PENGEMBANGAN PORTAL *MOBILE ONLINE SUPERMARKET* DENGAN TEKNOLOGI XHTML MP DAN WCSS

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Teknik

Disusun Oleh :

JATI RAKHMAWATI
NIM. 0410633045-63

KEMENTRIAN PENDIDIKAN NASIONAL
UNIVERSITAS BRAWIJAYA
FAKULTAS TEKNIK
MALANG
2010

repository.ub.ac.id
UNIVERSITAS BRAWIJAYA

PENGEMBANGAN PORTAL *MOBILE ONLINE SUPERMARKET* DENGAN TEKNOLOGI XHTML MP DAN WCSS

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Teknik

Disusun oleh :

JATI RAKHMAWATI
NIM. 0410633045-63

Telah diperiksa dan disetujