtmlIndex[1]
  VARIABLE dataBerita['isi'] = VARIABLE ls
ENDIF

RETURN VARIABLE dataBerita
```

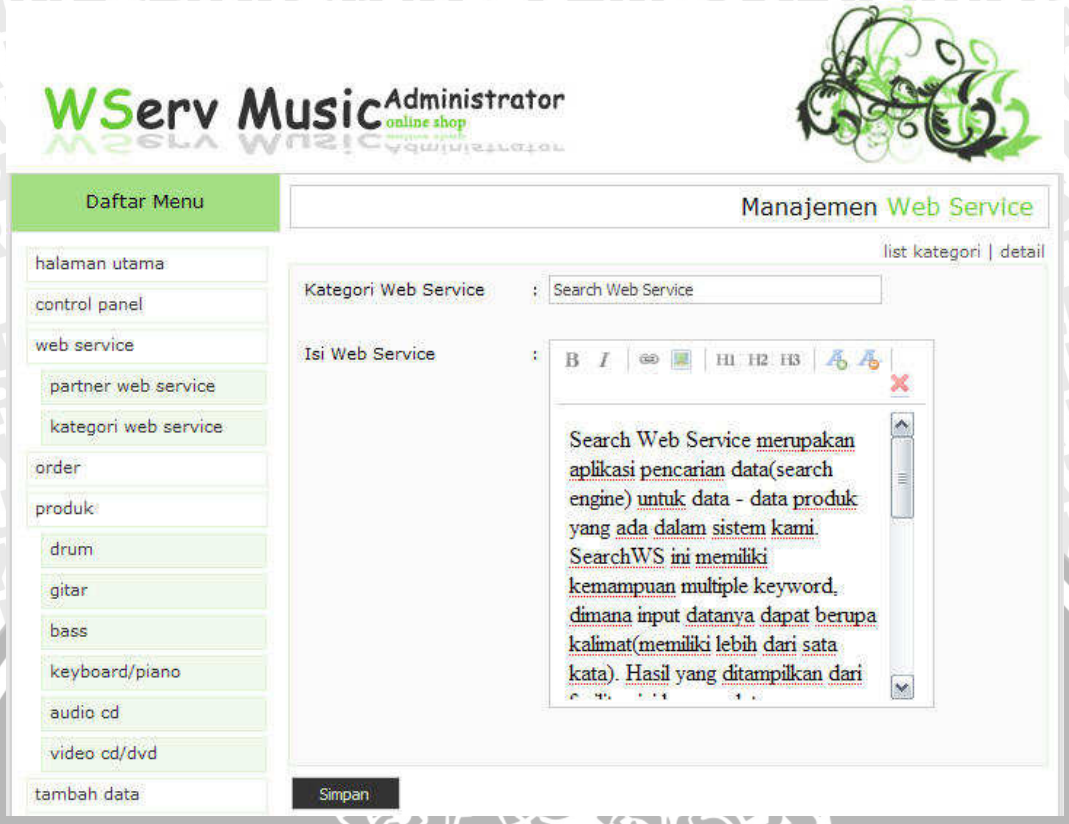
5.4.1.45. Implementasi Antarmuka Kebutuhan Fungsional Mengedit Kategori *Web Service*

Fasilitas mengedit kategori *web service* hanya dapat diakses oleh *administrator*. Fasilitas ini dapat diakses melalui halaman administrasi *website*. *Administrator* menekan tombol '`edit`' untuk membuka halaman *form* edit kategori *web service*. Gambar 5.95 menunjukkan implementasi antarmuka untuk mengedit kategori *web service*.



Gambar 5. 99 : Implementasi Antarmuka untuk Mengedit Kategori *Web Service*
Sumber : [Implementasi]

Setelah *administrator* menekan tombol '`edit`', sistem menampilkan *form* edit kategori *Web Service*. Gambar 5.96 menunjukkan hasil implementasi antarmuka untuk mengedit kategori *web service*.



Gambar 5. 100 : Hasil Implementasi Antarmuka untuk Mengedit Kategori *Web Service*
Sumber : [Implementasi]

Implementasi algoritma untuk Mengedit Kategori *Web Service* ditunjukkan dalam tabel berikut :

Algoritma 5. 45 : Algoritma Mengedit Kategori *Web Service*

```
FUNCTION cekFormEditKategoriWebService

IF FUNCTION isset BASED ON VARIABLE _POST['editWS'] THEN
  VARIABLE id=VARIABLE _GET['Id']
  VARIABLE fieldWS = FUNCTION array()
  foreach VARIABLE fieldWS as VARIABLE keyWS=>VARIABLE valWS
  BEGIN
    IF FUNCTION empty BASED ON VARIABLE _POST[VARIABLE keyWS] THEN
      VARIABLE error[VARIABLE keyWS] = VARIABLE valWS." web service harus diisi."
      VARIABLE post = ""
    ELSE
      VARIABLE error[VARIABLE keyWS] = " "
      VARIABLE post[VARIABLE keyWS] = VARIABLE _POST[VARIABLE keyWS]
    ENDIF
  END

  IF VARIABLE error['jdlWS'] = " " and VARIABLE error['isiWS'] =  " THEN
    VARIABLE this SET FUNCTION editKategoriWebService BASED ON VARIABLE
post,VARIABLE id
    VARIABLE msg = 'Data kategori web service berhasil disimpan'
  ELSE
    VARIABLE msg = ''
  ENDIF
```

```
  VARIABLE data = VARIABLE this SET FUNCTION formEditKategoriWebService BASED ON
VARIABLE error,VARIABLE msg
ENDIF

RETURN VARIABLE data
```

5.4.2. Implementasi Algoritma untuk SOAP Server

Pada subbab 5.4.2 ini membahas tentang implementasi algoritma untuk SOAP *Server*. Dalam proses *web service*, SOAP *server* membutuhkan *referensi* kelas diagram yang berisi modul – modul *web service*.

5.4.2.1. Implementasi Algoritma SOAP *Server*

SOAP *server* merupakan komponen *web service* yang melakukan *request* dan *response* terhadap dokumen WSDL. Implementasi SOAP *Server* sendiri menggunakan *library* SOAP yang terdapat pada PHP *Server*. Berikut ini merupakan Implementasi Algoritma SOAP *Server* untuk semua modul *web service*.

Algoritma 5. 46 : Algoritma SOAP *Server*

```
VARIABLE server = new FUNCTION  SoapServer BASED ON
“https://172.17.68.232/webservice-ecommerce/wsdl/allWs.wsdl”
VARIABLE server SET FUNCTION setClass BASED ON “wsdl”
VARIABLE server SET FUNCTION handle
```

5.4.2.2. Implementasi Algoritma Modul *Search* Produk

Modul *Search* Produk merupakan *action* dari Kelas `wsdl` yang berupa sintak PHP. Modul ini dapat digunakan oleh SOAP *Client* setelah SOAP *Server* mengaktifkannya. Berikut ini merupakan Implementasi Algoritma Modul *Search* Produk.

Algoritma 5. 47 : Algoritma Modul *Search* Produk

```
FUNCTION searchWs BASED ON VARIABLE query,VARIABLE kode

VARIABLE num = VARIABLE this SET dbWs SET FUNCTION numWebServicePartner BASED ON
VARIABLE kode,'','',''

IF FUNCTION isset BASED ON VARIABLE query THEN
  VARIABLE input = FUNCTION trim BASED ON VARIABLE query
  VARIABLE getInput = FUNCTION explode BASED ON " ",VARIABLE input
  VARIABLE gabung = ""
  VARIABLE jml = FUNCTION count BASED ON VARIABLE getInput

  for VARIABLE i=0;VARIABLE i<VARIABLE jml;VARIABLE i++
  BEGIN
    VARIABLE gabung.= FUNCTION ucfirst BASED ON VARIABLE getInput[VARIABLE i])." "
  END
```

```
  VARIABLE getInput2 = FUNCTION explode BASED ON " ",VARIABLE gabung
  VARIABLE combo = FUNCTION array_merge BASED ON VARIABLE getInput,VARIABLE
getInput2
  VARIABLE sql = "select *from barang where "

  for VARIABLE i=0;VARIABLE i<VARIABLE jml;VARIABLE i++
  BEGIN
    VARIABLE sql.= "namaBarang like '%".VARIABLE getInput[VARIABLE i]."%' or
deskripsiBarang like '%".VARIABLE getInput[VARIABLE i]."%'"
    IF VARIABLE i<VARIABLE jml THEN
      for VARIABLE i=1;VARIABLE i<VARIABLE jml;VARIABLE i++
      BEGIN
        VARIABLE sql.=" or namaBarang like '%".VARIABLE getInput[VARIABLE i]."%'
or deskripsiBarang like '%".VARIABLE getInput[VARIABLE i]."%'"
      END
    ENDIF
  END

  VARIABLE rep = ""
  for VARIABLE i=0;VARIABLE i<FUNCTION count BASED ON VARIABLE combo;VARIABLE i++
  BEGIN
    VARIABLE rep.= VARIABLE combo[VARIABLE i]
  END

  VARIABLE get = FUNCTION explode BASED ON " ",VARIABLE rep
  VARIABLE sql.= " limit 0,5"
  VARIABLE qry = VARIABLE this SET db SET FUNCTION getQuery BASED ON VARIABLE sql
  WHILE VARIABLE row = FUNCTION mysql_fetch_array BASED ON VARIABLE qry
    VARIABLE produk = VARIABLE row['namaBarang']
    VARIABLE desc = VARIABLE row['deskripsiBarang']
    VARIABLE namaPro = FUNCTION str_replace BASED ON VARIABLE combo,VARIABLE
get,VARIABLE produk
    VARIABLE desPro = FUNCTION str_replace BASED ON VARIABLE combo,VARIABLE
get,VARIABLE desc
    VARIABLE pro['search'][VARIABLE row['idBarang']]['id'] = VARIABLE
row['idBarang']
    VARIABLE pro['search'][VARIABLE row['idBarang']]['produk'] = VARIABLE namaPro
    VARIABLE pro['search'][VARIABLE row['idBarang']]['desc'] = VARIABLE this SET
cut SET FUNCTION cut BASED ON VARIABLE desPro,25

    IF FUNCTION empty BASED ON VARIABLE kode or VARIABLE num!=1 THEN
      VARIABLE msg = FUNCTION arrayBASED ON 'produk'=>'Masukkan kode dengan
benar.','desc'=>''
      VARIABLE ws['search'][] = VARIABLE msg
    ELSE
      VARIABLE ws = VARIABLE pro
    ENDIF
  ENDWHILE
ENDIF

RETURN VARIABLE ws
```

5.4.2.3. Implementasi Algoritma Modul Katalog Produk

Modul Katalog Produk merupakan *action* dari Kelas `wsdl` yang berupa sintak PHP. Modul ini dapat digunakan oleh SOAP *Client* setelah SOAP *Server* mengaktifkannya. Berikut ini merupakan Implementasi Algoritma Modul Katalog

Produk.

Algoritma 5. 48 : Algoritma Modul Katalog Produk

```
FUNCTION katalogWs BASED ON VARIABLE kode

VARIABLE list = VARIABLE this SET dbBarang SET FUNCTION listBarang BASED ON 0,3
VARIABLE num = VARIABLE this SET dbWs SET FUNCTION numWebServicePartner BASED ON
VARIABLE kode,'','',''

foreach VARIABLE list as VARIABLE katalog
BEGIN
  VARIABLE gb = "http://172.17.68.232/webservice-
ecommerce/gambar/gbBarang/".VARIABLE katalog SET gambarBarang
  VARIABLE katBarang['katalog'][VARIABLE katalog SET idBarang]['id'] = VARIABLE
katalog SET idBarang
  VARIABLE katBarang['katalog'][VARIABLE katalog SET idBarang]['produk'] =
VARIABLE katalog SET namaBarang
  VARIABLE katBarang['katalog'][VARIABLE katalog SET idBarang]['harga'] = VARIABLE
katalog SET hargaBarang
  VARIABLE katBarang['katalog'][VARIABLE katalog SET idBarang]['stok'] = VARIABLE
katalog SET stokBarang
  VARIABLE katBarang['katalog'][VARIABLE katalog SET idBarang]['gambar'] =
VARIABLE gb

  IF FUNCTION empty BASED ON VARIABLE kode or VARIABLE num!=1 THEN
    VARIABLE msg = FUNCTION array BASED ON 'id'=>'','produk'=>'Masukkan kode
dengan benar.','harga'=>'','stok'=>'','gambar'=>''
    VARIABLE ws['katalog'] = VARIABLE msg
  ELSE
    VARIABLE ws = VARIABLE katBarang
  ENDIF
END

RETURN VARIABLE ws
```

5.4.2.4. Implementasi Algoritma Modul *Review* Produk

Modul *Review* Produk merupakan *action* dari Kelas `wsdl` yang berupa sintak PHP. Modul ini dapat digunakan oleh SOAP *Client* setelah SOAP *Server* mengaktifkannya. Berikut ini merupakan Implementasi Algoritma Modul *Review* Produk.

Algoritma 5. 49 : Algoritma Modul *Review* Produk

```
FUNCTION reviewWs BASED ON VARIABLE idBarang,VARIABLE kode

VARIABLE det = VARIABLE this SET dbBarang SET FUNCTION detBarang BASED ON VARIABLE
idBarang
VARIABLE num = VARIABLE this SET dbWs SET FUNCTION numWebServicePartner BASED ON
VARIABLE kode,'','',''
VARIABLE re['id'] = VARIABLE det SET idBarang
VARIABLE re['produk'] = VARIABLE det SET namaBarang
VARIABLE re['desc'] = VARIABLE det SET deskripsiBarang
VARIABLE re['harga'] = VARIABLE det SET hargaBarang
VARIABLE re['gambar'] = VARIABLE det SET gambarBarang
IF FUNCTION empty BASED ON VARIABLE kode or VARIABLE num!=1 THEN
  VARIABLE msg = FUNCTION array BASED ON 'id'=>'','produk'=>'Masukkan kode dengan
benar.','harga'=>'','desc'=>'','gambar'=>''
  VARIABLE ws = VARIABLE msg
```

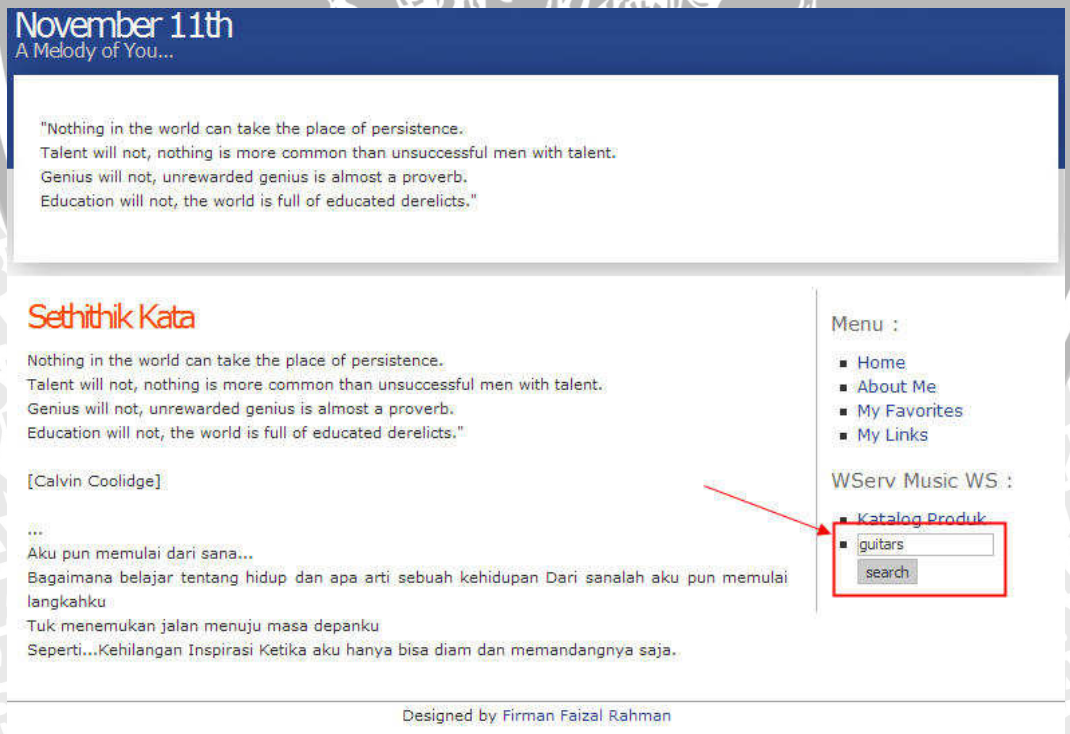
```
ELSE
  VARIABLE ws=VARIABLE re;
ENDIF

RETURN VARIABLE ws
```

5.4.3. Implementasi Antarmuka dan Algoritma untuk SOAP Client

Pada subbab 5.4.3 ini membahas tentang implementasi antarmuka dan algoritma untuk SOAP *Client*. SOAP *Client* ini merupakan bagian dari *requester entity* yang berada pada Aplikasi *Partner*. Berikut contoh implementasi SOAP *Client* pada aplikasi *partner*.

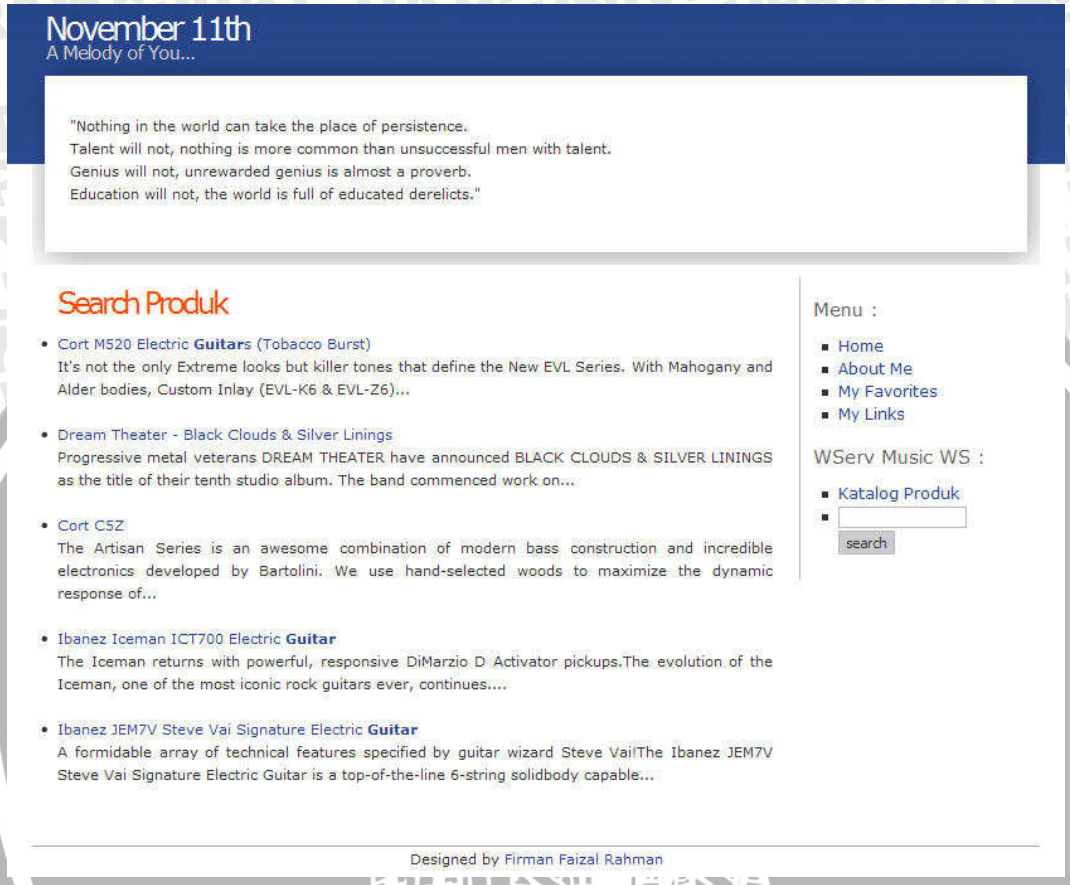
5.4.3.1. Implementasi Antarmuka dan Algoritma SOAP *Client* untuk *Search* Produk

Pada aplikasi *partner*, SOAP *Client* digunakan untuk mengaktifkan modul *web service* sehingga dapat ditampilkan melalui *browser*. Untuk menggunakan modul *Search* Produk, aplikasi *partner* harus memiliki *form search* dan hasil *searching* produk. Implementasi Antarmuka dan Algoritma SOAP *Client* untuk *Search* Produk ditunjukkan pada gambar 5.97.



Gambar 5. 101 : Implementasi Antarmuka SOAP *Client* untuk *Search* Produk
Sumber : [Implementasi]

Setelah *form search* diisi dengan data yang akan dicari, maka aplikasi *partner* menampilkan daftar produk yang memiliki unsur kata dari data yang dicari tersebut. Hasil Implementasi Antarmuka dan Algoritma SOAP *Client* untuk *Search* Produk ditunjukkan pada gambar 5.98.



Gambar 5. 102 : Hasil Implementasi Antarmuka SOAP *Client* untuk *Search* Produk
Sumber : [Implementasi]

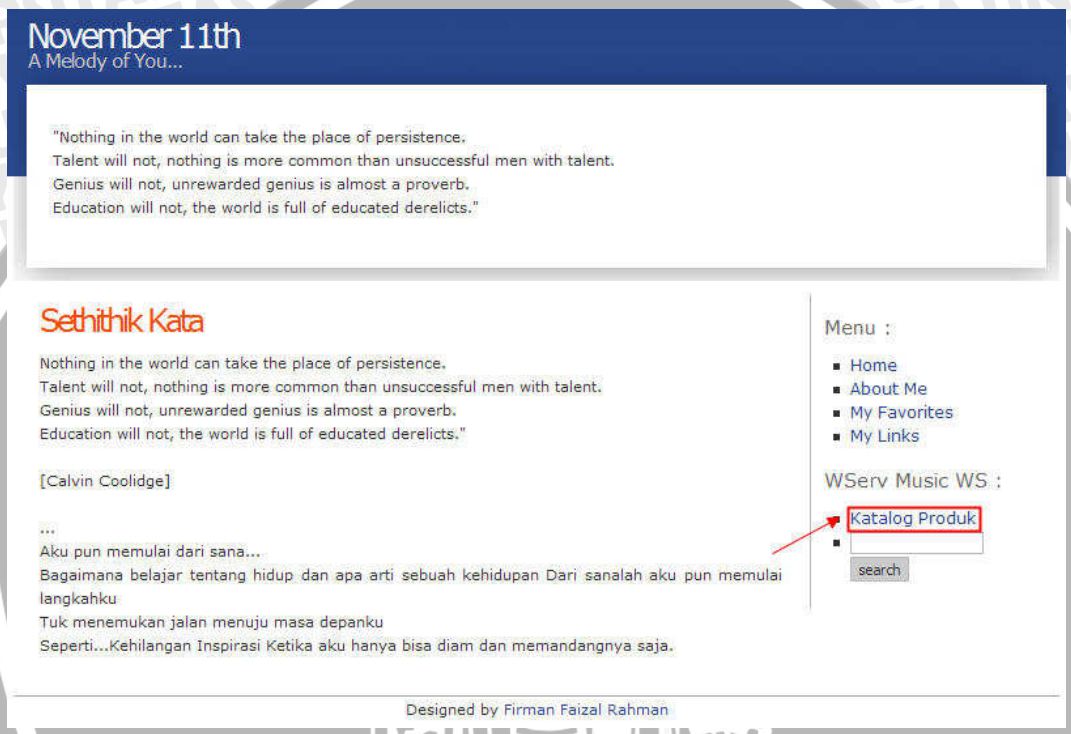
Algoritma Implementasi Antarmuka dan Algoritma SOAP *Client* untuk *Search* Produk ditunjukkan sebagai berikut :

Algoritma 5. 50 : Algoritma SOAP *Client* untuk *Search* Produk

```
IF FUNCTION isset BASED ON VARIABLE _GET['cari'] THEN
  VARIABLE client = new FUNCTION SoapClient BASED ON
'https://172.17.68.232/webservice-ecommerce/wsdl/allWs.wsdl'
  VARIABLE search = VARIABLE client SET FUNCTION searchWs BASED ON VARIABLE
_GET['cari'],_aut

  foreach VARIABLE search['search'] as VARIABLE cari
  BEGIN
    echo VARIABLE cari['produk']
    echo FUNCTION strip_tags BASED ON VARIABLE cari['desc']
  END
ENDIF
```

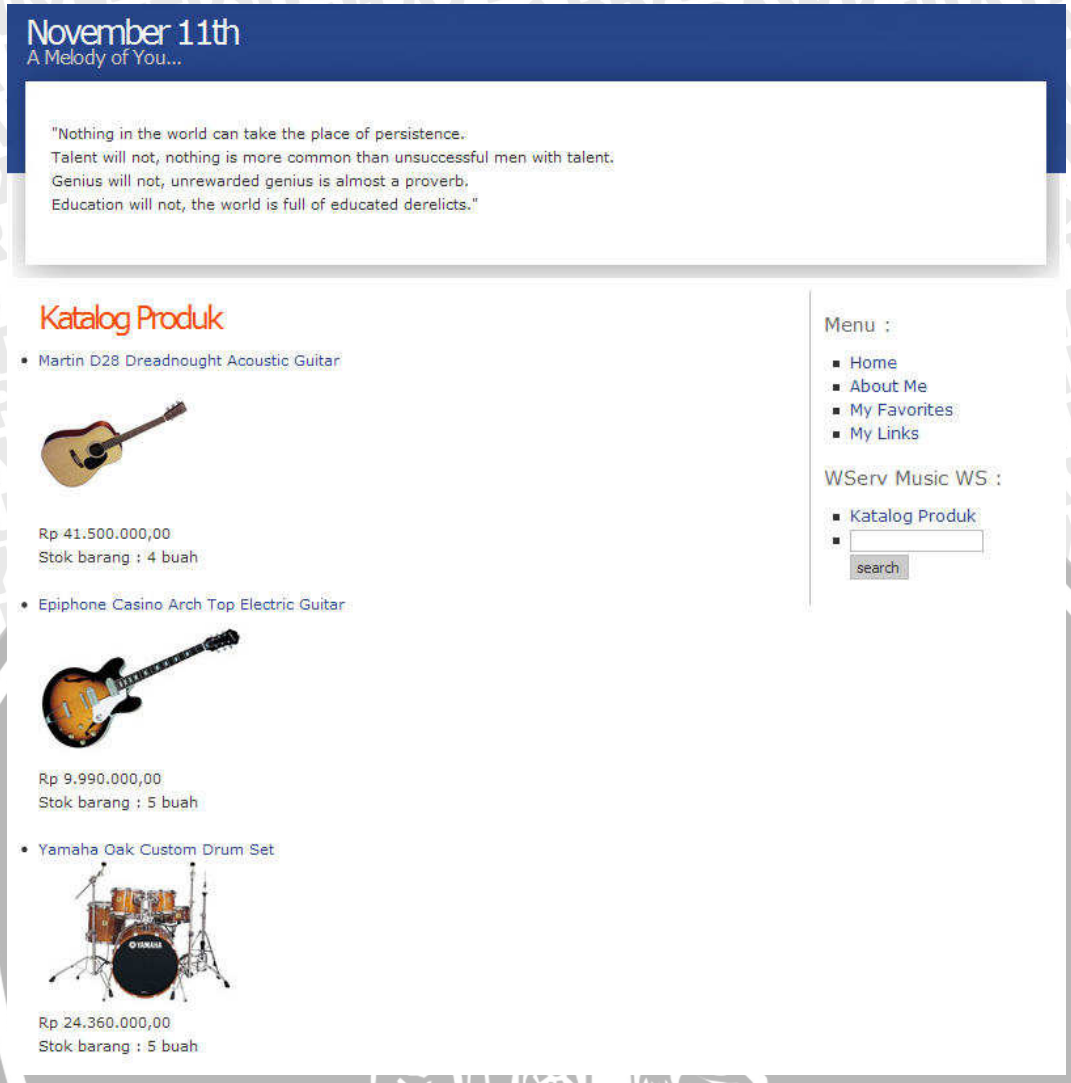
5.4.3.2. Implementasi Antarmuka dan Algoritma SOAP *Client* untuk Katalog Produk

Modul Katalog Produk merupakan fasilitas *web service* untuk menampilkan produk terbaru pada Aplikasi *Web Service E-Commerce*. Modul ini digunakan oleh Aplikasi *Partner* dengan mengaktifkan SOAP *Client*. Implementasi Antarmuka dan Algoritma SOAP *Client* untuk Katalog Produk ditunjukkan pada gambar 5.99.



Gambar 5. 103 : Implementasi Antarmuka SOAP *Client* untuk Katalog Produk
Sumber : [Implementasi]

Pada contoh Aplikasi *Partner* tersebut, *user* harus memilih menu "Katalog Produk" untuk menampilkan daftar produk terbaru. Hasil Implementasi Antarmuka dan Algoritma SOAP *Client* untuk Katalog Produk ditunjukkan pada gambar 5.100.



Gambar 5. 104 : Hasil Implementasi Antarmuka SOAP *Client* untuk Katalog Produk
Sumber : [Implementasi]

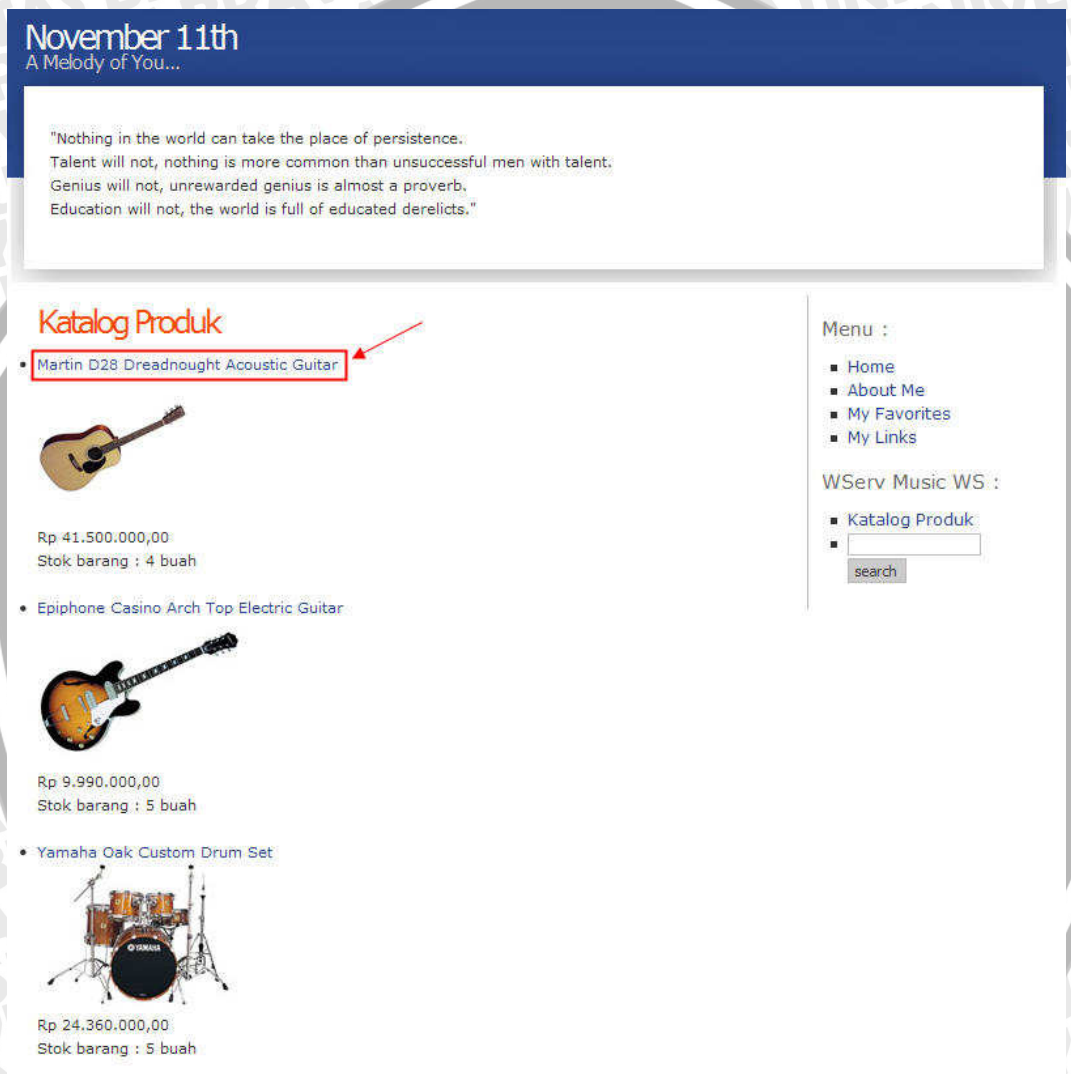
Algoritma Implementasi Antarmuka dan Algoritma SOAP *Client* untuk Katalog Produk ditunjukkan sebagai berikut :

Algoritma 5. 51 : Algoritma SOAP *Client* untuk Katalog Produk

```
VARIABLE client = new FUNCTION SoapClient BASED ON
'https://172.17.68.232/webservice-ecommerce/wsdl/allWs.wsdl'
VARIABLE data = VARIABLE client SET FUNCTION katalogWs BASED ON _aut

foreach VARIABLE data['katalog'] as VARIABLE x
BEGIN
  echo VARIABLE x['produk']
  echo VARIABLE x['gambar']
  echo "Rp ".FUNCTION number_format BASED ON VARIABLE x['harga'],2,',','.'
  echo VARIABLE x['stok']
END
```

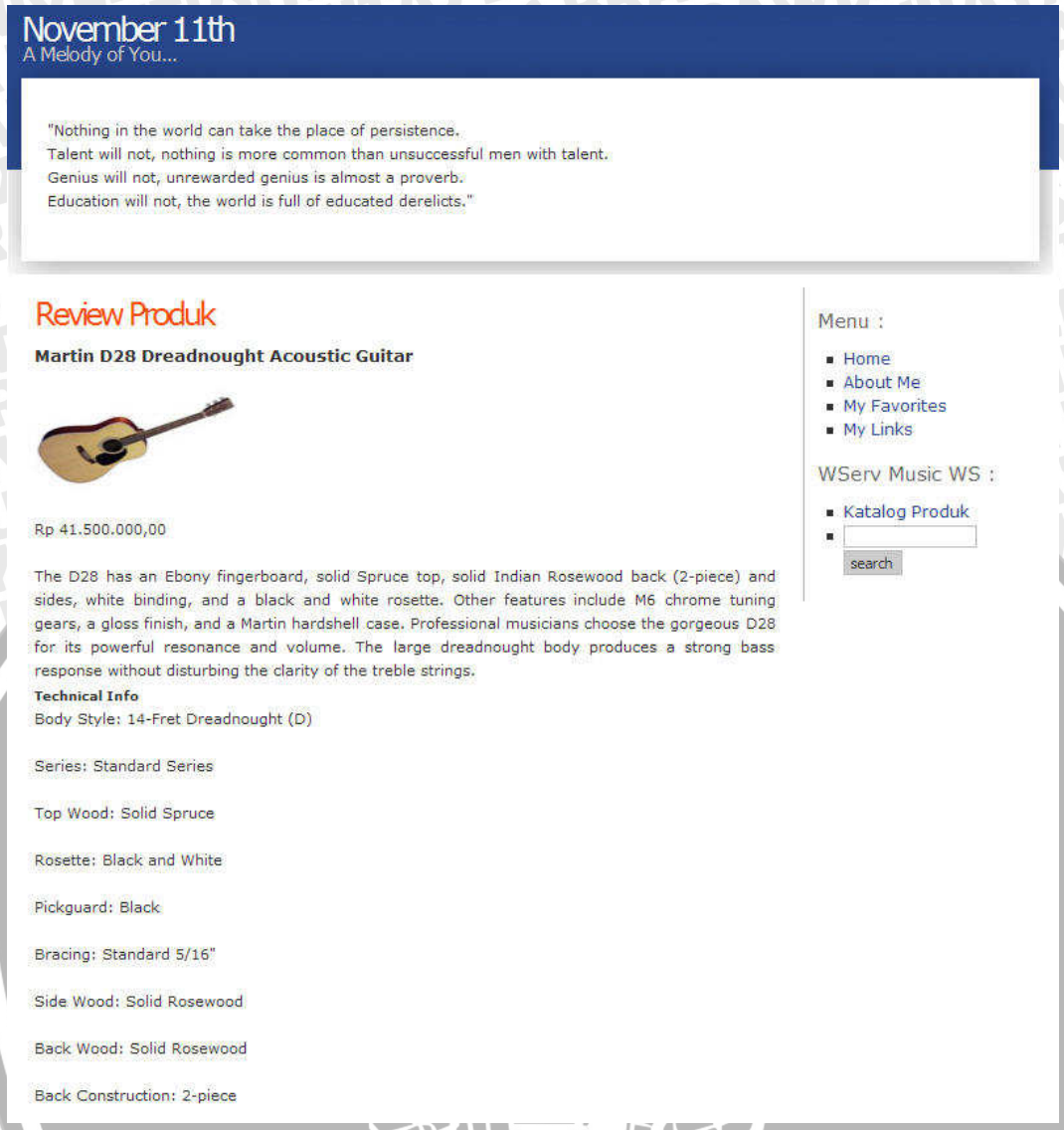
5.4.3.3. Implementasi Algoritma Antarmuka dan Algoritma SOAP *Client* untuk *Review* Produk

Modul *review* produk merupakan fasilitas modul *web service* yang tidak bisa berdiri sendiri. Modul ini membutuhkan modul katalog produk atau modul *search* produk. Implementasi Antarmuka dan Algoritma SOAP *Client* untuk *Review* Produk ditunjukkan pada gambar 5.101.



Gambar 5. 105 : Implementasi Antarmuka SOAP *Client* untuk *Review* Produk
Sumber : [Implementasi]

Modul *review* produk pada gambar 5.101 diproses melalui katalog produk. Hasil Implementasi Antarmuka dan Algoritma SOAP *Client* untuk *Search* Produk ditunjukkan pada gambar 5.102.



Gambar 5. 106 : Hasil Implementasi Antarmuka SOAP *Client* untuk *Review* Produk
Sumber : [Implementasi]

Algoritma Implementasi Antarmuka dan Algoritma SOAP *Client* untuk *Review* Produk ditunjukkan sebagai berikut :

Algoritma 5. 52 : Algoritma SOAP *Client* untuk *Review* Produk

```
VARIABLE client = new FUNCTION SoapClient BASED ON
'https://172.17.68.232/webservice-ecommerce/wsdl/allWs.wsdl'
IF FUNCTION isset BASED ON VARIABLE _GET['id'] THEN
  VARIABLE det = VARIABLE client SET reviewWs BASED ON VARIABLE _GET['id'],_aut
  echo VARIABLE det['produk']
  echo VARIABLE det['gambar']
  echo "Rp ".FUNCTION number_format BASED ON VARIABLE det['harga'],2,',','.'
  echo VARIABLE det['desc']
ENDIF
```

5.4.4. Implementasi WSDL

Pada subbab 5.4.4 ini membahas tentang implementasi algoritma untuk WSDL yang terdapat pada Aplikasi *Web Service E-Commerce*. WSDL memiliki format XML. Berdasarkan subbab 4.2.2.4.3, WSDL memiliki struktur utama, yakni *definition*, *message*, *portType*, *binding*, dan *service*.

5.4.4.1. Implementasi WSDL pada Stuktur *definition*

Struktur *definition* merupakan bagian WSDL yang berfungsi sebagai *root*. Struktur ini merupakan penyusun utama pada dokumen WSDL. Berikut ini Implementasi WSDL pada Struktur *definition.*

```
<definitions name='wsdl'
 targetNamespace='https://172.17.68.232/webservice-
ecommerce/wsdl/allWs.wsdl'
 xmlns:tns='https://172.17.68.232/webservice-ecommerce/wsdl/allWs.wsdl'
 xmlns:soap='http://schemas.xmlsoap.org/wsdl/soap/'
 xmlns='http://schemas.xmlsoap.org/wsdl/'>
...
</definitions>
```

5.4.4.2. Implementasi WSDL pada Stuktur *message*

Struktur *message* merupakan bagian WSDL yang berfungsi sebagai pesan yang digunakan untuk *web service*. Struktur ini terdiri atas pesan *request* dan pesan *response*. Berikut ini Implementasi WSDL pada Struktur *message*.

```
<message name='searchWsRequest'>
      <part name='name' type='xsd:string'/>
      <part name='kode' type='xsd:string'/>
</message>
<message name='searchWsResponse'>
      <part name='result' type='xsd:string'/>
</message>
<message name='katalogWsRequest'>
      <part name='name' type='xsd:string'/>
</message>
<message name='katalogWsResponse'>
      <part name='result' type='xsd:string'/>
</message>
<message name='reviewWsRequest'>
      <part name='name' type='xsd:string'/>
      <part name='kode' type='xsd:string'/>
</message>
<message name='reviewWsResponse'>
      <part name='result' type='xsd:string'/>
</message>
```

5.4.4.3. Implementasi WSDL pada Stuktur *portType*

Struktur *portType* merupakan bagian WSDL yang berfungsi sebagai operasi *web service*. Struktur ini mendeklarasikan *input* dan *output* dari modul yang digunakan untuk *web service*. Berikut ini Implementasi WSDL pada Struktur *portType*.

```
<portType name='wsdlPortType'>
      <operation name='searchWs'>
            <input message='tns:searchWsRequest'/>
            <output message='tns:searchWsResponse'/>
      </operation>
      <operation name='katalogWs'>
            <input message='tns:katalogWsRequest'/>
            <output message='tns:katalogWsResponse'/>
      </operation>
      <operation name='reviewWs'>
            <input message='tns:reviewWsRequest'/>
            <output message='tns:reviewWsResponse'/>
      </operation>
</portType>
```

5.4.4.4. Implementasi WSDL pada Stuktur *binding*

Struktur *binding* merupakan bagian WSDL yang berfungsi sebagai protokol yang digunakan *web service*. Struktur ini berinteraksi dengan SOAP *server* dan SOAP *client*. Berikut ini Implementasi WSDL pada Struktur *binding*.

```
<binding name='wsdlBinding' type='tns:wsdlPortType'>
  <soap:binding style='rpc'
transport='http://schemas.xmlsoap.org/soap/http'/>
  <operation name='searchWs'>
    <soap:operation soapAction='urn:xmethods-delayed-quotes#searchWs'/>
    <input>
      <soap:body use='encoded' namespace='urn:xmethods-delayed-quotes'
encodingStyle='http://schemas.xmlsoap.org/soap/encoding/'/>
    </input>
    <output>
      <soap:body use='encoded' namespace='urn:xmethods-delayed-quotes'
encodingStyle='http://schemas.xmlsoap.org/soap/encoding/'/>
    </output>
  </operation>
  <operation name='katalogWs'>
    <soap:operation soapAction='urn:xmethods-delayed-quotes#katalogWs'/>
    <input>
      <soap:body use='encoded' namespace='urn:xmethods-delayed-quotes'
encodingStyle='http://schemas.xmlsoap.org/soap/encoding/'/>
    </input>
    <output>
      <soap:body use='encoded' namespace='urn:xmethods-delayed-quotes'
encodingStyle='http://schemas.xmlsoap.org/soap/encoding/'/>
    </output>
```

```
  </operation>
  <operation name='reviewWs'>
    <soap:operation soapAction='urn:xmethods-delayed-quotes#reviewWs'/>
    <input>
    <soap:body use='encoded' namespace='urn:xmethods-delayed-quotes'
encodingStyle='http://schemas.xmlsoap.org/soap/encoding/'/>
    </input>
    <output>
      <soap:body use='encoded' namespace='urn:xmethods-delayed-quotes'
encodingStyle='http://schemas.xmlsoap.org/soap/encoding/'/>
    </output>
  </operation>
</binding>
```

5.4.4.5. Implementasi WSDL pada Stuktur *service*

Struktur *service* merupakan bagian WSDL yang berfungsi untuk mengaktifkan SOAP *server*. Struktur ini menunjukkan alamat dimana terjadinya proses *web service*. Berikut ini Implementasi WSDL pada Struktur *service*.

```
<service name='wsdlService'>
  <port name='wsdlPort' binding='wsdlBinding'>
    <soap:address location='https://172.17.68.232/webservice-
ecommerce/soap/allSoapServer.php'/>
  </port>
</service>
```

BAB VI
PENGUJIAN

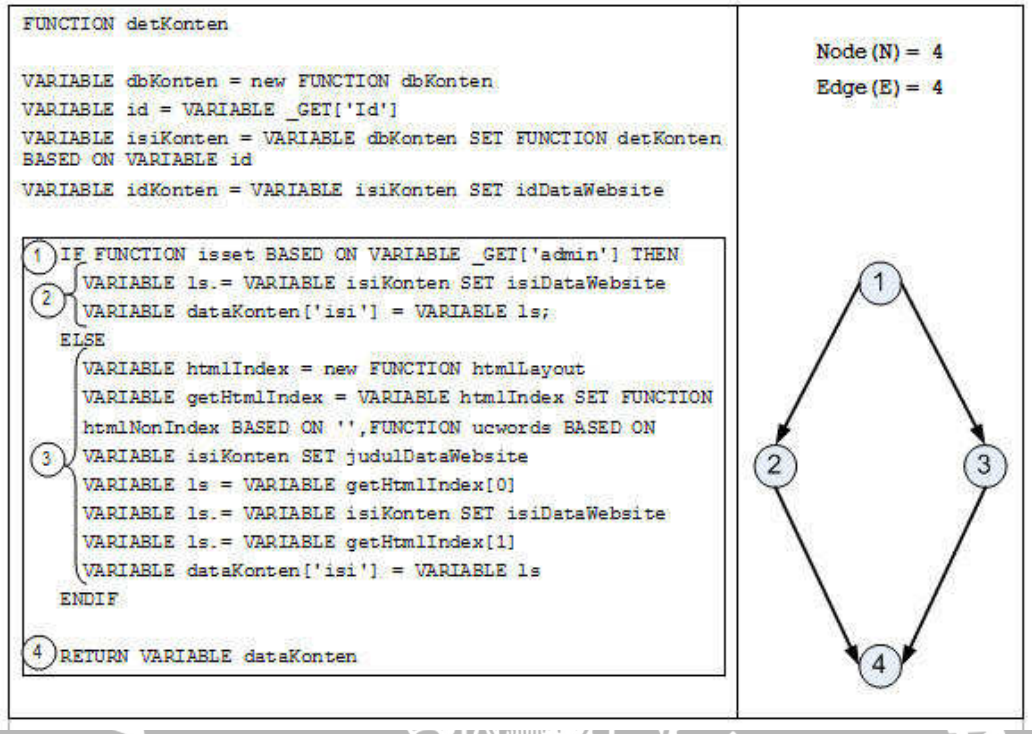
Bab ini membahas proses pengujian terhadap Aplikasi *Web Service E-Commerce* yang telah dibangun. Proses pengujian dilakukan melalui lima tahapan (strategi) yaitu pengujian unit, pengujian integrasi, pengujian validasi, pengujian perancangan basis data, dan pengujian performansi koneksi *web service*. Pengujian unit dan pengujian integrasi menggunakan teknik pengujian *white box (white box Testing)*. Pengujian validasi menggunakan teknik pengujian *black box (black box Testing)*.

6.1. Pengujian Unit

Sistem yang dibangun dengan paradigma pemrograman berorientasi objek menerapkan pengujian unit untuk suatu metode (operasi) dari suatu klas. Pada pengujian unit Aplikasi *Web Service E-Commerce* ini, digunakan teknik pengujian *White Box (White Box Testing)* dengan teknik *Basis Path Testing*. Pada teknik *Basis Path Testing*, proses pengujian dilakukan dengan memodelkan algoritma pada suatu *flow graph*, menentukan jumlah kompleksitas siklomatis *(cyclomatic complexity)*, menentukan sebuah basis set dari jalur independen dan memberikan kasus uji *(test case)* pada setiap basis set yang telah ditentukan. Penulisan laporan skripsi ini hanya dicantumkan hasil pengujian unit untuk algoritma dari beberapa metode (operasi) saja (tidak untuk keseluruhan metode).

6.1.1. Pengujian Unit untuk *Method* `detKonten()` `klas` `Konten`

Method `detKonten()` ini dipergunakan untuk menampilkan data detail konten *website*. Gambar 6.1 memperlihatkan proses pemodelan dalam *flow graph* pada *method* `detKonten()`.



Gambar 6. 1 : Pemodelan operasi `detKonten()` ke dalam *flow graph*
Sumber : [Pengujian]

Pemodelan ke dalam *flow graph* yang telah dilakukan terhadap *method* `detKonten` menghasilkan jumlah kompleksitas siklomatis *(cyclomatic complexity)* melalui persamaan V(G) = E – N + 2, dimana V(G) merupakan jumlah kompleksitas siklomatis, E merupakan sisi (garis penghubung antar *node*) dan N merupakan jumlah simpul *(node)*.

V(G) = E – N + 2
= 4 – 4 + 2
= 2

Dari nilai *cyclomatic complexity* yang telah dihasilkan dari perhitungan yaitu 1 ditentukan sebuah basis set dari jalur independen yaitu:

Jalur 1 : 1-2-4

Jalur 2 : 1-3-4

Penentuan kasus uji untuk masing-masing jalur dan hasil eksekusi untuk kasus uji dijelaskan pada Tabel 6.1.

Tabel 6. 1 : *Test case* untuk pengujian integrasi *method* `detKonten()`

Jalur	Kasus uji	Hasil yang diharapkan	Hasil yang didapatkan
1.	Sistem memproses halaman admin.	Sebuah data array berisi detail konten.	Sebuah data array berisi detail konten.
2.	Sistem memproses selain halaman admin	Sebuah data array berisi detail konten.	Sebuah data array berisi detail konten.

Sumber: [Pengujian]

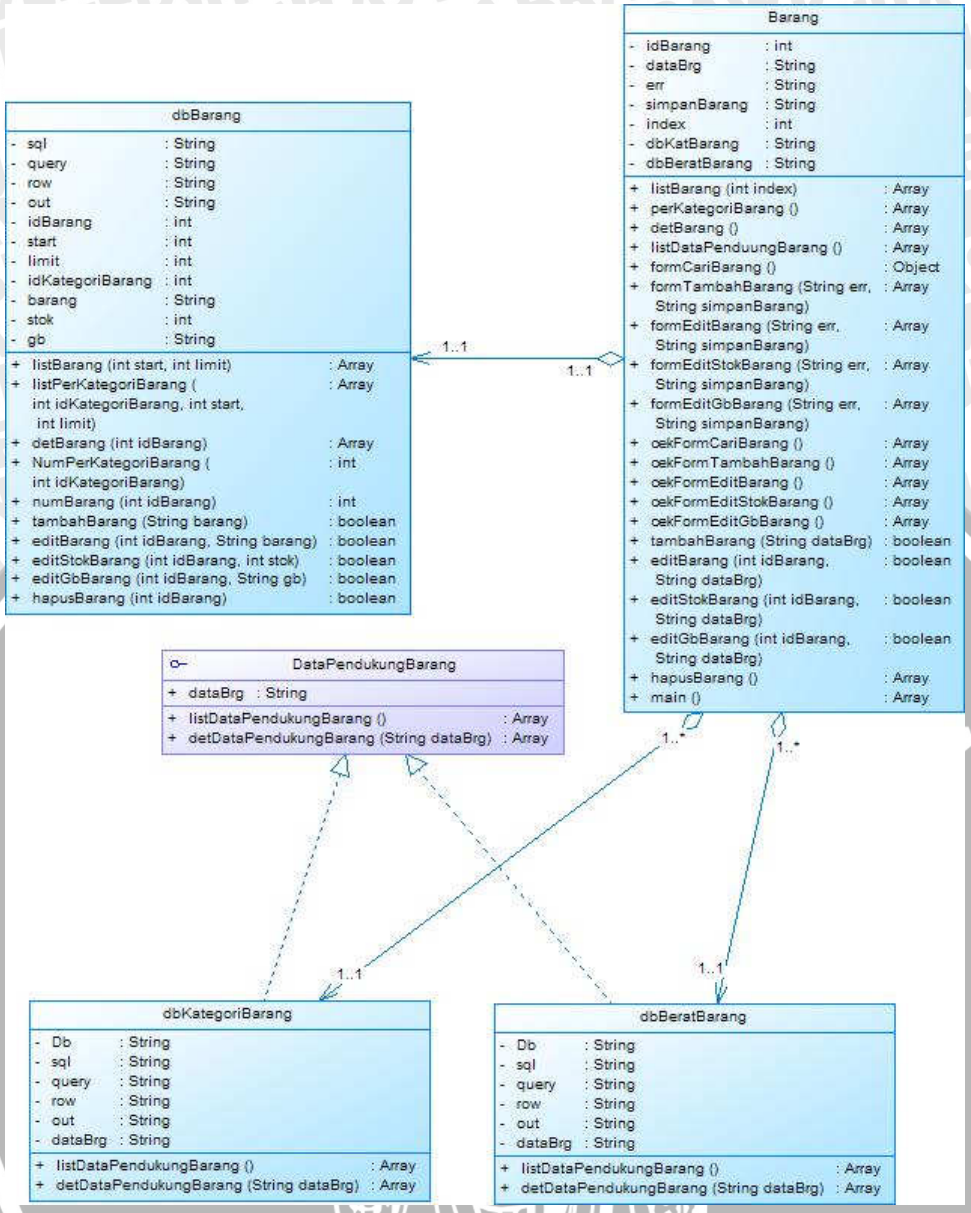
6.2. Pengujian Integrasi

Pengujian integrasi diterapkan pada proses yang mengintegrasikan fungsionalitas dari beberapa klas untuk melakukan sebuah operasi tertentu. Pada pengujian integrasi yang dijadikan sebagai obyek uji adalah klas - klas yang menggabungkan kinerja dari klas - klas yang lain. Klas - klas yang mengintegrasikan beberapa klas yang lain tersebut akan berperan juga sebagai *test driver* yang mengendalikan proses pengujian sehingga bisa memperlihatkan status valid atau tidaknya hasil integrasi. Dalam melakukan pengujian integrasi terhadap sistem perangkat lunak ini digunakan strategi *bottom-up*.

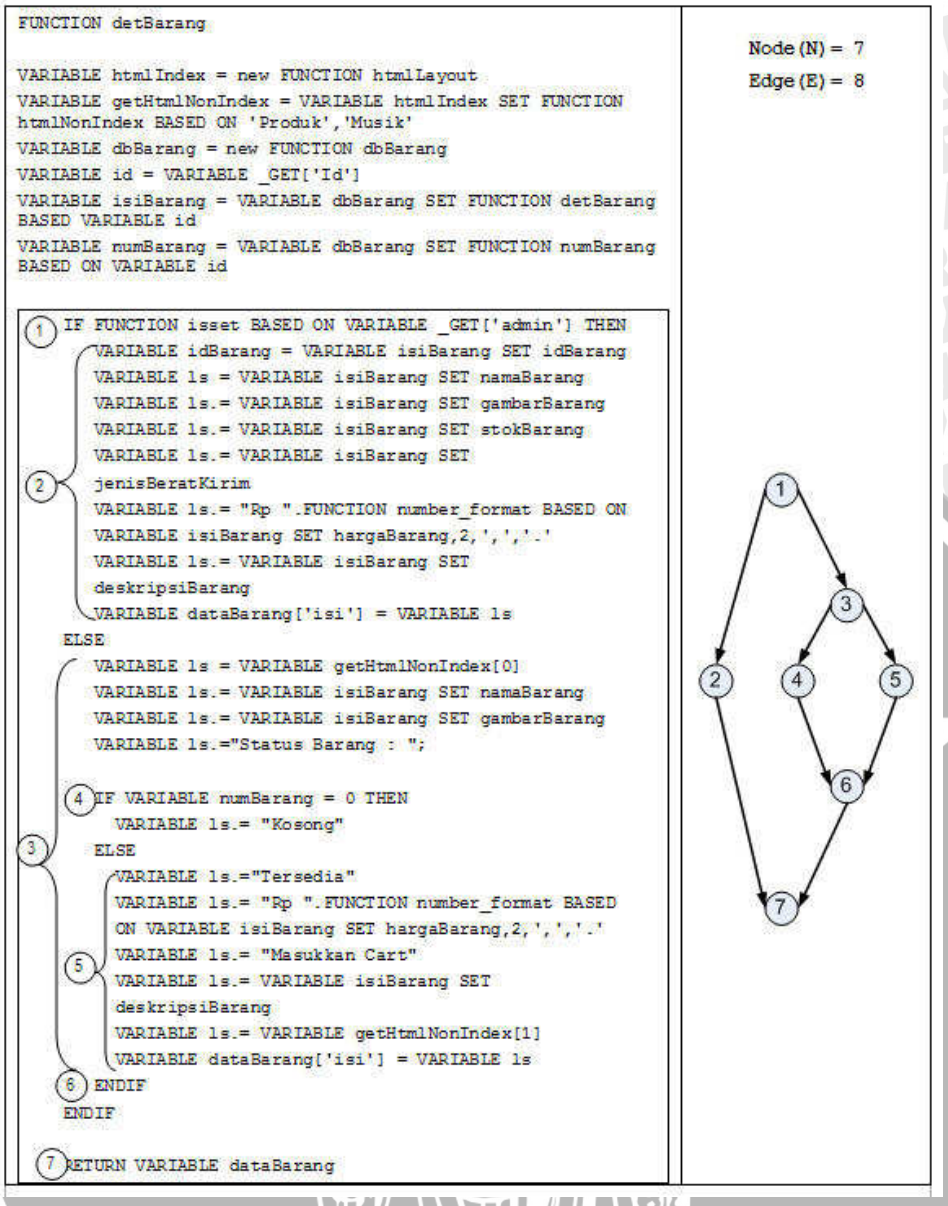
Pada pengujian integrasi aplikasi *Web Service E-Commerce* ini, digunakan teknik pengujian *White Box (White Box Testing)* dengan teknik *Basis Path Testing*. Pada teknik *Basis Path Testing*, proses pengujian dilakukan dengan memodelkan algoritma pada suatu *flow graph*, menentukan jumlah kompleksitas siklomatis *(cyclomatic complexity)*, menentukan sebuah basis set dari jalur independen dan memberikan kasus uji *(test case)* pada setiap basis set yang telah ditentukan. Penulisan laporan skripsi ini hanya dicantumkan hasil pengujian integrasi untuk algoritma dari beberapa metode (operasi) saja (tidak untuk keseluruhan metode).

6.2.1. Pengujian Integrasi untuk *Method* `detBarang()` **`klas`** `Barang`

Method `detBarang()` ini dipergunakan untuk menampilkan data detail produk. *Method* `detBarang()` terdapat pada klas `Barang`, dimana klas `Barang` memiliki relasi dengan klas `dbBarang`, klas `dbKategoriBarang`, klas `dbBeratBarang`, dan *interface* `DataPendukungBarang`. Gambar 6.2 menunjukkan relasi klas – klas tersebut, dan gambar 6.3 memperlihatkan proses pemodelan dalam *flow graph* pada *method* tersebut.



Gambar 6. 2 : Relasi antara klas `dbBarang`, klas `dbKategoriBarang`, klas `dbBeratBarang`, dan *interface* `DataPendukungBarang`
Sumber : [Pengujian]



Gambar 6. 3 : Pemodelan operasi `detBarang()` ke dalam *flow graph*
Sumber : [Pengujian]

Pemodelan ke dalam *flow graph* yang telah dilakukan terhadap *method* `detBarang()` menghasilkan jumlah kompleksitas siklomatis *(cyclomatic complexity)* melalui persamaan V(G) = E – N + 2, dimana V(G) merupakan jumlah kompleksitas siklomatis, E merupakan sisi (garis penghubung antar *node*) dan N merupakan jumlah simpul *(node)*.

V(G) = E – N + 2
= 8 – 7 + 2
= 3

Dari nilai *cyclomatic complexity* yang telah dihasilkan dari perhitungan yaitu 1 ditentukan sebuah basis set dari jalur independen yaitu:

Jalur 1 : 1-2-7

Jalur 2 : 1-3-4-6-7

Jalur 3 : 1-3-5-6-7

Penentuan kasus uji untuk masing-masing jalur dan hasil eksekusi untuk kasus uji dijelaskan pada Tabel 6.2.

Tabel 6. 2 : *Test case* untuk pengujian integrasi *method* `detBarang()`

Jalur	Kasus uji	Hasil yang diharapkan	Hasil yang didapatkan
1.	Sistem memproses halaman admin	Sebuah data array berisi detail barang.	Sebuah data array berisi detail barang.
2.	Sistem memproses selain halaman admin, dan jumlah barang = 0	Sebuah data array dengan value berupa string.	Sebuah data array dengan value berupa string.
3.	Sistem memproses selain halaman admin	Sebuah data array berisi detail barang.	Sebuah data array berisi detail barang.

Sumber: [Pengujian]

6.3. Pengujian Validasi

Pengujian validasi digunakan untuk mengetahui apakah sistem yang dibangun sudah benar sesuai dengan yang dibutuhkan. Item-item yang telah dirumuskan dalam daftar kebutuhan dan merupakan hasil analisis kebutuhan akan menjadi acuan untuk melakukan pengujian validasi. Pengujian validasi menggunakan metode pengujian *Black Box*, karena tidak memerlukan untuk berkonsentrasi terhadap alur jalannya algoritma program dan lebih ditekankan untuk menemukan konformitas antara kinerja sistem dengan daftar kebutuhan.

6.3.1. Kasus Uji Validasi

Untuk mengetahui kesesuaian antara kebutuhan dengan kinerja sistem, pada setiap kebutuhan *(requirement)* dilakukan proses pengujian dengan kasus uji masing-masing.

6.3.1.1. Kasus Uji Registrasi

Tabel 6. 3 : Kasus uji registrasi

Nama kasus uji	Kasus uji registrasi
Obyek uji	Registrasi (aliran utama)
Tujuan pengujian	Sistem dapat memberikan fasilitas registrasi *user*.
Prosedur uji	1. *User* mengakses halaman utama *website* 2. Memasukkan data pada *form* registrasi (*Username*=”firman”, *Password*=“firman”, *Re-Type Password*=”firman”, Nama Lengkap=”Firman FR”, *Email*=”firman@gmail.com”, Alamat=”Jalan Mastrip 96”, Propinsi=”Jawa Timur”, Kota=”Jember”, Kode Pos=”68212”, Telepon=”08563634363”, Masukkan kode=”77bed”). 3. Menekan tombol daftar.
Hasil yang diharapkan	Data *user* berhasil disimpan dalam sistem.

Sumber : Pengujian

Tabel 6. 4 : Kasus uji registrasi dengan masukkan *username*, *password*, *re-type password*, nama lengkap, *email*, alamat, propinsi, kota, kode pos. nomor telepon, atau kode keamanan **kosong**

Nama kasus uji	Kasus uji registrasi dengan masukkan *username*, *password*, *re-type password*, nama lengkap, *email*, alamat, propinsi, kota, kode pos. nomor telepon, atau kode keamanan kosong
Obyek uji	Registrasi (aliran alternatif 1)
Tujuan pengujian	Sistem dapat menampilkan pesan bahwa *username*, *password*, *re-type password*, nama lengkap, *email*, alamat, propinsi, kota, kode pos, nomor telepon, atau kode keamanan harus diisi.
Prosedur uji	1. *User* mengakses halaman utama *website* 2. Memasukkan data pada *form* registrasi (*Username*=””, *Password*=“”, *Re-Type Password*=””, Nama Lengkap=””, *Email*=””, Alamat=””, Propinsi=””, Kota=””, Kode Pos=””, Telepon=””, Masukkan kode=””). 3. Menekan tombol daftar.
Hasil yang diharapkan	Sistem menampilkan pesan bahwa *username*, *password*, *re-type password*, nama lengkap, *email*, alamat, propinsi, kota, kode pos, nomor telepon, atau kode keamanan harus diisi.

Sumber : Pengujian

Tabel 6. 5 : Kasus uji registrasi dengan masukkan *username*, *password*, *email*, nama lengkap, alamat, kode pos, atau nomor telepon memiliki karakter yang tidak diijinkan oleh sistem

Nama kasus uji	Kasus uji registrasi dengan masukkan *username*, *password*, *email*, nama lengkap, alamat, kode pos, atau nomor telepon memiliki karakter yang tidak diijinkan oleh sistem.
Obyek uji	Registrasi (aliran alternatif 2)
Tujuan pengujian	Sistem dapat menampilkan pesan bahwa karakter *username*, *password*, *email*, nama lengkap, alamat, kode pos, atau nomor telepon tidak diperbolehkan.
Prosedur uji	1. *User* mengakses halaman utama *website* 2. Memasukkan data pada *form* registrasi (*Username*=”!ch@nK”, *Password*=“12#$5”,Nama Lengkap=”F1r*4n”, *Email*=”firman.aa”, Alamat=”a&^tr”, Kode Pos=”aze$”, Telepon=”w32rf”). 3. Menekan tombol daftar.
Hasil yang diharapkan	Sistem menampilkan pesan bahwa karakter *username*, *password*, *email*, nama lengkap, alamat, kode pos, atau nomor telepon tidak diperbolehkan.

Sumber : Pengujian

Tabel 6. 6 : Kasus uji registrasi dengan masukkan *username* atau nama lengkap telah ada dalam sistem

Nama kasus uji	Kasus uji registrasi dengan masukkan *username* atau nama lengkap telah ada dalam sistem
Obyek uji	Registrasi (aliran alternatif 3)
Tujuan pengujian	Sistem dapat menampilkan pesan bahwa *username*, atau nama lengkap telah terdaftar.
Prosedur uji	1. *User* mengakses halaman utama *website* 2. Memasukkan data pada *form* registrasi (*Username*=”admin”, Nama Lrngkap=”administrator”). 3. Menekan tombol daftar.
Hasil yang diharapkan	Sistem menampilkan pesan bahwa *username*, atau nama lengkap telah terdaftar.

Sumber : Pengujian

Tabel 6. 7 : Kasus uji registrasi dengan masukkan kode keamanan tidak sesuai

Nama kasus uji	Kasus uji registrasi dengan masukkan kode keamanan tidak sesuai.
Obyek uji	Registrasi (aliran alternatif 4)
Tujuan pengujian	Sistem dapat menampilkan pesan bahwa kode keamanan tidak

	sesuai.
Prosedur uji	1. *User* mengakses halaman utama *website* 2. Memasukkan data pada *form* registrasi (Masukkan kode=”xxx”). 3. Menekan tombol daftar.
Hasil yang diharapkan	Sistem menampilkan pesan bahwa kode keamanan tidak sesuai.

Sumber : Pengujian

6.3.1.2. Kasus Uji *Login*

Tabel 6. 8 : Kasus uji *login*

Nama kasus uji	Kasus uji *login*.
Obyek uji	*Login* (aliran utama)
Tujuan pengujian	Sistem dapat mengijinkan *user* masuk ke sistem sesuai dengan hak aksesnya.
Prosedur uji	1. *User* mengakses halaman utama *website* 2. Memasukkan data pada *form login* (*username*=”firman”, *password*=”firman”). 3. Menekan tombol login.
Hasil yang diharapkan	Sistem mengijinkan *user* masuk sesuai hak aksesnya dan menampilkan status *login*.

Sumber : Pengujian

Tabel 6. 9 : Kasus uji *login* dengan masukkan *username* atau *password* kosong

Nama kasus uji	Kasus uji *login* dengan masukkan *username* atau *password* kosong.
Obyek uji	*Login* (aliran alternatif 1)
Tujuan pengujian	Sistem dapat menampilkan pesan bahwa *username/password* harus diisi.
Prosedur uji	1. *User* mengakses halaman utama *website* 2. Memasukkan data pada *form login* (*username*=””, *password*=””). 3. Menekan tombol login.
Hasil yang diharapkan	Sistem menampilkan pesan bahwa *username/password* harus diisi.

Sumber : Pengujian

Tabel 6. 10 : Kasus uji *login* dengan masukkan *username* atau *password* belum terdaftar pada sistem

Nama kasus uji	Kasus uji *login* dengan masukkan *username* atau *password* belum terdaftar pada sistem.
Obyek uji	*Login* (aliran alternatif 2)
Tujuan pengujian	Sistem dapat menampilkan pesan bahwa *username*/*password* salah.
Prosedur uji	1. *User* mengakses halaman utama *website* 2. Memasukkan data pada *form login* (*username*="", *password*=""). 3. Menekan tombol login.
Hasil yang diharapkan	Sistem menampilkan pesan bahwa *username*/*password* salah.

Sumber : Pengujian

6.3.1.3. Kasus Uji *Logout*

Tabel 6. 11 : Kasus uji *logout*

Nama kasus uji	Kasus uji *logout*.
Obyek uji	*Login* (aliran utama)
Tujuan pengujian	*User* keluar dari sistem.
Prosedur uji	1. *Administrator* atau *costumer* menekan tombol '`logout`' pada status *login*.
Hasil yang diharapkan	*User* keluar dari sistem

Sumber : Pengujian

6.3.1.4. Kasus Uji Melihat Daftar *User*

Tabel 6. 12 : Kasus uji melihat daftar *user*

Nama kasus uji	Kasus uji melihat daftar *user*.
Obyek uji	Melihat daftar *user* (aliran utama)
Tujuan pengujian	Sistem dapat menampilkan daftar *user* terdaftar.
Prosedur uji	1. *Administrator* mengakses halaman administrasi *website* 2. Memilih menu '`Manajemen User`'.
Hasil yang diharapkan	Sistem menampilkan daftar *user* yang ada pada sistem.

Sumber : Pengujian

6.3.1.5. Kasus Uji Melihat Data Profil *User*

Tabel 6. 13 : Kasus uji melihat data profil *user*

Nama kasus uji	Kasus uji melihat data profil *user*.
Obyek uji	Melihat data profil *user* (aliran utama)
Tujuan pengujian	Sistem dapat menampilkan data detail *user*.

Prosedur uji	1. *Customer* telah *login* ke dalam sistem. 2. Menekan tombol "profil" pada bagian status *login* di halaman *website*.
Hasil yang diharapkan	Sistem menampilkan data detail *user*.

Sumber : Pengujian

6.3.1.6. Kasus Uji Mengedit Data Profil *User*

Tabel 6. 14 : Kasus uji mengedit data profil *user*

Nama kasus uji	Kasus uji mengedit data profil *user*.
Obyek uji	Mengedit data profil *user* (aliran utama)
Tujuan pengujian	Sistem dapat menyimpan perubahan pada data *user*.
Prosedur uji	1. *Customer* menekan tombol "edit profil" di bagian data profil. 2. Mengedit data pada *form* edit profil (*username*="fir_man", nama lengkap="FirmanFaizalR", *email*="firman@gmail.com",alamat="Jalan Mastrip no 96", propinsi="Jawa Timur", kota="Jember", kode pos="68212", telepon="08564578954") 3. Menekan tombol "Edit"
Hasil yang diharapkan	Sistem menyimpan perubahan pada data *user*.

Sumber : Pengujian

Tabel 6. 15 : Kasus uji mengedit data profil *user* dengan masukkan *username*, nama lengkap, alamat *user*, *email*, propinsi, kota, kode pos, atau nomor telepon kosong

Nama kasus uji	Kasus uji mengedit data profil *user* dengan masukkan *username*, nama lengkap, alamat *user*, *email*, propinsi, kota, kode pos, atau nomor telepon kosong.
Obyek uji	Mengedit data profil *user* (aliran alternatif 1)
Tujuan pengujian	Sistem dapat menampilkan pesan bahwa *username*, nama lengkap, alamat *user*, *email*, propinsi, kota, kode pos, atau nomor telepon harus diisi.
Prosedur uji	1. *Customer* menekan tombol "edit profil" di bagian data profil. 2. Mengedit data pada *form* edit profil (*username*="", nama lengkap="", *email*="",alamat="", propinsi="", kota="", kode pos="", telepon="") 3. Menekan tombol "Edit"
Hasil yang diharapkan	Sistem menampilkan pesan bahwa *username*, nama lengkap, alamat *user*, *email*, propinsi, kota, kode pos, atau nomor telepon harus diisi.

Sumber : Pengujian

Tabel 6. 16 : Kasus uji mengedit data profil *user* dengan masukkan *username*, nama lengkap, *email*, alamat, kode pos, atau nomor telepon memiliki karakter yang tidak diijinkan oleh sistem

Nama kasus uji	Kasus uji mengedit data profil *user* dengan masukkan *username*, nama lengkap, *email*, alamat, kode pos, atau nomor telepon memiliki karakter yang tidak diijinkan oleh sistem.
Obyek uji	Mengedit data profil *user* (aliran alternatif 2)
Tujuan pengujian	Sistem dapat menampilkan pesan bahwa karakter *username*, nama lengkap, *email*, alamat, kode pos, atau nomor telepon tidak diperbolehkan.
Prosedur uji	1. *Customer* menekan tombol ”edit profil” di bagian data profil. 2. Mengedit data pada *form* edit profil (*username*=”f#$m”, nama lengkap=”I*&%^”, *email*=”Uysh&^”,alamat=”*!@#”, kode pos=”asdzxx”, telepon=”xxx”) 3. Menekan tombol ”Edit”
Hasil yang diharapkan	Sistem menampilkan pesan bahwa karakter *username*, nama lengkap, *email*, alamat, kode pos, atau nomor telepon tidak diperbolehkan.

Sumber : Pengujian

Tabel 6. 17 : Kasus uji mengedit data profil *user* dengan masukkan *username*, atau nama lengkap telah terdaftar

Nama kasus uji	Kasus uji mengedit data profil *user* dengan masukkan *username*, atau nama lengkap telah terdaftar.
Obyek uji	Mengedit data profil *user* (aliran alternatif 3)
Tujuan pengujian	Sistem dapat menampilkan pesan bahwa *username* atau .nama lengkap telah terdaftar.
Prosedur uji	1. *Customer* menekan tombol ”edit profil” di bagian data profil. 2. Mengedit data pada *form* edit profil (*username*=”admin”, nama lengkap=”administrator”) 3. Menekan tombol ”Edit”
Hasil yang diharapkan	Sistem menampilkan pesan bahwa *username* atau .nama lengkap telah terdaftar.

Sumber : Pengujian

6.3.1.7. Kasus Uji Mengedit *Password User*

Tabel 6. 18 : Kasus uji mengedit *password user*

Nama kasus uji	Kasus uji mengedit *password user*.
Obyek uji	Mengedit *password user* (aliran utama)
Tujuan pengujian	Sistem dapat menyimpan *password* baru *user*..
Prosedur uji	1. *Customer* menekan tombol "edit password" di bagian data profil. 2. Memasukkan data pada *form* edit *password* (*password* lama="firman", *password* baru="ichank", *re-type password*="ichank") 3. Menekan tombol "Edit"
Hasil yang diharapkan	Sistem menyimpan *password* baru *user*.

Sumber : Pengujian

Tabel 6. 19 : Kasus uji mengedit *password user* dengan masukkan *password* lama tidak sesuai

Nama kasus uji	Kasus uji mengedit *password user* dengan masukkan *password* lama tidak sesuai.
Obyek uji	Mengedit *password user* (aliran alternatif 1)
Tujuan pengujian	Sistem dapat menampilkan pesan bahwa *password* lama salah.
Prosedur uji	1. *Customer* menekan tombol "edit password" di bagian data profil. 2. Memasukkan data pada *form* edit *password* (*password* lama="xxx", *password* baru="ichank", *re-type password*="ichank") 3. Menekan tombol "Edit"
Hasil yang diharapkan	Sistem menampilkan pesan bahwa *password* lama salah.

Sumber : Pengujian

6.3.1.8. Kasus Uji Menghapus *User*

Tabel 6. 20 : Kasus uji menghapus *user*

Nama kasus uji	Kasus uji menghapus *user*.
Obyek uji	Menghapus *user* (aliran utama)
Tujuan pengujian	Sistem dapat menghapus *user*.
Prosedur uji	1. *Administrator* menekan tombol "hapus" pada daftar *user* di halaman administrasi *website*.
Hasil yang diharapkan	Sistem menghapus data *user*.

Sumber : Pengujian

6.3.1.9. Kasus Uji Mencari Data Barang

Tabel 6. 21 : Kasus uji mencari data barang

Nama kasus uji	Kasus uji mencari data barang.
Obyek uji	Mencari data barang (aliran utama)
Tujuan pengujian	Sistem dapat menampilkan data produk yang dicari oleh *user*.
Prosedur uji	1. *User* mengakses halaman utama *website*. 2. Memasukkan data pada *form search* (cari=”guitar”) 3. Menekan tombol ”Cari”
Hasil yang diharapkan	Sistem menampilkan data produk yang dicari oleh *user*.

Sumber : Pengujian

Tabel 6. 22 : Kasus uji mencari data barang dengan masukan *keyword* kosong

Nama kasus uji	Kasus uji mencari data barang dengan masukan *keyword* kosong.
Obyek uji	Mencari data barang (aliran alternatif 1)
Tujuan pengujian	Sistem dapat menampilkan pesan bahwa data tidak ditemukan.
Prosedur uji	1. *User* mengakses halaman utama *website*. 2. Memasukkan data pada *form search* (cari=””) 3. Menekan tombol ”Cari”
Hasil yang diharapkan	Sistem menampilkan pesan bahwa data tidak ditemukan.

Sumber : Pengujian

6.3.1.10. Kasus Uji Melihat Data Katalog Barang

Tabel 6. 23 : Kasus uji melihat data katalog barang

Nama kasus uji	Kasus uji melihat data katalog barang.
Obyek uji	Melihat data katalog barang (aliran utama)
Tujuan pengujian	Sistem menampilkan daftar barang/produk.
Prosedur uji	1. *User* mengakses halaman utama *website*. 2. Memilih menu ”produk” pada halaman *website*.
Hasil yang diharapkan	Sistem menampilkan daftar barang/produk.

Sumber : Pengujian

6.3.1.11. Kasus Uji Melihat Data Detail Barang

Tabel 6. 24 : Kasus uji melihat data detail barang

Nama kasus uji	Kasus uji melihat data detail barang.
Obyek uji	Melihat data detail barang (aliran utama)
Tujuan pengujian	Sistem menampilkan data detail barang/produk.
Prosedur uji	1. *User* mengakses halaman produk. 2. Menekan tombol ”selengkapnya” pada halaman *website*.

Hasil yang diharapkan	Sistem menampilkan data detail barang/produk.

Sumber : Pengujian

6.3.1.12. Kasus Uji Menambah Barang

Tabel 6. 25 : Kasus uji menambah barang

Nama kasus uji	Kasus uji menambah barang.
Obyek uji	Menambah barang (aliran utama)
Tujuan pengujian	Sistem dapat menyimpan data barang baru.
Prosedur uji	1. *Administrator* mengakses memilih menu "**tambah produk**" pada halaman administrasi *website*. 2. Memasukkan data barang pada *form* tambah produk (nama barang="Yamaha Oak Custom Drum Set", kategori barang="Drum", harga="24360000", berat="16kg-40kg", jumlah barang="5", gambar="YamahaOakCustom.jpg", deskripsi="*Following in the footsteps of our Birch, Maple, and Beech drum series…*"). 3. Menekan tombol "**simpan**"
Hasil yang diharapkan	Sistem menyimpan data barang baru.

Sumber : Pengujian

Tabel 6. 26 : Kasus uji menambah barang dengan m asukan nama barang, kategori barang, harga, berat barang, jumlah barang, gambar, atau deskripsi barang kosong

Nama kasus uji	Kasus uji menambah barang dengan m asukan nama barang, kategori barang, harga, berat barang, jumlah barang, gambar, atau deskripsi barang kosong..
Obyek uji	Menambah barang (aliran alternatif 1)
Tujuan pengujian	Sistem dapat menampilkan pesan bahwa nama barang, kategori barang, harga, berat barang, jumlah barang, gambar, atau deskripsi barang harus diisi.
Prosedur uji	1. *Administrator* mengakses memilih menu "**tambah produk**" pada halaman administrasi *website*. 2. Memasukkan data barang pada *form* tambah produk (nama barang="", kategori barang="", harga="", berat="", jumlah barang="", gambar="", deskripsi=""). 3. Menekan tombol "**simpan**"
Hasil yang diharapkan	Sistem menampilkan pesan bahwa nama barang, kategori barang, harga, berat barang, jumlah barang, gambar, atau deskripsi barang harus diisi.

Sumber : Pengujian

6.3.1.13. Kasus Uji Mengedit Stok Barang

Tabel 6. 27 : Kasus uji mengedit stok barang

Nama kasus uji	Kasus uji mengedit stok barang
Obyek uji	Mengedit stok barang (aliran utama)
Tujuan pengujian	Sistem dapat menyimpan perubahan data stok barang.
Prosedur uji	1. *Administrator* menekan tombol "edit stok" pada daftar barang di halaman administrasi *website*. 2. Mengedit data stok produk pada *form* edit stok produk (jumlah barang="3"). 3. Menekan tombol "simpan"
Hasil yang diharapkan	Sistem menyimpan perubahan data stok barang

Sumber : Pengujian

Tabel 6. 28 : Kasus uji mengedit stok barang dengan masukkan jumlah barang kosong

Nama kasus uji	Kasus uji mengedit stok barang dengan masukkan jumlah barang kosong
Obyek uji	Mengedit stok barang (aliran alternatif 1)
Tujuan pengujian	Sistem dapat menampilkan pesan bahwa jumlah barang harus diisi.
Prosedur uji	1. *Administrator* menekan tombol "edit stok" pada daftar barang di halaman administrasi *website*. 2. Mengedit data stok produk pada *form* edit stok produk (jumlah barang=""). 3. Menekan tombol "simpan"
Hasil yang diharapkan	Sistem menampilkan pesan bahwa jumlah barang harus diisi.

Sumber : Pengujian

6.3.1.14. Kasus Uji Mengedit Data Barang

Tabel 6. 29 : Kasus uji mengedit data barang

Nama kasus uji	Kasus uji mengedit data barang.
Obyek uji	Mengedit data barang (aliran utama)
Tujuan pengujian	Sistem menyimpan perubahan data barang/produk.
Prosedur uji	1. *Administrator* menekan tombol "edit data" pada daftar barang di halaman administrasi *website*. 2. Mengedit data produk pada *form* edit data produk (nama barang="Martin D28 Dreadnought Acoustic Guitar", kategori barang="Gitar", harga="4150000", berat="1kg-5kg",

	deskripsi="*The D28 has an Ebony fingerboard, solid Spruce top, solid Indian*"). 3. Menekan tombol "**simpan**"
Hasil yang diharapkan	Sistem menyimpan perubahan data barang/produk.

Sumber : Pengujian

Tabel 6. 30 : Kasus uji mengedit data barang dengan masukan nama barang, kategori barang, harga barang, berat barang, dan deskripsi barang kosong

Nama kasus uji	Kasus uji mengedit data barang dengan masukan nama barang, kategori barang, harga barang, berat barang, dan deskripsi barang kosong..
Obyek uji	Mengedit data barang (aliran alternatif 1)
Tujuan pengujian	Sistem menampilkan pesan bahwa nama barang, kategori barang, harga barang, berat barang, dan deskripsi barang harus diisi.
Prosedur uji	1. *Administrator* menekan tombol "**edit data**" pada daftar barang di halaman administrasi *website*. 2. Mengedit data produk pada *form* edit data produk (nama barang="", kategori barang="", harga="", berat="", deskripsi=""). 3. Menekan tombol "**simpan**"
Hasil yang diharapkan	Sistem menampilkan pesan bahwa nama barang, kategori barang, harga barang, berat barang, dan deskripsi barang harus diisi.

Sumber : Pengujian

6.3.1.15. Kasus Uji Mengedit Gambar Barang

Tabel 6. 31 : Kasus uji mengedit gambar barang

Nama kasus uji	Kasus uji mengedit gambar barang.
Obyek uji	Mengedit gambar barang (aliran utama)
Tujuan pengujian	Sistem menyimpan perubahan gambar barang/produk.
Prosedur uji	1. *Administrator* menekan tombol "**edit gambar**" pada daftar barang di halaman administrasi *website*. 2. Mengedit data produk pada *form* edit data produk (gambar="MartinD28Dreadnought.jpg"). 3. Menekan tombol "**simpan**"
Hasil yang diharapkan	Sistem menyimpan perubahan gambar barang/produk.

Sumber : Pengujian

Tabel 6. 32 : Kasus uji mengedit gambar barang dengan masukan gambar kosong

Nama kasus uji	Kasus uji mengedit gambar barang dengan masukan gambar kosong.
Obyek uji	Mengedit gambar barang (aliran alternatif 1)
Tujuan pengujian	Sistem Sistem menampilkan pesan bahwa gambar barang tidak boleh kosong.
Prosedur uji	1. *Administrator* menekan tombol "**edit gambar**" pada daftar barang di halaman administrasi *website*. 2. Mengedit data produk pada *form* edit data produk (gambar=""). 3. Menekan tombol "**simpan**"
Hasil yang diharapkan	Sistem Sistem menampilkan pesan bahwa gambar barang tidak boleh kosong.

Sumber : Pengujian

6.3.1.16. Kasus Uji Menghapus Data Barang

Tabel 6. 33 : Kasus uji menghapus data barang

Nama kasus uji	Kasus uji menghapus data barang.
Obyek uji	Menghapus data barang. (aliran utama)
Tujuan pengujian	Sistem dapat menghapus data barang/produk.
Prosedur uji	1. *Administrator* menekan tombol "**hapus**" pada daftar barang di halaman administrasi *website*.
Hasil yang diharapkan	Sistem menghapus data barang/produk

Sumber : Pengujian

6.3.1.17. Kasus Uji Melihat Daftar *Order* Barang

Tabel 6. 34 : Kasus uji melihat daftar order barang

Nama kasus uji	Kasus uji melihat daftar order barang.
Obyek uji	Melihat daftar order barang. (aliran utama)
Tujuan pengujian	Sistem dapat menampikan daftar order produk.
Prosedur uji	1. *Administrator* memilih menu "order" pada halaman administrasi *website*.
Hasil yang diharapkan	Sistem menampilkan daftar order produk

Sumber : Pengujian

6.3.1.18. Kasus Uji Melihat Detail *Order* Barang

Tabel 6. 35 : Kasus uji melihat detail order barang

Nama kasus uji	Kasus uji melihat detail order barang.
Obyek uji	Melihat detail order barang. (aliran utama)
Tujuan pengujian	Sistem dapat menampikan data detail order produk.
Prosedur uji	1. *Administrator* menekan tombol "detail" pada daftar order di halaman administrasi *website*.
Hasil yang diharapkan	Sistem menampilkan data detail order produk

Sumber : Pengujian

6.3.1.19. Kasus Uji Mengedit Status *Order* Barang

Tabel 6. 36 : Kasus uji mengedit status order barang

Nama kasus uji	Kasus uji mengedit status order barang.
Obyek uji	Mengedit status order barang. (aliran utama)
Tujuan pengujian	Sistem dapat menyimpan perubahan status order barang.
Prosedur uji	1. *Administrator* menekan tombol "edit status" pada data *order* barang di halaman administrasi *website*.
Hasil yang diharapkan	Sistem menyimpan perubahan status order barang

Sumber : Pengujian

6.3.1.20. Kasus Uji Menghapus *Order* Barang

Tabel 6. 37 : Kasus uji menghapus order barang

Nama kasus uji	Kasus uji menghapus order barang.
Obyek uji	Menghapus order barang. (aliran utama)
Tujuan pengujian	Sistem dapat menghapus order barang.
Prosedur uji	1. *Administrator* menekan tombol "hapus" pada data *order* barang di halaman administrasi *website*.
Hasil yang diharapkan	Sistem menghapus order barang

Sumber : Pengujian

6.3.1.21. Kasus Uji Melihat *Shopping Cart*

Tabel 6. 38 : Kasus uji melihat *shopping cart*

Nama kasus uji	Kasus uji melihat shopping cart.
Obyek uji	Melihat shopping cart. (aliran utama)
Tujuan pengujian	Sistem dapat menampilkan daftar shopping cart.
Prosedur uji	1. *Customer* menekan tombol "lihat cart" pada halaman *website*.

Hasil yang diharapkan	Sistem menampilkan daftar shopping cart.

Sumber : Pengujian

Tabel 6. 39 : Kasus uji melihat *shopping cart* dengan data item *shopping cart* kosong

Nama kasus uji	Kasus uji melihat shopping cart dengan data item shopping cart kosong.
Obyek uji	Melihat shopping cart. (aliran utama)
Tujuan pengujian	Sistem dapat menampilkan pesan ”Shopping Cart anda masih kosong”.
Prosedur uji	1. *Customer* menekan tombol ”lihat cart” pada halaman *website*.
Hasil yang diharapkan	Sistem menampilkan pesan ”Shopping Cart anda masih kosong”.

Sumber : Pengujian

6.3.1.22. Kasus Uji Menambah *Item Shopping Cart*

Tabel 6. 40 : Kasus uji menambah *item shopping cart*

Nama kasus uji	Kasus uji menambah item shopping cart.
Obyek uji	Menambah shopping cart. (aliran utama)
Tujuan pengujian	Sistem dapat menambah item pada shopping cart.
Prosedur uji	1. *Customer* menekan tombol ”lihat cart” pada halaman *website*.
Hasil yang diharapkan	Sistem menambah item pada shopping cart.

Sumber : Pengujian

6.3.1.23. Kasus Uji Edit *Item Shopping Cart*

Tabel 6. 41 : Kasus uji edit *item shopping cart*

Nama kasus uji	Kasus uji edit item shopping cart.
Obyek uji	Edit item shopping cart. (aliran utama)
Tujuan pengujian	Sistem dapat edit item pada shopping cart.
Prosedur uji	1. *Customer* mengakses halaman shopping cart 2. Mengedit data pada *form* jumlah item shopping cart (jumlah=3) 3. Menekan tombol ”update”
Hasil yang diharapkan	Sistem edit item pada shopping cart.

Sumber : Pengujian

Tabel 6. 42 : Kasus uji *edit item shopping cart* dengan masukan jumlah *item* kosong

Nama kasus uji	Kasus uji edit item shopping cart dengan masukan jumlah item kosong.
Obyek uji	Edit item shopping cart. (aliran alternatif 1)
Tujuan pengujian	Sistem dapat tidak menampilkan perubahan, dan jumlah barang tetap sebanyak data masukkan yang valid.
Prosedur uji	1. *Customer* mengakses halaman shopping cart 2. Mengedit data pada *form* jumlah item shopping cart (jumlah="") 3. Menekan tombol "update"
Hasil yang diharapkan	Sistem tidak menampilkan perubahan, dan jumlah barang tetap sebanyak data masukkan yang valid.

Sumber : Pengujian

Tabel 6. 43 : Kasus uji *edit item shopping cart* dengan masukan jumlah *item* selain angka

Nama kasus uji	Kasus uji edit item shopping cart dengan masukan jumlah item selain angka.
Obyek uji	Edit item shopping cart. (aliran alternatif 2)
Tujuan pengujian	Sistem tidak menampilkan perubahan, dan jumlah barang tetap sebanyak data masukkan yang valid.
Prosedur uji	1. *Customer* mengakses halaman shopping cart 2. Mengedit data pada *form* jumlah item shopping cart (jumlah="x") 3. Menekan tombol "update"
Hasil yang diharapkan	Sistem tidak menampilkan perubahan, dan jumlah barang tetap sebanyak data masukkan yang valid.

Sumber : Pengujian

6.3.1.24. Kasus Uji Menghapus Item *Shopping Cart*

Tabel 6. 44 : Kasus uji menghapus *item shopping cart*

Nama kasus uji	Kasus uji menghapus item shopping cart.
Obyek uji	Menghapus item shopping cart. (aliran utama)
Tujuan pengujian	Sistem dapat menghapus item pada shopping cart.
Prosedur uji	1. *Customer* mengakses halaman shopping cart. 2. Menekan tombol "hapus".
Hasil yang diharapkan	Sistem menghapus item pada shopping cart.

Sumber : Pengujian

6.3.1.25. Kasus Uji Memilih Jasa Kirim

Tabel 6. 45 : Kasus uji memilih jasa kirim

Nama kasus uji	Kasus uji memilih jasa kirim.
Obyek uji	Memilih jasa kirim. (aliran utama)
Tujuan pengujian	Sistem dapat menyimpan jasa kirim yang dipilih oleh *customer*.
Prosedur uji	1. *Customer* menekan tombol "lanjut" pada halaman shopping cart. 2. Memilih jasa kirim untuk pengiriman produk 3. Menekan tombol "lanjut".
Hasil yang diharapkan	Sistem menyimpan jasa kirim yang dipilih oleh *customer*.

Sumber : Pengujian

Tabel 6. 46 : Kasus uji memilih jasa kirim dengan masukan jasa pengiriman barang, alamat, kota, atau kode pos kosong

Nama kasus uji	Kasus uji memilih jasa kirim dengan masukan jasa pengiriman barang, alamat, kota, atau kode pos kosong.
Obyek uji	Memilih jasa kirim. (aliran alternatif 1)
Tujuan pengujian	Sistem dapat menampilkan pesan bahwa jasa pengiriman barang, alamat, kota, kode pos harus diisi.
Prosedur uji	1. *Customer* menekan tombol "lanjut" pada halaman shopping cart. 2. Memilih jasa kirim untuk pengiriman produk 3. Menekan tombol "lanjut".
Hasil yang diharapkan	Sistem menampilkan pesan bahwa jasa pengiriman barang, alamat, kota, kode pos harus diisi.

Sumber : Pengujian

6.3.1.26. Kasus Uji *Checkout Shoppping Cart*

Tabel 6. 47 : Kasus uji *checkout shoppping cart*

Nama kasus uji	Kasus uji checkout shoppping cart
Obyek uji	Checkout shoppping cart (aliran utama)
Tujuan pengujian	Sistem dapat menyimpan data pesanan barang ke dalam sistem.
Prosedur uji	1. *Customer* telah memilih jasa kirim untuk pengiriman produk 2. Menekan tombol "checkout".
Hasil yang diharapkan	Sistem menyimpan data pesanan barang ke dalam sistem.

Sumber : Pengujian

6.3.1.27. Kasus Uji Konfirmasi *Email*

Tabel 6. 48 : Kasus uji konfirmasi email

Nama kasus uji	Kasus uji konfirmasi email
Obyek uji	Konfimasi email (aliran utama)
Tujuan pengujian	Sistem dapat mengirim informasi barang pesanan ke email *customer*.
Prosedur uji	1. *Customer* telah memilih jasa kirim untuk pengiriman produk 2. Menekan tombol "checkout".
Hasil yang diharapkan	Sistem mengirim informasi barang pesanan ke email *customer*.

Sumber : Pengujian

6.3.1.28. Kasus Uji Melihat Daftar Berita

Tabel 6. 49 : Kasus uji melihat daftar berita

Nama kasus uji	Kasus uji melihat daftar berita
Obyek uji	Melihat daftar berita (aliran utama)
Tujuan pengujian	Sistem dapat menampilkan daftar berita.
Prosedur uji	1. *User* mengakses halaman utama website 2. Memilih menu "berita" pada halaman *website*.
Hasil yang diharapkan	Sistem menampilkan daftar berita.

Sumber : Pengujian

6.3.1.29. Kasus Uji Melihat Detail Berita

Tabel 6. 50 : Kasus uji melihat detail berita

Nama kasus uji	Kasus uji melihat detail berita
Obyek uji	Melihat detail berita (aliran utama)
Tujuan pengujian	Sistem dapat menampilkan data detail berita.
Prosedur uji	1. *User* mengakses daftar berita pada halaman utama website 2. Menekan tombol "detail"
Hasil yang diharapkan	Sistem menampilkan data detail berita.

Sumber : Pengujian

6.3.1.30. Kasus Uji Menambah Berita

Tabel 6. 51 : Kasus uji menambah berita

Nama kasus uji	Kasus uji menambah berita
Obyek uji	Menambah berita (aliran utama)
Tujuan pengujian	Sistem dapat menyimpan data berita baru.
Prosedur uji	1. *Administrator* memilih menu "tambah berita" pada halaman

	administrasi *website*. 2. Memasukkan data barang pada *form* tambah barang (judul berita=" Green Day Konser Tur",tanggal=" 06/08/2009",isi berita="Green Day segera memulai tur panjang mereka…") 3. Menekan tombol "Simpan"
Hasil yang diharapkan	Sistem menyimpan data berita baru.

Sumber : Pengujian

Tabel 6. 52 : Kasus uji menambah berita dengan masukkan judul berita, tanggal berita, dan isi berita kosong

Nama kasus uji	Kasus uji menambah berita dengan masukkan judul berita, tanggal berita, dan isi berita kosong.
Obyek uji	Menambah berita (aliran alternatif 1)
Tujuan pengujian	Sistem dapat menampilkan pesan bahwa judul berita, tanggal berita, dan isi berita harus diisi.
Prosedur uji	1. *Administrator* memilih menu "tambah berita" pada halaman administrasi *website*. 2. Memasukkan data barang pada *form* tambah barang (judul berita="", tanggal="", isi berita="") 3. Menekan tombol "Simpan"
Hasil yang diharapkan	Sistem menampilkan pesan bahwa judul berita, tanggal berita, dan isi berita harus diisi.

Sumber : Pengujian

6.3.1.31. Kasus Uji Mengedit Data Berita

Tabel 6. 53 : Kasus uji mengedit data berita

Nama kasus uji	Kasus uji mengedit data berita
Obyek uji	Mengedit data berita (aliran utama)
Tujuan pengujian	Sistem dapat menyimpan perubahan data berita.
Prosedur uji	1. *Administrator* memilih menekan tombol "edit" pada daftar berita di halaman administrasi *website*. 2. Mengedit data barang pada *form* edit barang (judul berita=" Green Day Konser Tur",tanggal=" 06/08/2009",isi berita="Green Day segera memulai tur panjang mereka…") 3. Menekan tombol "Simpan"
Hasil yang diharapkan	Sistem menyimpan data perubahan berita.

Sumber : Pengujian

Tabel 6. 54 : Kasus uji mengedit data berita dengan masukkan judul berita atau isi berita kosong

Nama kasus uji	Kasus uji mengedit data berita dengan masukkan judul berita atau isi berita kosong
Obyek uji	Mengedit data berita (aliran alternatif 1)
Tujuan pengujian	Sistem dapat menampilkan pesan bahwa judul berita dan isi berita harus diisi.
Prosedur uji	1. *Administrator* memilih menekan tombol "edit" pada daftar berita di halaman administrasi *website*. 2. Mengedit data barang pada *form* edit barang (judul berita=" Green Day Konser Tur",tanggal=" 06/08/2009",isi berita="Green Day segera memulai tur panjang mereka…") 3. Menekan tombol "Simpan"
Hasil yang diharapkan	Sistem menampilkan pesan bahwa judul berita dan isi berita harus diisi.

Sumber : Pengujian

6.3.1.32. Kasus Uji Menghapus Data Berita

Tabel 6. 55 : Kasus uji menghapus berita

Nama kasus uji	Kasus uji menghapus berita
Obyek uji	Menghapus berita (aliran utama)
Tujuan pengujian	Sistem dapat menghapus data berita.
Prosedur uji	1. *Administrator* mengakses daftar berita pada halaman administrasi *website*. 2. Menekan tombol "hapus"
Hasil yang diharapkan	Sistem menghapus data berita.

Sumber : Pengujian

6.3.1.33. Kasus Uji Melihat Daftar Konten *Website*

Tabel 6. 56 : Kasus uji melihat daftar konten *website*

Nama kasus uji	Kasus uji melihat daftar konten *website*
Obyek uji	Melihat daftar konten *website* (aliran utama)
Tujuan pengujian	Sistem dapat menampilkan daftar konten *website*.
Prosedur uji	1. *Administrator* mengakses halaman administrasi *website*. 2. Memilih menu "konten website"
Hasil yang diharapkan	Sistem menampilkan daftar konten *website*.

Sumber : Pengujian

6.3.1.34. Kasus Uji Melihat Detail Konten *Website*

Tabel 6. 57 : Kasus uji melihat detail konten *website*

Nama kasus uji	Kasus uji melihat detail konten *website*
Obyek uji	Melihat detail konten *website* (aliran utama)
Tujuan pengujian	Sistem dapat menampilkan data detail konten *website*.
Prosedur uji	1. *Administrator* mengakses halaman *website*. 2. *User* memilih menu "Cara order", "Kontak", "Site Map", atau "Site Credit" pada halaman *website*.
Hasil yang diharapkan	Sistem menampilkan data detail konten *website*.

Sumber : Pengujian

6.3.1.35. Kasus Uji Mengedit Konten *Website*

Tabel 6. 58 : Kasus uji mengedit konten *website*

Nama kasus uji	Kasus uji mengedit konten *website*
Obyek uji	Mengedit konten *website* (aliran utama)
Tujuan pengujian	Sistem dapat menyimpan perubahan data konten *website*.
Prosedur uji	1. *Administrator* menekan tombol "edit" pada daftar konten website di halaman administrasi *website*. 2. Mengedit konten *website* pada *form* edit konten *website* (judul konten=" Site Map", isi konten="1.Home 2. Web Service"). 3. Menekan tombol "Simpan"
Hasil yang diharapkan	Sistem menyimpan perubahan data konten *website*.

Sumber : Pengujian

Tabel 6. 59 : Kasus uji mengedit konten *website* dengan masukkan judul konten dan isi konten kosong

Nama kasus uji	Kasus uji mengedit konten *website* dengan masukkan judul konten dan isi konten kosong
Obyek uji	Mengedit konten *website* (aliran alternatif 1)
Tujuan pengujian	Sistem dapat menampilkan pesan bahwa judul konten dan isi konten harus diisi.
Prosedur uji	1. *Administrator* menekan tombol "edit" pada daftar konten website di halaman administrasi *website*. 2. Mengedit konten *website* pada *form* edit konten *website* (judul konten="", isi konten=""). 3. Menekan tombol "Simpan"
Hasil yang diharapkan	Sistem menampilkan pesan bahwa judul konten dan isi konten

	harus diisi.

Sumber : Pengujian

6.3.1.36. Kasus Uji Registrasi *Website Partner*

Tabel 6. 60 : Kasus uji registrasi w*ebsite partner*

Nama kasus uji	Kasus uji registrasi *website partner*
Obyek uji	Registrasi *website partner* (aliran utama)
Tujuan pengujian	Sistem dapat menyimpan data *partner* baru.
Prosedur uji	1. *Customer* melakukan *login* ke dalam sistem dan tombol ”**registrasi partner**” pada halaman *website* di bagian *web service.* 2. Memasukkan data *partner* pada *form* registrasi *partner* (*website*=”http://www.firman.net”, *web service*=”Katalog Web Service”). 3. Menekan tombol ”**Daftar**”
Hasil yang diharapkan	Sistem menyimpan data *partner* baru.

Sumber : Pengujian

Tabel 6. 61 : Kasus uji registrasi w*ebsite partner* dengan masukkan *website partner* atau jenis *web service* kosong

Nama kasus uji	Kasus uji registrasi *website partner* dengan masukkan *website partner* atau jenis *web service* kosong
Obyek uji	Registrasi *website partner* (aliran alternatif 1)
Tujuan pengujian	Sistem dapat menampilkan pesan bahwa *form website partner* atau jenis *web service* harus diisi..
Prosedur uji	1. *Customer* melakukan *login* ke dalam sistem dan tombol ”**registrasi partner**” pada halaman *website* di bagian *web service.* 2. Memasukkan data *partner* pada *form* registrasi *partner* (*website*=””, *web service*=””). 3. Menekan tombol ”**Daftar**”
Hasil yang diharapkan	Sistem menampilkan pesan bahwa *form website partner* atau jenis *web service* harus diisi.

Sumber : Pengujian

Tabel 6. 62 : Kasus uji registrasi w*ebsite partner* dengan masukkan *website partner* telah ada dalam sistem

Nama kasus uji	Kasus uji registrasi *website partner* dengan masukkan *website partner* telah ada dalam sistem
Obyek uji	Registrasi *website partner* (aliran alternatif 2)
Tujuan pengujian	Sistem dapat menampilkan pesan bahwa *website* telah terdaftar.
Prosedur uji	1. *Customer* melakukan *login* ke dalam sistem dan tombol "registrasi partner" pada halaman *website* di bagian *web service*. 2. Memasukkan data *partner* pada *form* registrasi *partner* (*website*="http://www.wservmusic.com", *web service*="Katalog Web Service"). 3. Menekan tombol "Daftar"
Hasil yang diharapkan	Sistem menampilkan pesan bahwa *website* telah terdaftar.

Sumber : Pengujian

6.3.1.37. Kasus Uji Download SOAP *file*

Tabel 6. 63 : Kasus uji download SOAP *file*

Nama kasus uji	Kasus uji download SOAP *file*
Obyek uji	Download SOAP *file* (aliran utama)
Tujuan pengujian	Sistem dapat menampilkan kotak dialog untuk menyimpan SOAP *file* tersebut dan *user* dapat men-*download*-nya.
Prosedur uji	1. *Customer* berhasil melakukan proses registrasi *partner*. 2. Menekan tombol "klik disini"
Hasil yang diharapkan	Sistem menampilkan kotak dialog untuk menyimpan SOAP *file* tersebut dan *user* dapat men-*download*-nya.

Sumber : Pengujian

6.3.1.38. Kasus Uji Melihat Daftar *Partner Web Service*

Tabel 6. 64 : Kasus uji melihat daftar *partner web service*

Nama kasus uji	Kasus uji melihat daftar *partner web service*
Obyek uji	Melihat daftar *partner web service* (aliran utama)
Tujuan pengujian	Sistem dapat menampilkan daftar *partner web service*.
Prosedur uji	1. *Administrator* mengakses halaman administrasi *website*. 2. Memilih menu "partner web service"
Hasil yang diharapkan	Sistem menampilkan daftar *partner web service*.

Sumber : Pengujian

6.3.1.39. Kasus Uji Melihat Detail *Partner Web Service*

Tabel 6. 65 : Kasus uji melihat detail *partner web service*

Nama kasus uji	Kasus uji melihat detail *partner web service*
Obyek uji	Melihat detail *partner web service* (aliran utama)
Tujuan pengujian	Sistem dapat menampilkan data detail *partner web service*.
Prosedur uji	1. *Administrator* mengakses daftar *partner web service* pada halaman administrasi *website*. 2. Menekan tombol "detail"
Hasil yang diharapkan	Sistem menampilkan data detail *partner web service*.

Sumber : Pengujian

6.3.1.40. Kasus Uji Menghapus *Partner Web Service*

Tabel 6. 66 : Kasus uji menghapus *partner web service*

Nama kasus uji	Kasus uji menghapus *partner web service*
Obyek uji	Menghapus *partner web service* (aliran utama)
Tujuan pengujian	Sistem dapat menghapus data *partner web service*.
Prosedur uji	1. *Administrator* mengakses daftar *partner web service* pada halaman administrasi *website*. 2. Menekan tombol "hapus"
Hasil yang diharapkan	Sistem menampilkan data detail *partner web service*.

Sumber : Pengujian

6.3.1.41. Kasus Uji Melihat Daftar Kategori *Web Service*

Tabel 6. 67 : Kasus uji melihat daftar kategori *web service*

Nama kasus uji	Kasus uji melihat daftar kategori *web service*
Obyek uji	Melihat daftar kategori *web service* (aliran utama)
Tujuan pengujian	Sistem dapat menampilkan daftar kategori *web service*.
Prosedur uji	1. *Administrator* mengakses halaman administrasi *website*. 2. Memilih menu "kategori web service"
Hasil yang diharapkan	Sistem menampilkan daftar kategori *web service*

Sumber : Pengujian

6.3.1.42. Kasus Uji Melihat Detail Kategori *Web Service*

Tabel 6. 68 : Kasus uji melihat detail kategori *web service*

Nama kasus uji	Kasus uji melihat detail kategori *web service*
Obyek uji	Melihat detail kategori *web service* (aliran utama)
Tujuan pengujian	Sistem dapat menampilkan data detail kategori *web service*.

Prosedur uji	1. *Administrator* mengakses daftar kategori *web service* pada halaman administrasi *website*. 2. Menekan tombol "**detail**"
Hasil yang diharapkan	Sistem menampilkan data detail kategori *web service*

Sumber : Pengujian

6.3.1.43. Kasus Uji Mengedit Kategori *Web Service*

Tabel 6. 69 : Kasus uji mengedit kategori *web service*

Nama kasus uji	Kasus uji mengedit kategori *web service*
Obyek uji	Mengedit kategori *web service* (aliran utama)
Tujuan pengujian	Sistem dapat menyimpan perubahan data kategori *web service*.
Prosedur uji	1. *Administrator* menekan tombol "**edit**" pada daftar kategori *web service* di halaman administrasi *website*. 2. Mengedit data kategori *web service* pada *form* edit *web service* (kategori *web service*="Search Web Service", isi *web service*="Search Web Service merupakan aplikasi pencarian data") 3. Menekan tombol "**simpan**"
Hasil yang diharapkan	Sistem menyimpan perubahan data kategori *web service*

Sumber : Pengujian

Tabel 6. 70 : Kasus uji mengedit kategori *web service* dengan masukkan kategori *web service* atau isi *web service* kosong

Nama kasus uji	Kasus uji mengedit kategori *web service* dengan masukkan kategori *web service* atau isi *web service* kosong
Obyek uji	Mengedit kategori *web service* (aliran alternatif 1)
Tujuan pengujian	Sistem dapat menampilkan pesan bahwa kategori *web service* atau isi *web service* harus diisi.
Prosedur uji	1. *Administrator* menekan tombol "**edit**" pada daftar kategori *web service* di halaman administrasi *website*. 2. Mengedit data kategori *web service* pada *form* edit *web service* (kategori *web service*="", isi *web service*="") 3. Menekan tombol "**simpan**"
Hasil yang diharapkan	Sistem menampilkan pesan bahwa kategori *web service* atau isi *web service* harus diisi.

Sumber : Pengujian

6.3.2. Hasil Pengujian Validasi

Dari kasus uji yang telah dilaksanakan sesuai dengan prosedur pengujian pada sub pokok bahasan 6.3.1, didapatkan hasil seperti ditunjukkan pada Tabel 6.71.

Tabel 6. 71 : Hasil Pengujian Validasi

No	Nama kasus uji	Hasil yang didapat	Status validitas
1.	Kasus uji registrasi	Data *user* berhasil disimpan dalam sistem	valid
2.	Kasus uji registrasi dengan masukkan *username*, *password*, *re-type password*, nama lengkap, *email*, alamat, propinsi, kota, kode pos. nomor telepon, atau kode keamanan kosong	Sistem menampilkan pesan bahwa *username*, *password*, *re-type password*, nama lengkap, *email*, alamat, propinsi, kota, kode pos, nomor telepon, atau kode keamanan harus diisi.	valid
3.	Kasus uji registrasi dengan masukkan *username*, *password*, *email*, nama lengkap, alamat, kode pos, atau nomor telepon memiliki karakter yang tidak diijinkan oleh sistem.	Sistem menampilkan pesan bahwa karakter *username*, *password*, *email*, nama lengkap, alamat, kode pos, atau nomor telepon tidak diperbolehkan.	valid
4.	Kasus uji registrasi dengan masukkan *username* atau nama lengkap telah ada dalam sistem.	Sistem menampilkan pesan bahwa *username*, atau nama lengkap telah terdaftar.	valid
5.	Kasus uji registrasi dengan masukkan kode keamanan tidak sesuai.	Sistem menampilkan pesan bahwa kode keamanan tidak sesuai.	valid
6.	Kasus uji *login*	Sistem mengijinkan *user* masuk sesuai hak aksesnya dan menampilkan status *login*.	valid
7.	Kasus uji *login* dengan masukkan *username* atau *password* kosong.	Sistem menampilkan pesan bahwa *username/password* harus diisi.	valid
8.	Kasus uji *login* dengan masukkan *username* atau *password* belum terdaftar pada sistem.	Sistem menampilkan pesan bahwa *username*/*password* salah.	valid
9.	Kasus uji *logout*.	*User* keluar dari sistem	valid
10.	Kasus uji melihat daftar *user*.	Sistem menampilkan daftar *user* yang ada	valid

		pada sistem.	
11.	Kasus uji melihat data profil *user*.	Sistem menampilkan data detail *user*.	valid
12.	Kasus uji mengedit data profil *user*.	Sistem menyimpan perubahan pada data *user*.	valid
13.	Kasus uji mengedit data profil *user* dengan masukkan *username*, nama lengkap, alamat *user*, *email*, propinsi, kota, kode pos, atau nomor telepon kosong.	Sistem menampilkan pesan bahwa *username*, nama lengkap, alamat *user*, *email*, propinsi, kota, kode pos, atau nomor telepon harus diisi.	valid
14.	: Kasus uji mengedit data profil *user* dengan masukkan *username*, nama lengkap, *email*, alamat, kode pos, atau nomor telepon memiliki karakter yang tidak diijinkan oleh sistem.	Sistem menampilkan pesan bahwa karakter *username*, nama lengkap, *email*, alamat, kode pos, atau nomor telepon tidak diperbolehkan.	valid
15.	Kasus uji mengedit data profil *user* dengan masukkan *username*, atau nama lengkap telah terdaftar	Sistem menampilkan pesan bahwa *username* atau .nama lengkap telah terdaftar.	valid
16.	Kasus uji mengedit *password user*	Sistem menyimpan *password* baru *user*.	valid
17.	Kasus uji mengedit *password user* dengan masukkan *password* lama tidak sesuai	Sistem menampilkan pesan bahwa *password* lama salah.	valid
18.	Kasus uji menghapus *user*	Sistem menghapus data *user*.	valid
19.	Kasus uji mencari data barang.	Sistem menampilkan data produk yang dicari oleh *user*.	valid
20.	Kasus uji mencari data barang dengan masukan *keyword* kosong	Sistem menampilkan pesan bahwa data tidak ditemukan.	valid
21.	Kasus uji melihat data katalog barang	Sistem menampilkan daftar barang/produk.	valid
22.	Kasus uji melihat data detail barang	Sistem menampilkan data detail barang/produk.	valid
23.	Kasus uji menambah barang	Sistem menyimpan data barang baru.	valid
24.	Kasus uji menambah barang dengan m asukan nama barang,	Sistem menampilkan pesan bahwa nama barang, kategori barang, harga, berat	valid

	kategori barang, harga, berat barang, jumlah barang, gambar, atau deskripsi barang kosong.	barang, jumlah barang, gambar, atau deskripsi barang harus diisi.	
25.	Kasus uji mengedit stok barang.	Sistem menyimpan perubahan data stok barang	valid
26.	Kasus uji mengedit stok barang dengan masukkan jumlah barang kosong.	Sistem menampilkan pesan bahwa jumlah barang harus diisi.	valid
27.	Kasus uji mengedit data barang.	Sistem menyimpan perubahan data barang/produk.	valid
28.	Kasus uji mengedit data barang dengan masukan nama barang, kategori barang, harga barang, berat barang, dan deskripsi barang kosong.	Sistem menampilkan pesan bahwa nama barang, kategori barang, harga barang, berat barang, dan deskripsi barang harus diisi.	valid
29.	Kasus uji mengedit gambar barang.	Sistem menyimpan perubahan gambar barang/produk.	valid
30.	Kasus uji mengedit gambar barang dengan masukan gambar kosong.	Sistem Sistem menampilkan pesan bahwa gambar barang tidak boleh kosong.	valid
31.	Kasus uji menghapus data barang.	Sistem menghapus data barang/produk	valid
32.	Kasus uji melihat daftar order barang.	Sistem menampilkan daftar order produk	valid
33.	Kasus uji melihat detail order barang.	Sistem menampilkan data detail order produk	valid
34.	Kasus uji mengedit status order barang.	Sistem menyimpan perubahan status order barang	valid
35.	Kasus uji menghapus order barang	Sistem menghapus order barang	valid
36.	Kasus uji melihat *shopping cart*.	Sistem menampilkan daftar *shopping cart*.	valid
37.	Kasus uji melihat *shopping cart* dengan data item *shopping cart* kosong.	Sistem menampilkan pesan ”Shopping Cart anda masih kosong”.	valid
38.	Kasus uji menambah item *shopping cart*.	Sistem menambah item pada *shopping cart*.	valid
39.	Kasus uji edit item *shopping*	Sistem edit item pada *shopping cart*.	valid

	cart.		
40.	Kasus uji edit item *shopping cart* dengan masukan jumlah item kosong.	Sistem tidak menampilkan perubahan, dan jumlah barang tetap sebanyak data masukkan yang valid.	valid
41.	Kasus uji edit item *shopping cart* dengan masukan jumlah item selain angka.	Sistem tidak menampilkan perubahan, dan jumlah barang tetap sebanyak data masukkan yang valid.	valid
42.	Kasus uji menghapus item *shopping cart*.	Sistem menghapus item pada *shopping cart*.	valid
43.	Kasus uji memilih jasa kirim.	Sistem menyimpan jasa kirim yang dipilih oleh *customer*.	valid
44.	Kasus uji memilih jasa kirim dengan masukan jasa pengiriman barang, alamat, kota, atau kode pos kosong.	Sistem menampilkan pesan bahwa jasa pengiriman barang, alamat, kota, kode pos harus diisi.	valid
45.	Kasus uji checkout shoppping cart	Sistem menyimpan data pesanan barang ke dalam sistem.	valid
46.	Kasus uji konfirmasi email	Sistem mengirim informasi barang pesanan ke email *customer*.	valid
47.	Kasus uji melihat daftar berita	Sistem menampilkan daftar berita.	valid
48.	Kasus uji melihat detail berita	Sistem menampilkan data detail berita.	valid
49.	Kasus uji menambah berita	Sistem menyimpan data berita baru.	valid
50.	Kasus uji menambah berita dengan masukkan judul berita, tanggal berita, dan isi berita kosong.	Sistem menampilkan pesan bahwa judul berita, tanggal berita, dan isi berita harus diisi.	valid
51.	Kasus uji mengedit data berita	Sistem menyimpan data perubahan berita.	valid
52.	Kasus uji mengedit data berita dengan masukkan judul berita atau isi berita kosong	Sistem menampilkan pesan bahwa judul berita dan isi berita harus diisi.	valid
53.	Kasus uji menghapus berita	Sistem menghapus data berita.	valid
54.	Kasus uji melihat daftar konten *website*	Sistem menampilkan daftar konten *website*.	valid
55.	Kasus uji melihat detail konten *website*	Sistem menampilkan data detail konten *website*.	valid
56.	Kasus uji mengedit konten *website*	Sistem menyimpan perubahan data konten *website*.	valid
57.	Kasus uji mengedit konten	Sistem menampilkan pesan bahwa judul	valid

	website dengan masukkan judul konten dan isi konten kosong	konten dan isi konten harus diisi.	
58.	Kasus uji registrasi *website partner*	Sistem menyimpan data *partner* baru.	valid
59.	Kasus uji registrasi *website partner* dengan masukkan *website partner* atau jenis *web service* kosong	Sistem menampilkan pesan bahwa *form website partner* atau jenis *web service* harus diisi.	valid
60.	Kasus uji registrasi *website partner* dengan masukkan *website partner* telah ada dalam sistem	Sistem menampilkan pesan bahwa *website* telah terdaftar	valid
61.	Kasus uji download SOAP *file*	Sistem menampilkan kotak dialog untuk menyimpan SOAP *file* tersebut dan *user* dapat men-*download*-nya.	valid
62.	Kasus uji melihat daftar *partner web service*	Sistem menampilkan daftar *partner web service*.	valid
63.	Kasus uji melihat detail *partner web service*	Sistem menampilkan data detail *partner web service*.	valid
64.	Kasus uji menghapus *partner web service*	Sistem menampilkan data detail *partner web service*.	valid
65.	Kasus uji melihat daftar kategori *web service*	Sistem menampilkan daftar kategori *web service*	valid
66.	Kasus uji melihat detail kategori *web service*	Sistem menampilkan data detail kategori *web service*	valid
67.	Kasus uji mengedit kategori *web service*	Sistem menyimpan perubahan data kategori *web service*	valid
68.	Kasus uji mengedit kategori *web service* dengan masukkan kategori *web service* atau isi *web service* kosong	Sistem menampilkan pesan bahwa kategori *web service* atau isi *web service* harus diisi.	valid

Sumber : [Pengujian]

6.4. Pengujian Perancangan Basis Data

Pengujian perancangan basis data bertujuan untuk mengetahui apakah implementasi perancangan basis data yang dilakukan telah sesuai dengan *entity relationship diagram*. Pengujian perancangan basis data dilakukan dengan

menggunakan perangkat lunak Sybase PowerDesigner 15.0.

Pengujian meliputi pembuatan tabel `user`, `kategori_user`, `barang`, `kategori_barang`, `berat_kirim`, `data_kirim`, `jasa_kirim`, `order_user`, `shopping_cart`, `status_bayar`, `data_website`, `kategori_data_website`, `web_service`, `kategori_webservice`, `propinsi`, dan `kota` pada basis data `db_wse`.

A. Tujuan

Pengujian dilakukan untuk mengetahui apakah proses pembuatan basis data `simada` dan pemeriksaan validitas rancangan tabel yang dilakukan telah sesuai dengan *Entity Relationship Diagram.*

B. Spesifikasi dan Konfigurasi Komputer

Spesifikasi perangkat keras yang digunakan dalam melakukan pengujian perancangan basis data ialah :

- Prosesor Intel® Pentium® IV 2.66 Hz.
- Sistem operasi Microsoft Windows XP Professional Service pack 3.

C. Software Aplikasi

Perangkat lunak atau *software* yang digunakan dalam pengujian perancangan basis data ialah :

- Sybase PowerDesigner 15.0
- *Server* basis data MySQL Versi MySQL version 5.1.30 (**`mysqld-nt.exe`**)
- SQL Shell (`mysql.exe`)

D. Prosedur Pengujian

- *Window* Command Prompt dijalankan dengan memberikan perintah sebagai berikut:

  ```
  Start | Run... | Open: cmd.exe
  ```

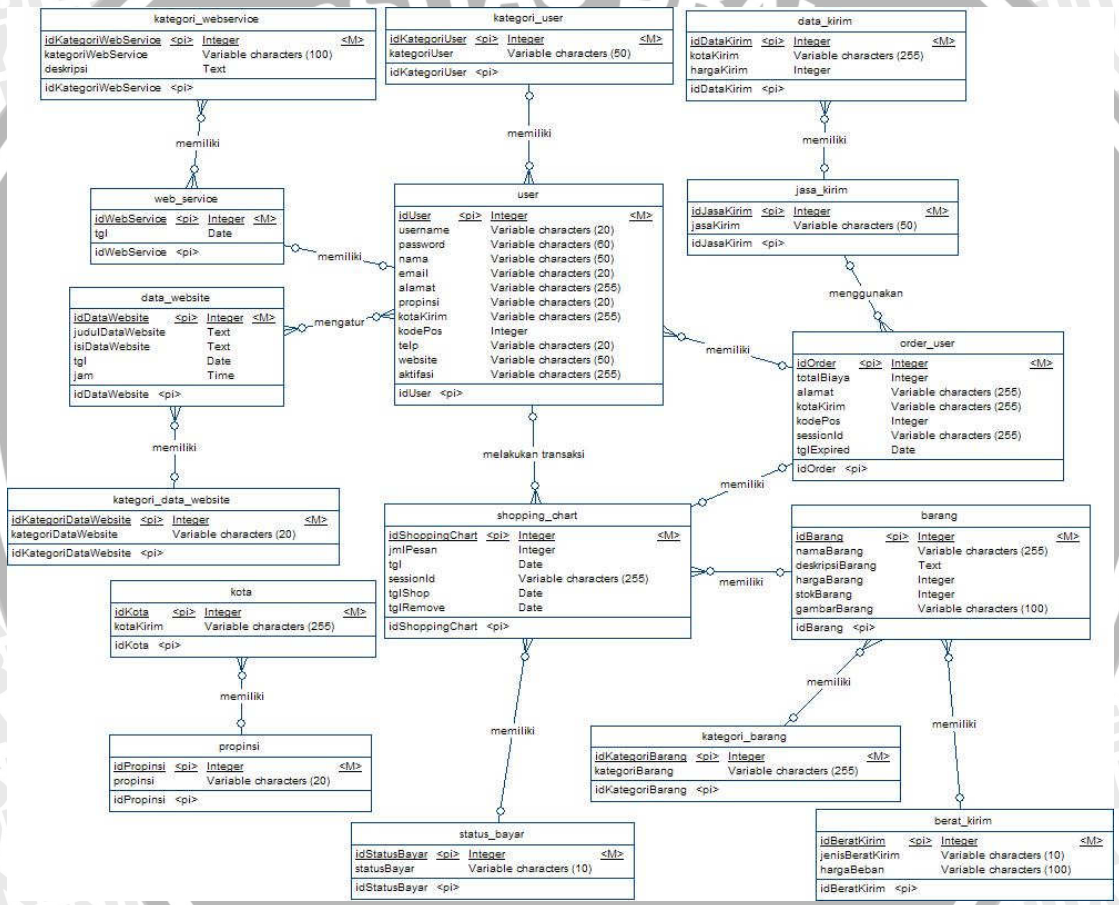
- *Server* basis data MySQL dijalankan dengan memberikan perintah sebagai berikut:

  ```
  C:\> net start mysql
  ```

- SQL *Shell* dijalankan dengan memberikan perintah sebagai berikut:

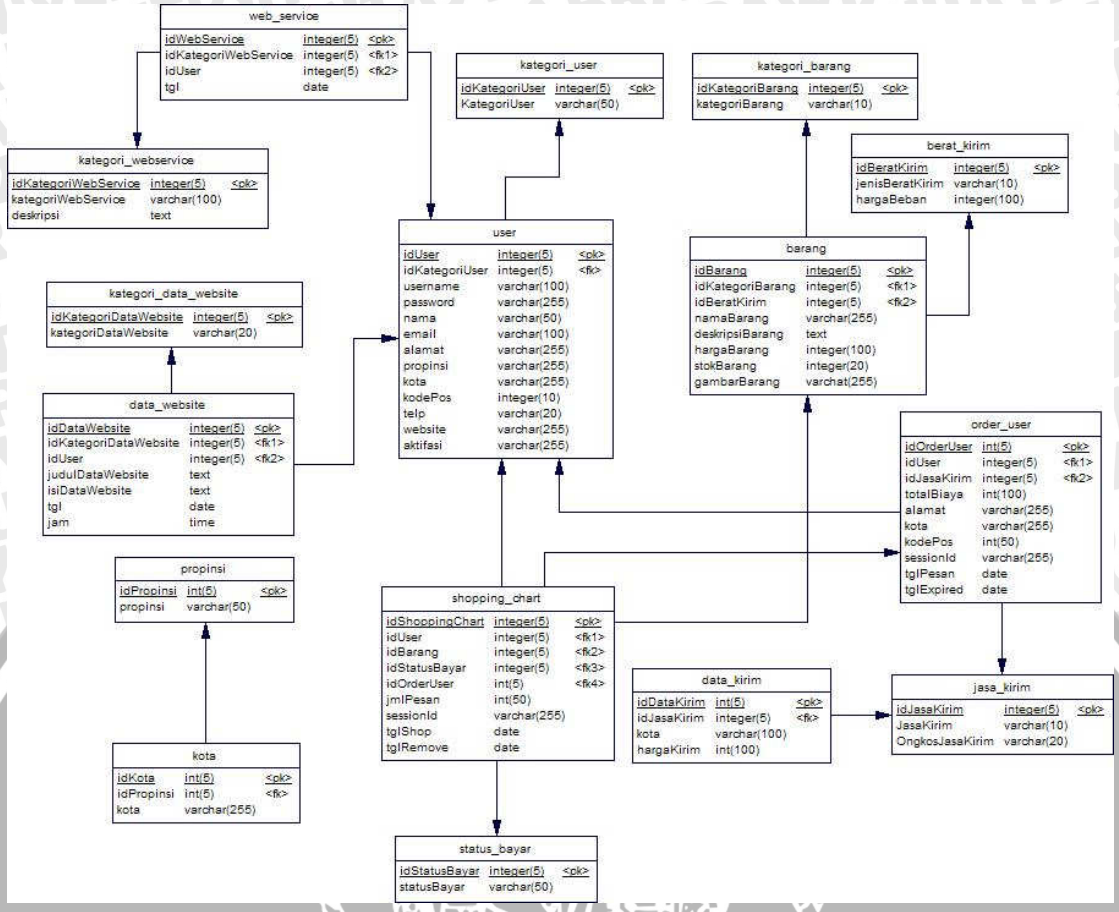
```
C:\>mysql –u root –p
Enter password:
```

- *Software* Sybase Power Designer 15 dijalankan dengan memilih menu pada *start button windows* sebagai berikut:
  `Start | All Programs | Sybase PowerDesigner 15 | PowerDesigner`
- *Entity Relationship Diagram* dari basis data simada digambarkan kembali pada area kerja *Conceptual Data Model* (CDM).
- *Entity Relationship Diagram* diperiksa kembali dengan cara menekan tombol *Check Model* pada *toolbar*. Hasil pemeriksaan ini disebut dengan CDM *Object* yang ditunjukkan dalam Gambar 6.4.



Gambar 6. 4 : *Conceptual Data Model Object* untuk basis data `db_wse`
Sumber : [Pengujian]

- CDM *Object* kemudian diubah menjadi *Physical Data Model* (PDM) *Object* dengan menekan *link Generate Physical Data Model* pada *toolbar Tools*. PDM *Object* ditunjukkan dalam gambar 6.5.



Gambar 6. 5 : *Physical Data Model Object* untuk basis data `db_wse`
Sumber : [Pengujian]

- PDM *Object* dapat diubah ke basis data MySQL dengan cara menekan *link Generate Database* pada *toolbar Database*.
- Basis data `db_wse` ditampilkan dengan menggunakan perintah SQL pada SQL *Shell* sebagai berikut:

```
mysql> show databases;
```

E. Proses Pengujian dan Analisis

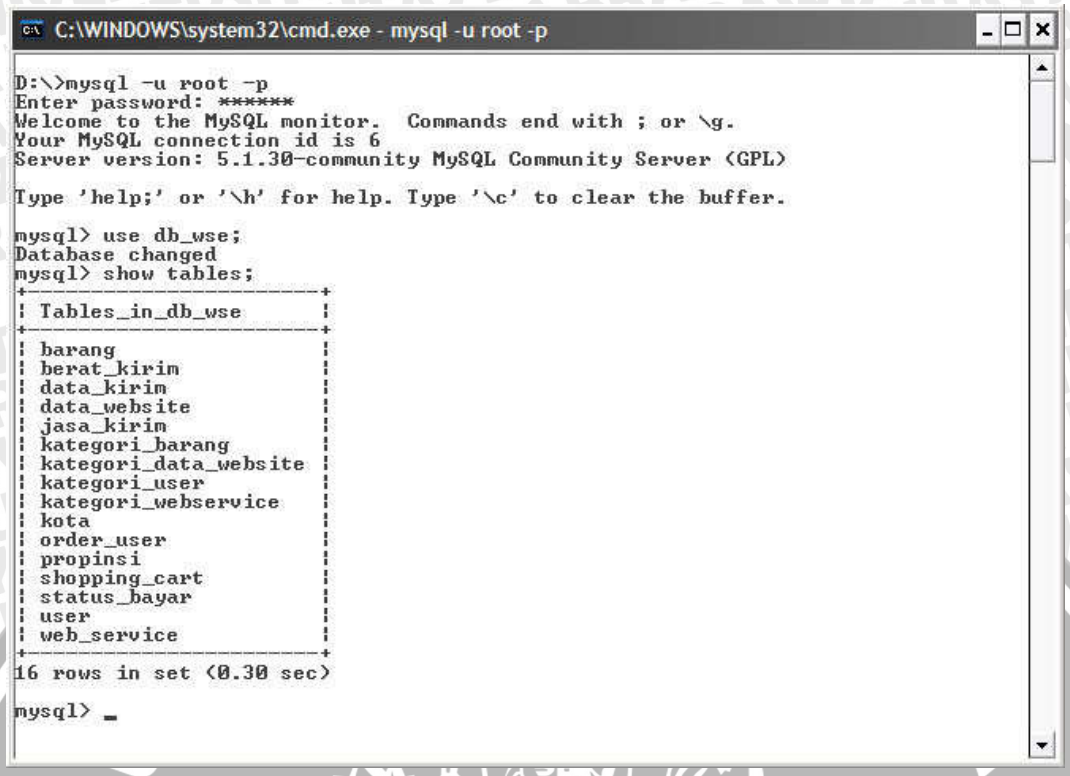
- Berikut ditunjukkan hasil *Generate Database* untuk basis data `db_wse`:

```
Database Generation
Generation: Check model starting...
Generation: Check model successful.
Sorting objects...
Sort completed.
```

```
Script Generation...
-> Table: barang (barang)
-> Table: berat_kirim (berat_kirim)
-> Table: data_kirim (data_kirim)
-> Table: data_website (data_website)
-> Table: jasa_kirim (jasa_kirim)
-> Table: kategori_barang (kategori_barang)
-> Table: kategori_data_website (kategori_data_website)
-> Table: kategori_user (kategori_user)
-> Table: kategori_webservice (kategori_webservice)
-> Table: kota (kota)
-> Table: order_user (order_user)
-> Table: propinsi (propinsi)
-> Table: shopping_chart (shopping_chart)
-> Table: status_bayar (status_bayar)
-> Table: user (user)
-> Table: web_service (web_service)
-> Table: barang (barang)
-> Table: berat_kirim (berat_kirim)
-> Table: data_kirim (data_kirim)
-> Table: data_website (data_website)
-> Table: jasa_kirim (jasa_kirim)
-> Table: kategori_barang (kategori_barang)
-> Table: kategori_data_website (kategori_data_website)
-> Table: kategori_user (kategori_user)
-> Table: kategori_webservice (kategori_webservice)
-> Table: kota (kota)
-> Table: order_user (order_user)
-> Table: propinsi (propinsi)
-> Table: shopping_chart (shopping_chart)
-> Table: status_bayar (status_bayar)
-> Table: user (user)
-> Table: web_service (web_service)
-> Reference: Reference_6 (REFERENCE_6)
-> Reference: Reference_7 (REFERENCE_7)
-> Reference: Reference_8 (REFERENCE_8)
```

```
-> Reference: Reference_5 (REFERENCE_5)
-> Reference: Reference_9 (REFERENCE_9)
-> Reference: Reference_4 (REFERENCE_4)
-> Reference: Reference_10 (REFERENCE_10)
-> Reference: Reference_11 (REFERENCE_11)
-> Reference: Reference_12 (REFERENCE_12)
-> Reference: Reference_13 (REFERENCE_13)
-> Reference: Reference_14 (REFERENCE_14)
-> Reference: Reference_15 (REFERENCE_15)
-> Reference: Reference_1 (REFERENCE_1)
-> Reference: Reference_2 (REFERENCE_2)
-> Reference: Reference_3 (REFERENCE_3)
Script Generation completed
Generation successful

Usage:
  (1) Start command prompt
  (2) Go to the directory G:\
  (3) Start the SQL interpreter:
        mysql.exe
  (4) Run the database creation script:
        mysql> source db_wse.sql
```

- *Script Generate Database* yang dihasilkan sama dengan *query* SQL. Hasil *Generate Database* menunjukkan bahwa desain basis data **db_wse** yang dibuat sudah benar (*valid*).
- Basis data **db_wse** hasil proses *Generate Database* dari PDM *Object* ditunjukkan dalam Gambar 6.6.



Gambar 6. 6 : Basis data db_wse hasil proses *Generate Database* pada Sybase PowerDesigner 15.0

Sumber : [Pengujian]

F. Hasil Pengujian dan Analisis

- *Software* Sybase PowerDesigner 15.0 dapat digunakan untuk pembuatan basis data db_wse dan pemeriksaan validitas tabel.
- Basis data db_wse hasil proses *Generate Database* dari PDM *Object* sama dengan hasil implementasi basis data db_wse pada MySQL

6.5. Pengujian Performansi Koneksi *Web Service*

Pengujian performansi *web service* dilakukan untuk mengetahui performa koneksi antara komputer *server* (*provider entity*) dengan komputer *partner* (*requester entity*), dan komputer *server* (*provider entity*) dengan komputer *client*. Pengujian performansi koneksi terdiri dari pengujian koneksi basis data dan *web server*, pengujian waktu akses *query*, dan pengujian performansi *web server*.

6.5.1. Pengujian Koneksi Basis Data Dan *Web Server*

A. Tujuan

- Pengujian dilakukan untuk mengetahui koneksi *server* basis data MySQL dan *web server* Apache HTTP yang terletak di komputer *server* dalam sebuah jaringan LAN (*Local Area Network*).

A. Prosedur Pengujian

Prosedur pengujian untuk mengetahui performansi koneksi antara komputer *server* (*provider entity*) dengan komputer *partner* (*requester entity*) sebagai berikut :

1. **PC *Server (provider entity)* :**

- *Window* Command Prompt dijalankan dengan memberikan perintah sebagai berikut : `Start | Run... | Open: cmd.exe`
- *Server* basis data MySQL dijalankan dengan memberikan perintah sebagai berikut : `C:\Documents and Settings\>net start mysql`
- Aplikasi dan koneksi yang sedang aktif dapat ditampilkan dengan memberikan perintah sebagai berikut : `C:\Documents and Settings\>netstat –an`

2. **PC *Partner* (*requester entity*) :**

- Aplikasi *Web Partner* dijalankan dengan menuliskan alamat `http://172.17.68.245/webservice-partner/index.php` (*localhost partner*) pada halaman *web browser* di komputer *partner*.
- Aplikasi dan koneksi yang sedang aktif dapat ditampilkan dengan memberikan perintah sebagai berikut:
  `C:\Documents and Settings\ >netstat –an`

3. **PC *Server (provider entity)*:**

- Aplikasi dan koneksi yang sedang aktif dapat ditampilkan dengan memberikan perintah sebagai berikut : `c:\Documents and Settings\>netstat –an`

Prosedur pengujian untuk mengetahui performansi koneksi antara komputer *server* (*provider entity*) dengan dengan komputer *client* sebagai berikut :

1. **PC *Server (provider entity)* :**

- *Window* Command Prompt dijalankan dengan memberikan perintah sebagai berikut : `Start | Run... | Open: cmd.exe`
- *Server* basis data MySQL dijalankan dengan memberikan perintah sebagai berikut : `C:\Documents and Settings\>net start mysql`
- Aplikasi dan koneksi yang sedang aktif dapat ditampilkan dengan memberikan perintah sebagai berikut : `C:\Documents and Settings\>netstat –an`

2. **PC *Client* :**

- Aplikasi *Web Service E-Commerce* dijalankan dengan menuliskan alamat `http://172.17.68.232/webservice-ecommerce/index.php` pada halaman *web browser*.
- Aplikasi dan koneksi yang sedang aktif dapat ditampilkan dengan memberikan perintah sebagai berikut:
  `C:\Documents and Settings\ >netstat –an`

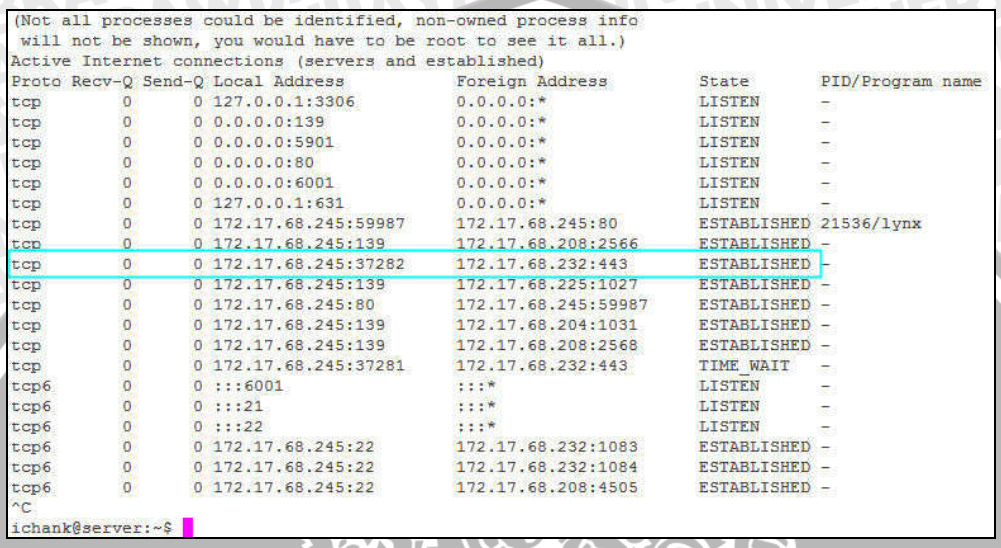
3. **PC *Server (provider entity)* :**

- Aplikasi dan koneksi yang sedang aktif dapat ditampilkan dengan memberikan perintah sebagai berikut : `c:\Documents and Settings\>netstat –an`

B. Proses Pengujian

- Perintah `netstat –a –n -tcp` pada komputer *partner* (*requester entity*) yang digunakan saat ada koneksi dengan komputer *server* (*provider entity*) ditunjukkan dalam gambar 6.7. Koneksi terbentuk setelah komputer *partner* mengakses SOAP dan WSDL yang terdapat pada komputer *server* (*provider entity*). Koneksi SOAP dan WSDL menggunakan HTTPS (port

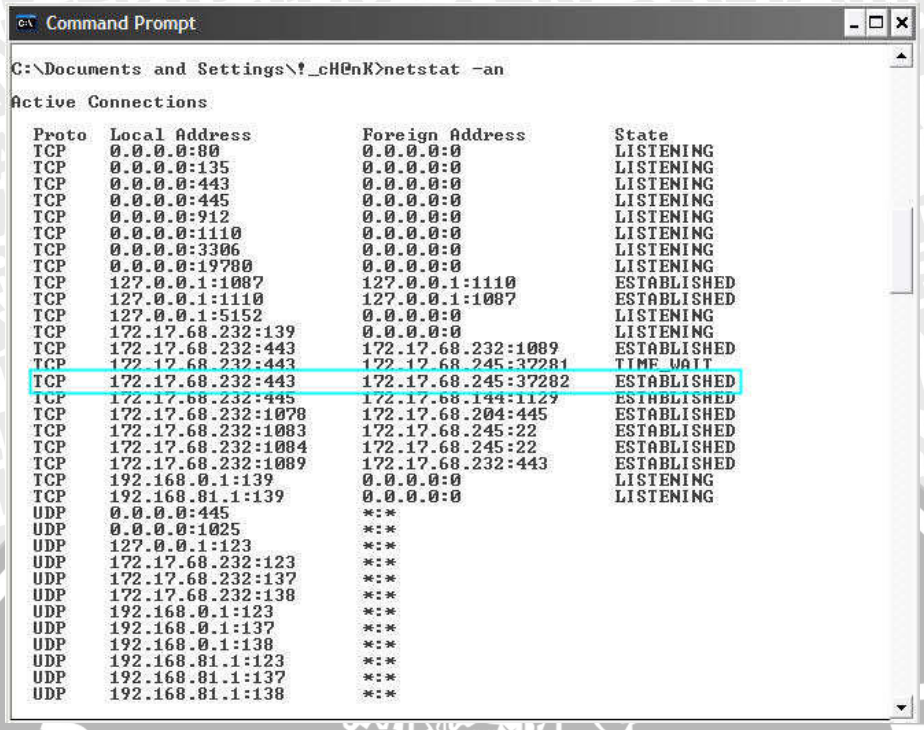
443). Koneksi antara komputer *server* (*provider entity*) dengan komputer *partner* (*requester entity*) ditunjukkan dengan adanya kondisi (*state*): `ESTABLISHED`. Kondisi (*state*): `ESTABLISHED` menunjukkan bahwa modul *web service* pada komputer *server* (*provider entity*) dapat diakses melalui komputer *partner* (*requester entity*).



```
(Not all processes could be identified, non-owned process info
 will not be shown, you would have to be root to see it all.)
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 127.0.0.1:3306          0.0.0.0:*               LISTEN      -
tcp        0      0 0.0.0.0:139             0.0.0.0:*               LISTEN      -
tcp        0      0 0.0.0.0:5901            0.0.0.0:*               LISTEN      -
tcp        0      0 0.0.0.0:80              0.0.0.0:*               LISTEN      -
tcp        0      0 0.0.0.0:6001            0.0.0.0:*               LISTEN      -
tcp        0      0 127.0.0.1:631           0.0.0.0:*               LISTEN      -
tcp        0      0 172.17.68.245:59987     172.17.68.245:80        ESTABLISHED 21536/lynx
tcp        0      0 172.17.68.245:139       172.17.68.208:2566      ESTABLISHED -
tcp        0      0 172.17.68.245:37282     172.17.68.232:443       ESTABLISHED -
tcp        0      0 172.17.68.245:139       172.17.68.225:1027      ESTABLISHED -
tcp        0      0 172.17.68.245:80        172.17.68.245:59987     ESTABLISHED -
tcp        0      0 172.17.68.245:139       172.17.68.204:1031      ESTABLISHED -
tcp        0      0 172.17.68.245:139       172.17.68.208:2568      ESTABLISHED -
tcp        0      0 172.17.68.245:37281     172.17.68.232:443       TIME_WAIT   -
tcp6       0      0 :::6001                 :::*                    LISTEN      -
tcp6       0      0 :::21                   :::*                    LISTEN      -
tcp6       0      0 :::22                   :::*                    LISTEN      -
tcp6       0      0 172.17.68.245:22        172.17.68.232:1083      ESTABLISHED -
tcp6       0      0 172.17.68.245:22        172.17.68.232:1084      ESTABLISHED -
tcp6       0      0 172.17.68.245:22        172.17.68.208:4505      ESTABLISHED -
^C
ichank@server:~$
```

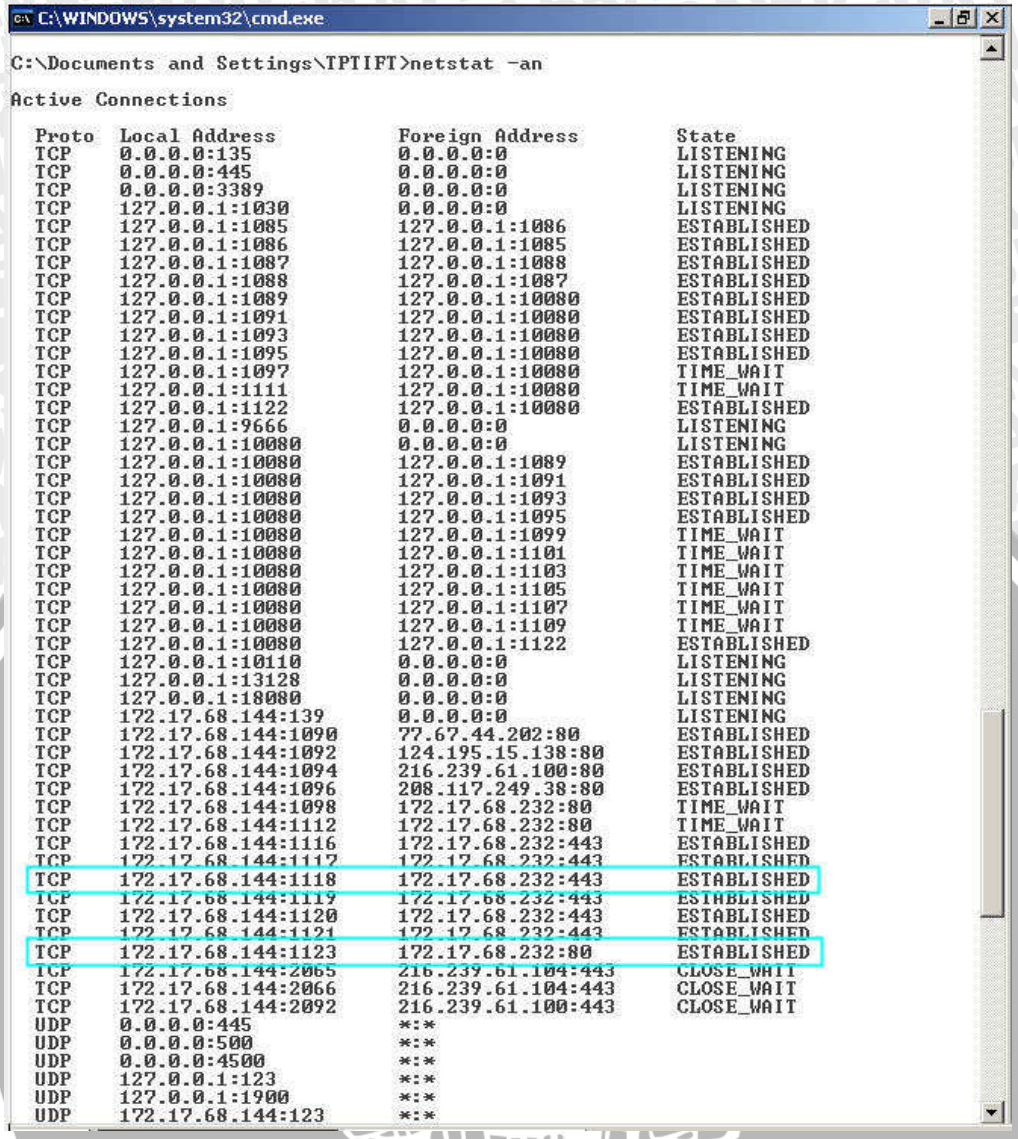
Gambar 6. 7 : Perintah `netstat -an` pada komputer *partner* saat ada koneksi
Sumber : [Pengujian]

- Perintah `netstat -an` pada komputer *server* yang digunakan saat ada koneksi dengan komputer *partner* (*requester entity*) ditunjukkan dalam Gambar 6.8. Koneksi terbentuk setelah komputer *partner* (*requester entity*) membuka aplikasi *partner*. Koneksi antara komputer *server* pada alamat IP 172.17.68.232 dengan komputer *partner* (*requester entity*) pada alamat IP 172.17.68.245 ditunjukkan dengan adanya kondisi (*state*): `ESTABLISHED`. Kondisi (*state*): `ESTABLISHED` menunjukkan bahwa pertukaran data antara komputer *server* (*provider entity*) dan komputer *partner* (*requester entity*) dapat dilakukan.



Gambar 6. 8 : Perintah `netstat –an` pada komputer *server* saat ada koneksi
Sumber : [Pengujian]

- Perintah `netstat -an` pada komputer *client* yang digunakan saat ada koneksi dengan komputer *server* (*provider entity*) ditunjukkan dalam Gambar 6.9. Koneksi terbentuk setelah komputer *client* membuka Aplikasi *Web Service E-Commerce*. Koneksi antara komputer *server* (*provider entity*) pada alamat IP 172.17.68.232 dengan komputer *client* pada alamat IP 172.17.68.144 ditunjukkan dengan adanya kondisi (*state*): `ESTABLISHED`. Kondisi (*state*): `ESTABLISHED` menunjukkan komputer *client* dapat terhubung dengan *server* basis data MySQL dan *web server* Apache HTTP di komputer *server* (*provider entity*).

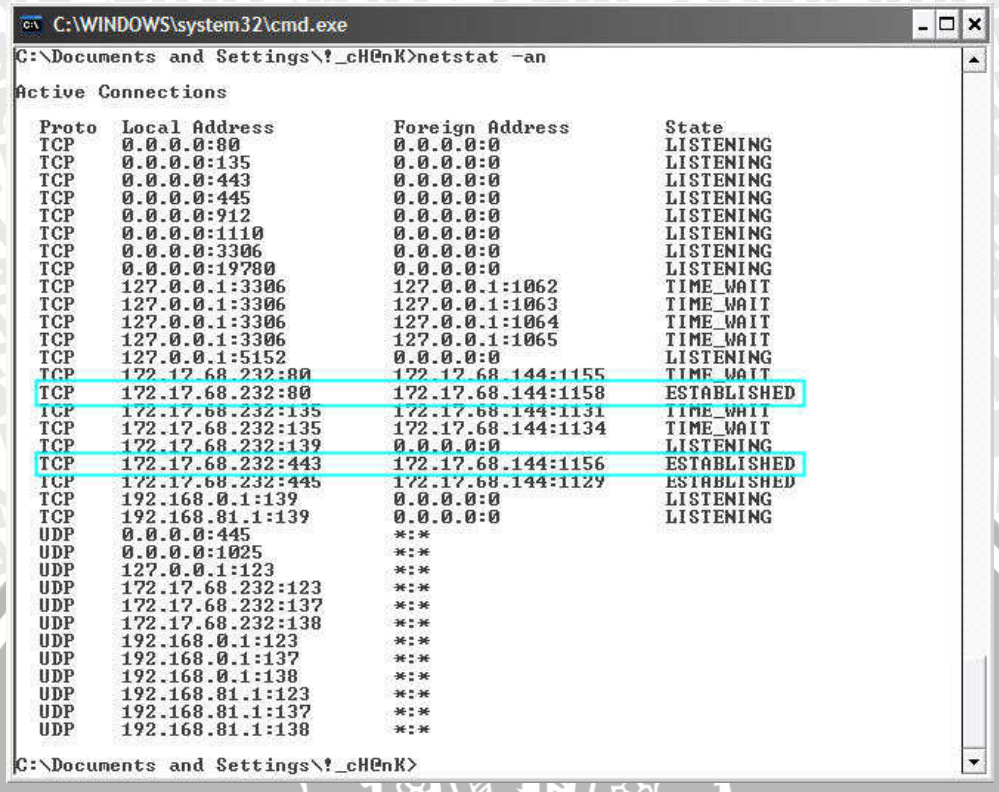


C:\WINDOWS\system32\cmd.exe

```
C:\Documents and Settings\TPTIFT>netstat -an

Active Connections

  Proto  Local Address          Foreign Address        State
  TCP    0.0.0.0:135            0.0.0.0:0              LISTENING
  TCP    0.0.0.0:445            0.0.0.0:0              LISTENING
  TCP    0.0.0.0:3389           0.0.0.0:0              LISTENING
  TCP    127.0.0.1:1030         0.0.0.0:0              LISTENING
  TCP    127.0.0.1:1085         127.0.0.1:1086         ESTABLISHED
  TCP    127.0.0.1:1086         127.0.0.1:1085         ESTABLISHED
  TCP    127.0.0.1:1087         127.0.0.1:1088         ESTABLISHED
  TCP    127.0.0.1:1088         127.0.0.1:1087         ESTABLISHED
  TCP    127.0.0.1:1089         127.0.0.1:10080        ESTABLISHED
  TCP    127.0.0.1:1091         127.0.0.1:10080        ESTABLISHED
  TCP    127.0.0.1:1093         127.0.0.1:10080        ESTABLISHED
  TCP    127.0.0.1:1095         127.0.0.1:10080        ESTABLISHED
  TCP    127.0.0.1:1097         127.0.0.1:10080        TIME_WAIT
  TCP    127.0.0.1:1111         127.0.0.1:10080        TIME_WAIT
  TCP    127.0.0.1:1122         127.0.0.1:10080        ESTABLISHED
  TCP    127.0.0.1:9666         0.0.0.0:0              LISTENING
  TCP    127.0.0.1:10080        0.0.0.0:0              LISTENING
  TCP    127.0.0.1:10080        127.0.0.1:1089         ESTABLISHED
  TCP    127.0.0.1:10080        127.0.0.1:1091         ESTABLISHED
  TCP    127.0.0.1:10080        127.0.0.1:1093         ESTABLISHED
  TCP    127.0.0.1:10080        127.0.0.1:1095         ESTABLISHED
  TCP    127.0.0.1:10080        127.0.0.1:1099         TIME_WAIT
  TCP    127.0.0.1:10080        127.0.0.1:1101         TIME_WAIT
  TCP    127.0.0.1:10080        127.0.0.1:1103         TIME_WAIT
  TCP    127.0.0.1:10080        127.0.0.1:1105         TIME_WAIT
  TCP    127.0.0.1:10080        127.0.0.1:1107         TIME_WAIT
  TCP    127.0.0.1:10080        127.0.0.1:1109         TIME_WAIT
  TCP    127.0.0.1:10080        127.0.0.1:1122         ESTABLISHED
  TCP    127.0.0.1:10110        0.0.0.0:0              LISTENING
  TCP    127.0.0.1:13128        0.0.0.0:0              LISTENING
  TCP    127.0.0.1:18080        0.0.0.0:0              LISTENING
  TCP    172.17.68.144:139      0.0.0.0:0              LISTENING
  TCP    172.17.68.144:1090     77.67.44.202:80        ESTABLISHED
  TCP    172.17.68.144:1092     124.195.15.138:80      ESTABLISHED
  TCP    172.17.68.144:1094     216.239.61.100:80      ESTABLISHED
  TCP    172.17.68.144:1096     208.117.249.38:80      ESTABLISHED
  TCP    172.17.68.144:1098     172.17.68.232:80       TIME_WAIT
  TCP    172.17.68.144:1112     172.17.68.232:80       TIME_WAIT
  TCP    172.17.68.144:1116     172.17.68.232:443      ESTABLISHED
  TCP    172.17.68.144:1117     172.17.68.232:443      ESTABLISHED
  TCP    172.17.68.144:1118     172.17.68.232:443      ESTABLISHED
  TCP    172.17.68.144:1119     172.17.68.232:443      ESTABLISHED
  TCP    172.17.68.144:1120     172.17.68.232:443      ESTABLISHED
  TCP    172.17.68.144:1121     172.17.68.232:443      ESTABLISHED
  TCP    172.17.68.144:1123     172.17.68.232:80       ESTABLISHED
  TCP    172.17.68.144:2065     216.239.61.104:443     CLOSE_WAIT
  TCP    172.17.68.144:2066     216.239.61.104:443     CLOSE_WAIT
  TCP    172.17.68.144:2092     216.239.61.100:443     CLOSE_WAIT
  UDP    0.0.0.0:445            *:*
  UDP    0.0.0.0:500            *:*
  UDP    0.0.0.0:4500           *:*
  UDP    127.0.0.1:123          *:*
  UDP    127.0.0.1:1900         *:*
  UDP    172.17.68.144:123      *:*
```

Gambar 6. 9 : Perintah `netstat -an` pada komputer *client* saat ada koneksi
Sumber : [Pengujian]

- Perintah `netstat -an` pada komputer *server* yang digunakan saat ada koneksi dengan komputer *client* ditunjukkan dalam Gambar 6.10. Koneksi terbentuk setelah komputer *client* membuka aplikasi *Web Service E-Commerce*. Koneksi antara komputer *server* pada alamat IP 172.17.68.232 dengan komputer *client* pada alamat IP 172.17.68.144 ditunjukkan dengan adanya kondisi (*state*): ESTABLISHED. Kondisi (*state*): ESTABLISHED menunjukkan bahwa pertukaran data antara komputer *server* (*provider entity*) dan komputer *client* dapat dilakukan.



Gambar 6. 10 : Perintah `netstat -an` pada komputer *server* saat ada koneksi dari *client*
Sumber : [Pengujian]

D. Hasil Pengujian dan Analisis

- Aplikasi *Web Service E-Commerce* dapat dijalankan pada jaringan komputer yang menggunakan protokol TCP/IP.
- SOAP dan WSDL yang terhubung pada basis data MySQL dan *web server* Apache HTTP yang terdapat pada komputer *server* (*provider entity*) dapat diakses dari komputer *partner* (*requester entity*).
- Koneksi basis data MySQL dan *web server* Apache HTTP yang terdapat pada komputer *server* (*provider entity*) dapat diakses dari komputer *client*

6.5.2. Pengujian Waktu Akses *Query*

A. Tujuan

- Pengujian dilakukan untuk mengetahui waktu yang dibutuhkan untuk melakukan *query* pada basis data `db_wse` melalui komputer *client*.
- Pengujian dilakukan untuk mendapatkan perbandingan waktu *query* yang dilakukan terhadap jumlah data yang berbeda pada basis data `db_wse`.

B. Prosedur Pengujian

PC *Client* :

- Mengakses `query_tes.php`. Halaman ini dibuat khusus untuk melakukan pengujian waktu akses *query* dengan menggunakan fungsi `microtime` pada php.
- Pengujian dilakukan pada tabel - tabel yang terdapat pada basis data `db_wse`.
- Melakukan perhitungan untuk mendapatkan nilai rata-rata waktu akses *query* yang dilakukan terhadap jumlah data yang berbeda pada basis data `db_wse`.

C. Proses Pengujian

- Hasil pengujian waktu akses *query* terhadap tabel `user` dengan jumlah data sebanyak 500 dan 1000 data *entry* ditunjukkan pada Tabel 6.70.

Tabel 6. 72 : Hasil pengujian waktu akses *query* terhadap tabel `user`

Jumlah Data	**500**	**1000**
Hasil Waktu *Query*	Konfigurasi: Query: "select count(*) as jumlah from user" Hasil Query: "500" Loop: 10 loop Jumlah data: 500 data Query 1: waktu = 0.000401973724365 ms Query 2: waktu = 0.000314950942993 ms Query 3: waktu = 0.00029182434082 ms Query 4: waktu = 0.000294923782349 ms Query 5: waktu = 0.000288009643555 ms Query 6: waktu = 0.000284910202026 ms Query 7:	Konfigurasi: Query: "select count(*) as jumlah from user" Hasil Query: "1000" Loop: 10 loop Jumlah data: 1000 data Query 1: waktu = 0.0721490383148 ms Query 2: waktu = 0.000487804412842 ms Query 3: waktu = 0.00034499168396 ms Query 4: waktu = 0.000323057174683 ms Query 5: waktu = 0.000324964523315 ms Query 6: waktu = 0.000318050384521 ms Query 7:

	waktu = 0.000440120697021 ms Query 8: waktu = 0.000368118286133 ms Query 9: waktu = 0.000298023223877 ms Query 10: waktu = 0.000302076339722 ms	waktu = 0.00031590461731 ms Query 8: waktu = 0.000324010848999 ms Query 9: waktu = 0.000313997268677 ms Query 10: waktu = 0.000324010848999 ms
Rata – Rata Waktu Query	0.000328493118286 ms	0.00752258300781 ms

Sumber : [Pengujian]

- Hasil pengujian waktu akses *query* terhadap tabel `kategori_user` dengan jumlah data sebanyak 500 dan 1000 data *entry* ditunjukkan pada Tabel 6.71.

Tabel 6. 73 : Hasil pengujian waktu akses *query* terhadap tabel `kategori_user`

Jumlah Data	**500**	**1000**
Hasil Waktu *Query*	Konfigurasi: Query: "select count(*) as jumlah from kategori_user" Hasil Query: "500" Loop: 10 loop Jumlah data: 500 data Query 1: waktu = 0.00315594673157 ms Query 2: waktu = 0.000515937805176 ms Query 3: waktu = 0.000365018844604 ms Query 4: waktu = 0.00029993057251 ms Query 5: waktu = 0.00655007362366 ms Query 6: waktu = 0.000347852706909 ms Query 7: waktu = 0.000293016433716 ms Query 8: waktu = 0.000284910202026 ms Query 9:	Konfigurasi: Query: "select count(*) as jumlah from kategori_user" Hasil Query: "1000" Loop: 10 loop Jumlah data: 1000 data Query 1: waktu = 0.0474469661713 ms Query 2: waktu = 0.000402927398682 ms Query 3: waktu = 0.00032114982605 ms Query 4: waktu = 0.000305891036987 ms Query 5: waktu = 0.000329971313477 ms Query 6: waktu = 0.00031304359436 ms Query 7: waktu = 0.000308990478516 ms Query 8: waktu = 0.000310897827148 ms Query 9:

	waktu = 0.00032114982605 ms Query 10: waktu = 0.000438928604126 ms	waktu = 0.000306129455566 ms Query 10: waktu = 0.00029993057251 ms
Rata – Rata Waktu Query	0.00125727653503 ms	0.00503458976746 ms

Sumber : [Pengujian]

- Hasil pengujian waktu akses *query* terhadap tabel barang dengan jumlah data sebanyak 500 dan 1000 data *entry* ditunjukkan pada Tabel 6.72.

Tabel 6. 74 : Hasil pengujian waktu akses *query* terhadap tabel barang

Jumlah Data	**500**	**1000**
Hasil Waktu *Query*	Konfigurasi: Query: "select count(*) as jumlah from barang" Hasil Query: "500" Loop: 10 loop Jumlah data: 500 data Query 1: waktu = 0.00740694999695 ms Query 2: waktu = 0.000388860702515 ms Query 3: waktu = 0.000298023223877 ms Query 4: waktu = 0.000321865081787 ms Query 5: waktu = 0.000483989715576 ms Query 6: waktu = 0.000300168991089 ms Query 7: waktu = 0.000324964523315 ms Query 8: waktu = 0.000288009643555 ms Query 9: waktu = 0.00028395652771 ms Query 10: waktu = 0.00682711601257 ms	Konfigurasi: Query: "select count(*) as jumlah from barang" Hasil Query: "1000" Loop: 10 loop Jumlah data: 1000 data Query 1: waktu = 0.000388860702515 ms Query 2: waktu = 0.000324964523315 ms Query 3: waktu = 0.000292062759399 ms Query 4: waktu = 0.000288009643555 ms Query 5: waktu = 0.000442028045654 ms Query 6: waktu = 0.000359058380127 ms Query 7: waktu = 0.000290155410767 ms Query 8: waktu = 0.000290870666504 ms Query 9: waktu = 0.000283002853394 ms Query 10: waktu = 0.000282049179077 ms
Rata – Rata	0.00169239044189 ms	0.000324106216431 ms

Waktu Query		

Sumber : [Pengujian]

- Hasil pengujian waktu akses *query* terhadap tabel `kategori_barang` dengan jumlah data sebanyak 500 dan 1000 data *entry* ditunjukkan pada Tabel 6.73.

Tabel 6. 75 : Hasil pengujian waktu akses *query* terhadap tabel `kategori_barang`

Jumlah Data	**500**	**1000**
Hasil Waktu *Query*	Konfigurasi: Query: "select count(*) as jumlah from kategori_barang" Hasil Query: "500" Loop: 10 loop Jumlah data: 500 data Query 1: waktu = 0.00317192077637 ms Query 2: waktu = 0.000571012496948 ms Query 3: waktu = 0.000312089920044 ms Query 4: waktu = 0.000318050384521 ms Query 5: waktu = 0.000284910202026 ms Query 6: waktu = 0.000281810760498 ms Query 7: waktu = 0.00029993057251 ms Query 8: waktu = 0.000285148620605 ms Query 9: waktu = 0.00028395652771 ms Query 10: waktu = 0.000290155410767 ms	Konfigurasi: Query: "select count(*) as jumlah from kategori_barang" Hasil Query: "1000" Loop: 10 loop Jumlah data: 1000 data Query 1: waktu = 0.00656390190125 ms Query 2: waktu = 0.000369071960449 ms Query 3: waktu = 0.000295162200928 ms Query 4: waktu = 0.000284910202026 ms Query 5: waktu = 0.000290870666504 ms Query 6: waktu = 0.000284194946289 ms Query 7: waktu = 0.000283002853394 ms Query 8: waktu = 0.000279903411865 ms Query 9: waktu = 0.000292062759399 ms Query 10: waktu = 0.000285148620605 ms
Rata – Rata Waktu Query	0.0006098985672 ms	0.000922822952271 ms

Sumber : [Pengujian]

- Hasil pengujian waktu akses *query* terhadap tabel `berat_kirim` dengan jumlah data sebanyak 500 dan 1000 data *entry* ditunjukkan pada Tabel 6.74.

Tabel 6. 76 : Hasil pengujian waktu akses *query* terhadap tabel `berat_kirim`

Jumlah Data	**500**	**1000**
Hasil Waktu *Query*	Konfigurasi: Query: "select count(*) as jumlah from berat_kirim" Hasil Query: "500" Loop: 10 loop Jumlah data: 500 data Query 1: waktu = 0.00323891639709 ms Query 2: waktu = 0.000393867492676 ms Query 3: waktu = 0.000306844711304 ms Query 4: waktu = 0.000319004058838 ms Query 5: waktu = 0.000291109085083 ms Query 6: waktu = 0.000285863876343 ms Query 7: waktu = 0.000296115875244 ms Query 8: waktu = 0.000288009643555 ms Query 9: waktu = 0.000286102294922 ms Query 10: waktu = 0.000410079956055 ms	Konfigurasi: Query: "select count(*) as jumlah from berat_kirim" Hasil Query: "1000" Loop: 10 loop Jumlah data: 1000 data Query 1: waktu = 0.000397920608521 ms Query 2: waktu = 0.000312805175781 ms Query 3: waktu = 0.000294923782349 ms Query 4: waktu = 0.000279903411865 ms Query 5: waktu = 0.000279903411865 ms Query 6: waktu = 0.000278949737549 ms Query 7: waktu = 0.000288009643555 ms Query 8: waktu = 0.000278949737549 ms Query 9: waktu = 0.000279903411865 ms Query 10: waktu = 0.000330924987793 ms
Rata – Rata Waktu Query	0.000611591339111 ms	0.000302219390869 ms

Sumber : [Pengujian]

- Hasil pengujian waktu akses *query* terhadap tabel `data_kirim` dengan jumlah data sebanyak 500 dan 1000 data *entry* ditunjukkan pada Tabel 6.75.

Tabel 6. 77 : Hasil pengujian waktu akses *query* terhadap tabel `data_kirim`

Jumlah Data	**500**	**1000**
Hasil Waktu *Query*	Konfigurasi: Query: "select count(*) as jumlah from data_kirim" Hasil Query: "500" Loop: 10 loop Jumlah data: 500 data Query 1: waktu = 0.00342988967896 ms Query 2: waktu = 0.000404119491577 ms Query 3: waktu = 0.000312089920044 ms Query 4: waktu = 0.000308990478516 ms Query 5: waktu = 0.000298976898193 ms Query 6: waktu = 0.000296115875244 ms Query 7: waktu = 0.000515937805176 ms Query 8: waktu = 0.000313997268677 ms Query 9: waktu = 0.000303030014038 ms Query 10: waktu = 0.000296115875244 ms	Konfigurasi: Query: "select count(*) as jumlah from data_kirim" Hasil Query: "1000" Loop: 10 loop Jumlah data: 1000 data Query 1: waktu = 0.000404834747314 ms Query 2: waktu = 0.000571012496948 ms Query 3: waktu = 0.000352144241333 ms Query 4: waktu = 0.000309944152832 ms Query 5: waktu = 0.000295162200928 ms Query 6: waktu = 0.000293016433716 ms Query 7: waktu = 0.000323057174683 ms Query 8: waktu = 0.000297069549561 ms Query 9: waktu = 0.000292062759399 ms Query 10: waktu = 0.00029993057251 ms
Rata – Rata Waktu Query	0.000647926330566 ms	0.000343823432922 ms

Sumber : [Pengujian]

- Hasil pengujian waktu akses *query* terhadap tabel `jasa_kirim` dengan jumlah data sebanyak 500 dan 1000 data *entry* ditunjukkan pada Tabel 6.76.

Tabel 6. 78 : Hasil pengujian waktu akses *query* terhadap tabel `jasa_kirim`

Jumlah Data	**500**	**1000**
Hasil Waktu *Query*	Konfigurasi: Query: "select count(*) as	Konfigurasi: Query: "select count(*) as

	jumlah from jasa_kirim" Hasil Query: "500" Loop: 10 loop Jumlah data: 500 data Query 1: waktu = 0.00342202186584 ms Query 2: waktu = 0.000401973724365 ms Query 3: waktu = 0.000304937362671 ms Query 4: waktu = 0.000295162200928 ms Query 5: waktu = 0.000301837921143 ms Query 6: waktu = 0.000289916992188 ms Query 7: waktu = 0.000292062759399 ms Query 8: waktu = 0.000300168991089 ms Query 9: waktu = 0.000290870666504 ms Query 10: waktu = 0.000291109085083 ms	jumlah from jasa_kirim" Hasil Query: "1000" Loop: 10 loop Jumlah data: 1000 data Query 1: waktu = 0.000388860702515 ms Query 2: waktu = 0.000306129455566 ms Query 3: waktu = 0.000333070755005 ms Query 4: waktu = 0.000278949737549 ms Query 5: waktu = 0.000277042388916 ms Query 6: waktu = 0.000284910202026 ms Query 7: waktu = 0.000278949737549 ms Query 8: waktu = 0.0002760887146 ms Query 9: waktu = 0.000409841537476 ms Query 10: waktu = 0.000298976898193 ms
Rata – Rata Waktu Query	0.000619006156921 ms	0.000313282012939 ms

Sumber : [Pengujian]

- Hasil pengujian waktu akses *query* terhadap tabel `order_user` dengan jumlah data sebanyak 500 dan 1000 data *entry* ditunjukkan pada Tabel 6.77.

Tabel 6. 79 : Hasil pengujian waktu akses *query* terhadap tabel `order_user`

Jumlah Data	**500**	**1000**
Hasil Waktu *Query*	Konfigurasi: Query: "select count(*) as jumlah from order_user" Hasil Query: "500" Loop: 10 loop	Konfigurasi: Query: "select count(*) as jumlah from order_user" Hasil Query: "1000" Loop: 10 loop

	Jumlah data: 500 data Query 1: waktu = 0.00453996658325 ms Query 2: waktu = 0.000388860702515 ms Query 3: waktu = 0.000298976898193 ms Query 4: waktu = 0.000305891036987 ms Query 5: waktu = 0.000288009643555 ms Query 6: waktu = 0.000282049179077 ms Query 7: waktu = 0.000288963317871 ms Query 8: waktu = 0.00028395652771 ms Query 9: waktu = 0.000285148620605 ms Query 10: waktu = 0.00028395652771 ms	Jumlah data: 1000 data Query 1: waktu = 0.000408887863159 ms Query 2: waktu = 0.000727891921997 ms Query 3: waktu = 0.00032901763916 ms Query 4: waktu = 0.000350952148438 ms Query 5: waktu = 0.000294923782349 ms Query 6: waktu = 0.000298023223877 ms Query 7: waktu = 0.000290155410767 ms Query 8: waktu = 0.000289916992188 ms Query 9: waktu = 0.000416994094849 ms Query 10: waktu = 0.000385999679565 ms
Rata – Rata Waktu Query	0.000724577903748 ms	0.000379276275635 ms

Sumber : [Pengujian]

- Hasil pengujian waktu akses *query* terhadap tabel `shopping_cart` dengan jumlah data sebanyak 500 dan 1000 data *entry* ditunjukkan pada Tabel 6.78.

Tabel 6. 80 : Hasil pengujian waktu akses *query* terhadap tabel `shopping_cart`

Jumlah Data	**500**	**1000**
Hasil Waktu *Query*	Konfigurasi: Query: "select count(*) as jumlah from shopping_cart" Hasil Query: "500" Loop: 10 loop Jumlah data: 500 data Query 1: waktu = 0.00348997116089 ms	Konfigurasi: Query: "select count(*) as jumlah from shopping_cart" Hasil Query: "1000" Loop: 10 loop Jumlah data: 1000 data Query 1: waktu = 0.000401973724365 ms

	Query 2: waktu = 0.000397920608521 ms Query 3: waktu = 0.000300168991089 ms Query 4: waktu = 0.000284910202026 ms Query 5: waktu = 0.000284910202026 ms Query 6: waktu = 0.000320196151733 ms Query 7: waktu = 0.000284194946289 ms Query 8: waktu = 0.000281095504761 ms Query 9: waktu = 0.00029182434082 ms Query 10: waktu = 0.00028395652771 ms	Query 2: waktu = 0.000314950942993 ms Query 3: waktu = 0.000288009643555 ms Query 4: waktu = 0.000281095504761 ms Query 5: waktu = 0.0002760887146 ms Query 6: waktu = 0.000541925430298 ms Query 7: waktu = 0.000293970108032 ms Query 8: waktu = 0.00029182434082 ms Query 9: waktu = 0.000279188156128 ms Query 10: waktu = 0.000273942947388 ms
Rata – Rata Waktu Query	0.000621914863586 ms	0.000324296951294 ms

Sumber : [Pengujian]

- Hasil pengujian waktu akses *query* terhadap tabel `status_bayar` dengan jumlah data sebanyak 500 dan 1000 data *entry* ditunjukkan pada Tabel 6.79.

Tabel 6. 81 : Hasil pengujian waktu akses *query* terhadap tabel `status_bayar`

Jumlah Data	**500**	**1000**
Hasil Waktu *Query*	Konfigurasi: Query: "select count(*) as jumlah from status_bayar" Hasil Query: "500" Loop: 10 loop Jumlah data: 500 data Query 1: waktu = 0.00302290916443 ms Query 2: waktu = 0.000570058822632 ms Query 3: waktu = 0.000314950942993 ms	Konfigurasi: Query: "select count(*) as jumlah from status_bayar" Hasil Query: "1000" Loop: 10 loop Jumlah data: 1000 data Query 1: waktu = 0.000424861907959 ms Query 2: waktu = 0.00031590461731 ms Query 3: waktu = 0.000286102294922 ms

	Query 4: waktu = 0.000295162200928 ms Query 5: waktu = 0.000285148620605 ms Query 6: waktu = 0.000285863876343 ms Query 7: waktu = 0.000289916992188 ms Query 8: waktu = 0.000281095504761 ms Query 9: waktu = 0.000281095504761 ms Query 10: waktu = 0.000283002853394 ms	Query 4: waktu = 0.000319957733154 ms Query 5: waktu = 0.000283002853394 ms Query 6: waktu = 0.000275135040283 ms Query 7: waktu = 0.0068039894104 ms Query 8: waktu = 0.000601053237915 ms Query 9: waktu = 0.000306129455566 ms Query 10: waktu = 0.000756025314331 ms
Rata – Rata Waktu Query	0.000590920448303 ms	0.00103721618652 ms

Sumber : [Pengujian]

- Hasil pengujian waktu akses *query* terhadap tabel `data_website` dengan jumlah data sebanyak 500 dan 1000 data *entry* ditunjukkan pada Tabel 6.80.

Tabel 6. 82 : Hasil pengujian waktu akses *query* terhadap tabel `data_website`

Jumlah Data	**500**	**1000**
Hasil Waktu *Query*	Konfigurasi: Query: "select count(*) as jumlah from data_website" Hasil Query: "500" Loop: 10 loop Jumlah data: 500 data Query 1: waktu = 0.00343799591064 ms Query 2: waktu = 0.000387191772461 ms Query 3: waktu = 0.000304937362671 ms Query 4: waktu = 0.000283002853394 ms Query 5: waktu = 0.000282049179077 ms	Konfigurasi: Query: "select count(*) as jumlah from data_website" Hasil Query: "1000" Loop: 10 loop Jumlah data: 1000 data Query 1: waktu = 0.000396013259888 ms Query 2: waktu = 0.000473976135254 ms Query 3: waktu = 0.00036096572876 ms Query 4: waktu = 0.000289916992188 ms Query 5: waktu = 0.000840902328491 ms

	Query 6: waktu = 0.000463008880615 ms Query 7: waktu = 0.000296831130981 ms Query 8: waktu = 0.000282049179077 ms Query 9: waktu = 0.000282049179077 ms Query 10: waktu = 0.000279903411865 ms	Query 6: waktu = 0.0065929889679 ms Query 7: waktu = 0.00049901008606 ms Query 8: waktu = 0.000331163406372 ms Query 9: waktu = 0.000287055969238 ms Query 10: waktu = 0.000458955764771 ms
Rata – Rata Waktu Query	0.000629901885986 ms	0.00105309486389 ms

Sumber : [Pengujian]

- Hasil pengujian waktu akses *query* terhadap tabel `kategori_data_website` dengan jumlah data sebanyak 500 dan 1000 data *entry* ditunjukkan pada Tabel 6.81.

Tabel 6. 83 : Hasil pengujian waktu akses *query* terhadap tabel `kategori_data_website`

Jumlah Data	**500**	**1000**
Hasil Waktu *Query*	Konfigurasi: Query: "select count(*) as jumlah from kategori_data_website" Hasil Query: "500" Loop: 10 loop Jumlah data: 500 data Query 1: waktu = 0.00948095321655 ms Query 2: waktu = 0.000448942184448 ms Query 3: waktu = 0.0060818195343 ms Query 4: waktu = 0.000416994094849 ms Query 5: waktu = 0.000304222106934 ms Query 6: waktu = 0.000771999359131 ms Query 7:	Konfigurasi: Query: "select count(*) as jumlah from kategori_data_website" Hasil Query: "1000" Loop: 10 loop Jumlah data: 1000 data Query 1: waktu = 0.116312980652 ms Query 2: waktu = 0.000393152236938 ms Query 3: waktu = 0.000362157821655 ms Query 4: waktu = 0.0011830329895 ms Query 5: waktu = 0.000329971313477 ms Query 6: waktu = 0.000311136245728 ms Query 7:

	waktu = 0.000365972518921 ms Query 8: waktu = 0.000298976898193 ms Query 9: waktu = 0.000286817550659 ms Query 10: waktu = 0.000290155410767 ms	waktu = 0.00029993057251 ms Query 8: waktu = 0.000302076339722 ms Query 9: waktu = 0.000304937362671 ms Query 10: waktu = 0.000303030014038 ms
Rata – Rata Waktu Query	0.00187468528748 ms	0.0120102405548 ms

Sumber : [Pengujian]

- Hasil pengujian waktu akses *query* terhadap tabel `web_service` dengan jumlah data sebanyak 500 dan 1000 data *entry* ditunjukkan pada Tabel 6.82.

Tabel 6. 84 : Hasil pengujian waktu akses *query* terhadap tabel `web_service`

Jumlah Data	**500**	**1000**
Hasil Waktu *Query*	Konfigurasi: Query: "select count(*) as jumlah from web_service" Hasil Query: "500" Loop: 10 loop Jumlah data: 500 data Query 1: waktu = 0.0097279548645 ms Query 2: waktu = 0.000391960144043 ms Query 3: waktu = 0.000501871109009 ms Query 4: waktu = 0.000300168991089 ms Query 5: waktu = 0.000412225723267 ms Query 6: waktu = 0.000343084335327 ms Query 7: waktu = 0.000288963317871 ms Query 8: waktu = 0.000289916992188 ms Query 9:	Konfigurasi: Query: "select count(*) as jumlah from web_service" Hasil Query: "1000" Loop: 10 loop Jumlah data: 1000 data Query 1: waktu = 0.000401973724365 ms Query 2: waktu = 0.000309944152832 ms Query 3: waktu = 0.000294923782349 ms Query 4: waktu = 0.000283002853394 ms Query 5: waktu = 0.000280857086182 ms Query 6: waktu = 0.000607967376709 ms Query 7: waktu = 0.000532865524292 ms Query 8: waktu = 0.000342845916748 ms Query 9:

	waktu = 0.000289916992188 ms Query 10: waktu = 0.000284910202026 ms	waktu = 0.000281095504761 ms Query 10: waktu = 0.000280141830444 ms
Rata – Rata Waktu Query	0.00128309726715 ms	0.000361561775208 ms

Sumber : [Pengujian]

- Hasil pengujian waktu akses *query* terhadap tabel `kategori_webservice` dengan jumlah data sebanyak 500 dan 1000 data *entry* ditunjukkan pada Tabel 6.83.

Tabel 6. 85 : Hasil pengujian waktu akses *query* terhadap tabel `kategori_webservice`

Jumlah Data	**500**	**1000**
Hasil Waktu *Query*	Konfigurasi: Query: "select count(*) as jumlah from kategori_webservice" Hasil Query: "500" Loop: 10 loop Jumlah data: 500 data Query 1: waktu = 0.00335097312927 ms Query 2: waktu = 0.000401020050049 ms Query 3: waktu = 0.000309944152832 ms Query 4: waktu = 0.000300168991089 ms Query 5: waktu = 0.00033712387085 ms Query 6: waktu = 0.000550031661987 ms Query 7: waktu = 0.000319957733154 ms Query 8: waktu = 0.000297069549561 ms Query 9: waktu = 0.000293970108032 ms Query 10: waktu = 0.000293970108032 ms	Konfigurasi: Query: "select count(*) as jumlah from kategori_webservice" Hasil Query: "1000" Loop: 10 loop Jumlah data: 1000 data Query 1: waktu = 0.000411033630371 ms Query 2: waktu = 0.000488996505737 ms Query 3: waktu = 0.000385999679565 ms Query 4: waktu = 0.000302791595459 ms Query 5: waktu = 0.000429153442383 ms Query 6: waktu = 0.000300884246826 ms Query 7: waktu = 0.000413179397583

		ms Query 8: waktu = 0.000360012054443 ms Query 9: waktu = 0.000298976898193 ms Query 10: waktu = 0.00029993057251 ms
Rata – Rata Waktu Query	0.000645422935486 ms	0.000369095802307 ms

Sumber : [Pengujian]

- Hasil pengujian waktu akses *query* terhadap tabel `propinsi` dengan jumlah data sebanyak 500 dan 1000 data *entry* ditunjukkan pada Tabel 6.84.

Tabel 6. 86 : Hasil pengujian waktu akses *query* terhadap tabel `propinsi`

Jumlah Data	**500**	**1000**
Hasil Waktu *Query*	Konfigurasi: Query: "select count(*) as jumlah from propinsi" Hasil Query: "500" Loop: 10 loop Jumlah data: 500 data Query 1: waktu = 0.00331807136536 ms Query 2: waktu = 0.000391960144043 ms Query 3: waktu = 0.000296115875244 ms Query 4: waktu = 0.00656485557556 ms Query 5: waktu = 0.000604152679443 ms Query 6: waktu = 0.000303983688354 ms Query 7: waktu = 0.000286102294922 ms Query 8:	Konfigurasi: Query: "select count(*) as jumlah from propinsi" Hasil Query: "1000" Loop: 10 loop Jumlah data: 1000 data Query 1: waktu = 0.00164103507996 ms Query 2: waktu = 0.000415086746216 ms Query 3: waktu = 0.000308036804199 ms Query 4: waktu = 0.000288009643555 ms Query 5: waktu = 0.000280141830444 ms Query 6: waktu = 0.00028395652771 ms Query 7: waktu = 0.000272989273071 ms Query 8:

	waktu = 0.000288963317871 ms Query 9: waktu = 0.000280857086182 ms Query 10: waktu = 0.000280141830444 ms	waktu = 0.000274896621704 ms Query 9: waktu = 0.000572919845581 ms Query 10: waktu = 0.000417947769165 ms
Rata – Rata Waktu Query	0.00126152038574 ms	0.00047550201416 ms

Sumber : [Pengujian]

- Hasil pengujian waktu akses *query* terhadap tabel `kota` dengan jumlah data sebanyak 500 dan 1000 data *entry* ditunjukkan pada Tabel 6.85.

Tabel 6. 87 : Hasil pengujian waktu akses *query* terhadap tabel `kota`

Jumlah Data	**500**	**1000**
Hasil Waktu *Query*	Konfigurasi: Query: "select count(*) as jumlah from kota" Hasil Query: "500" Loop: 10 loop Jumlah data: 500 data Query 1: waktu = 0.0098888874054 ms Query 2: waktu = 0.000388860702515 ms Query 3: waktu = 0.00029993057251 ms Query 4: waktu = 0.000292062759399 ms Query 5: waktu = 0.000286102294922 ms Query 6: waktu = 0.000282764434814 ms Query 7: waktu = 0.000318050384521 ms Query 8: waktu = 0.000445127487183 ms Query 9: waktu = 0.000298023223877 ms Query 10: waktu = 0.000309944152832 ms	Konfigurasi: Query: "select count(*) as jumlah from kota" Hasil Query: "1000" Loop: 10 loop Jumlah data: 1000 data Query 1: waktu = 0.000468969345093 ms Query 2: waktu = 0.00031590461731 ms Query 3: waktu = 0.000770092010498 ms Query 4: waktu = 0.0030210018158 ms Query 5: waktu = 0.000322103500366 ms Query 6: waktu = 0.000294923782349 ms Query 7: waktu = 0.000286102294922 ms Query 8: waktu = 0.000282049179077 ms Query 9: waktu = 0.000469923019409 ms Query 10: waktu = 0.000298976898193 ms

Rata – Rata Waktu Query	0.0012809753418 ms	0.000653004646301 ms

Sumber : [Pengujian]

D. Hasil Pengujian dan Analisis

- Basis data db_wse pada komputer *server* dapat menangani permintaan *query* dari komputer *client* melalui jaringan komputer dengan jumlah 1000 data, dengan waktu akses *query* rata-rata terlama adalah 0.012 ms.

Tabel 6. 88 : Tabel rata-rata pengujian waktu akses *query*

Nama Tabel	waktu akses dengan 500 data entry (ms)	waktu akses dengan 1000 data entry (ms)
user	0.000328493118286	0.00752258300781
kategori_user	0.00125727653503	0.00503458976746
barang	0.00169239044189	0.000324106216431
kategori_barang	0.0006098985672	0.000922822952271
berat_kirim	0.000611591339111	0.000302219390869
data_kirim	0.000647926330566	0.000343823432922
jasa_kirim	0.000619006156921	0.000313282012939
order_user	0.00072457790374 8	0.000379276275635
shopping_cart	0.000621914863586	0.000324296951294
status_bayar	0.000590920448303	0.00103721618652
data_website	0.000629901885986	0.00105309486389
kategori_data_website	0.00187468528748	0.0120102405548
web_service	0.00128309726715	0.000361561775208
kategori_webservice	0.000645422935486	0.000369095802307
propinsi	0.00126152038574	0.00047550201416
kota	0.0012809753418	0.000653004646301

Sumber : [pengujian]

6.5.3. Pengujian Performansi *Web Server*

Pengujian performansi *web server* dilakukan untuk mengetahui kinerja dari kemampuan *web server* untuk menangani *request* dari *client*. Tabel 6.90 menerangkan *hardware* yang digunakan untuk diuji kemampuannya sebagai *web server*.

Tabel 6. 89 : Spesifikasi perangkat keras untuk *server* Aplikasi *Web Service E-Commerce*

Nama Komponen	Spesifikasi
Prosesor	Intel® Pentium® IV 2.66 Hz
Memori (RAM)	1.024 MB
Hardisk	Maxtor ATA 80 GB
Motherboard	ASUSTek Computer Inc. P5PE-VM
VGA Card	ATI Radeon 9550 AGP 256 MB

Sumber : [Implementasi]

Untuk mengetahui kinerja dari kemampuan *web server* digunakan dua alat bantu yaitu httperf dan autobench, sekilas penjelasan mengenai masing alat bantu tersebut :

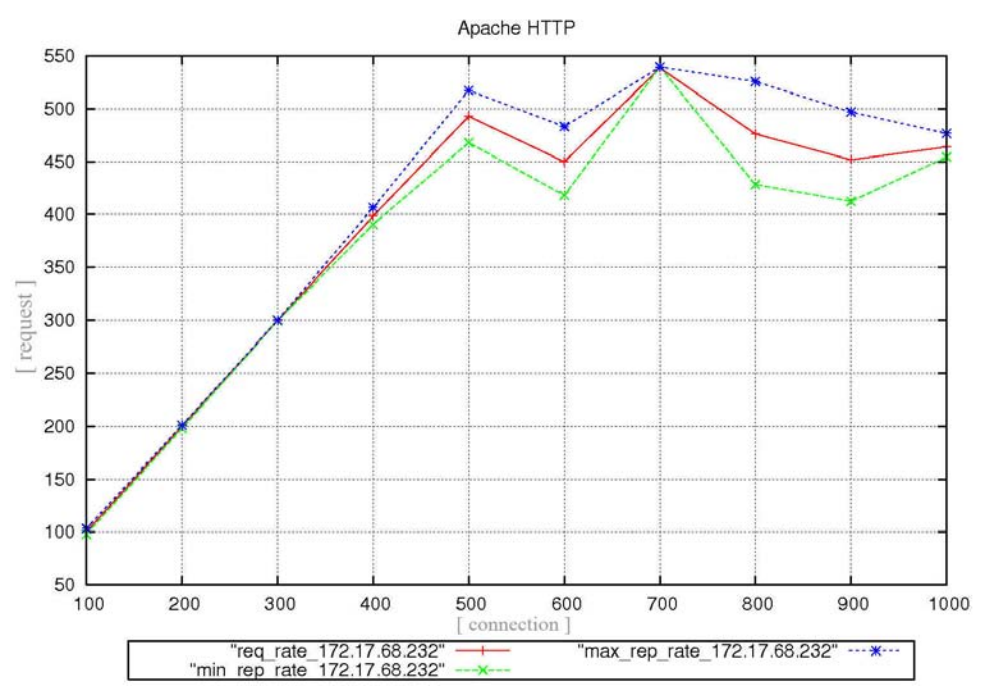
1. Httperf

Httperf adalah alat bantu untuk menguji performansi dari *web server*. Menyediakan berbagai fasilitas untuk menghasilkan berbagai macam beban kerja pada pengukuran terhadap kinerja *web server*.

2. Autobench

Adalah alat bantu untuk mengotomasi proses *benchmark* dari *web server*, dan digunakan bersama httperf untuk menghasilkan grafik hasil pengukuran.

Gambar 6.11 menunjukkan pengujian performansi *web server*, dengan 100 sampai 1000 koneksi yang me-*request* secara bersamaan. Berikut adalah hasil pengujian dari httperf dan autobench :



Gambar 6. 11 : Pengujian Performansi *Web Server* Apache HTTP
Sumber : [Pengujian]

Httperf dan Autobench tidak dijalankan pada komputer yang menjadi *server* melainkan dijalankan pada sebuah komputer *client* supaya tidak mempengaruhi hasil pengujian, dan diterapkan pada jaringan lokal sesuai dengan tahapan implementasi Aplikasi *Web Service E-Commerce* yang diterapkan pada jaringan lokal.

Parameter yang diberikan untuk pengujian ini adalah dengan memberi kasus uji dimana httperf membuka antara 100 koneksi sampai dengan 1000 koneksi dengan ketentuan apabila *server* tidak merespon dalam waktu 10 detik maka akan dihitung sebagai *error*.

Berdasarkan gambar 6.11 tersebut, performansi *web server* Apache HTTP mengalami dua kali kenaikan, yakni pada titik (500,480) dan titik (700,540). Dari data tersebut didapatkan hasil bahwa *web server* Apache HTTP memiliki performansi terbaik pada saat terdapat 700 koneksi dengan 540 *request* yang dilakukan oleh *client* secara bersamaan.

BAB VII
PENUTUP

7.1. Kesimpulan

Aplikasi *Web Service E-Commerce* dapat berfungsi sesuai dengan perancangan dan pengujian yang ditunjukkan melalui :

1. Aplikasi berbasis *web* dengan menggunakan bahasa pemrograman PHP dengan basis data MySQL untuk menangani proses pada sebuah aplikasi *Web Service E-Commerce* telah berhasil dirancang sesuai dengan analisis kebutuhan, dan dikembangkan menggunakan pendekatan berorientasi obyek.
2. Aplikasi *web service* dengan menggunakan bahasa pemrograman XML *Web Service* (SOAP dan WSDL) untuk menangani modul pencarian data, katalog data, dan *review* data yang mendukung aplikasi *E-Commerce* telah berhasil dirancang, yang kemudian diimplementasikan dan diujikan pada aplikasi *partner*.
3. Analisis kebutuhan dari sistem yang dirancang menghasilkan 43 *use case* dalam daftar kebutuhan, dan dapat digunakan sebagai acuan perancangan Aplikasi *Web Service E-Commerce*.
4. Pengujian validasi menggunakan 68 kasus uji telah berhasil dilakukan dengan hasil tanpa kesalahan sama sekali.
5. Pengujian koneksi *server* basis data MySQL dan *web server* Apache HTTP yang terjadi antara *server* (*provider entity*) dengan *partner* (*requester entity*) dan antara *server* (*provider entity*) dengan *client* dapat terhubung dengan baik melalui protokol TCP/IP.
6. Pengujian waktu akses *query* pada basis data `db_wse` dengan 500 dan 1000 data *entry* memberikan hasil bahwa rata – rata waktu akses *query* terlama adalah 0.012 ms dengan 1000 data *entry*.
7. Pengujian performansi *web server* Apache HTTP pada komputer *server* memiliki performansi terbaik pada 700 koneksi dengan 540 *request* yang dilakukan oleh *client* secara bersamaan.

7.2. Saran

Saran yang dapat diberikan untuk pengembangan Aplikasi *Web Service E-Commerce* antara lain:

1. Aplikasi *Web Service E-Commerce* dapat digunakan untuk menjalin kerjasama antar situs – situs *E-Commerce* yang ada di internet berupa integrasi aplikasi, guna meningkatkan performa pelayanan perdagangan *online* kepada masyarakat.
2. Aplikasi *Web Service E-Commerce* dapat menggunakan *web server* selain Apache HTTP guna meningkatkan performansi.
3. Keamanan Aplikasi *Web Service E-Commerce* dapat dikembangkan dengan tidak hanya menggunakan keamanan standar yang telah tersedia pada bahasa pemrograman PHP dan XML *Web Service*, atau basis data MySQL.

DAFTAR PUSTAKA

[SIS-04] Siswoutomo, Wiwit, 2004, "Membangun *Web Service Open Source* Menggunakan PHP", PT Elex Media Komputindo.

[AND-08] Andriana, Dian, "Pengenalan Pemrograman *E-Commerce* dengan PHP dan MySQL", `http://ilmukomputer.com/2006/08/25/pemrograman-e-commerce-dengan-php-dan-mysql`. Diakses tanggal 31-Agustus-2008 pukul 20:19.

[KEN-06] Kennedy, Bill, and Musciano, Chuck, 2006,"HTML & XHTML: *The Definitive Guide*, 6th *Edition*", O'Reilly Media, Inc.

[OLS-09] Olson, Philip, 2009, "PHP Manual", the PHP Documentation Group.

[BEY-04] Beynon-Davies, Paul, 2004, "*Database Systems, Third Edition*", Palgrave Macmillan.

[SIK-05] Silberschatz, Avi, Korth, Henry F., 2005,"*Database System Concept, Fifth Editon*", McGraw-Hill.

[Anonymous-08] Anonymous, 2008, "MySQL 5.1 *Reference Manual*".

[PRE-05] Pressman, Roger S., 2005, "*Software engineering : A Practitioner's Approach. 6th edition*", McGraw-Hill.

[BOR-05] Booch, Grady, Rumbaugh, James, Jacobson, Ivar, 2005, "*The Unified Modeling Language User Guide Second Edition*", Addison Wesley.

[WUW-08] Wulandari, Lily, dan Wicaksana, I Wayan Simri, "*Toward Web Service*", `http://iwayan.staff.gunadarma.ac.id/Publications/files/711/2006_Kommit_TowardWebServ_IWS.pdf`, diakses tanggal 20-

November-2008 pukul 16:50.

[RAH-08] Rahmawati, Mega Puspita, "*Electronic Commerce*", http://bebas.vlsm.org/v06/Kuliah/Seminar-MIS/2007/208/208-11-E-Commerce.pdf, diakses tanggal 03-September-2008 pukul 16:04.

[COG-04] Coggeshall, John, 2004, "PHP *Unleashed*", Sams Publishing.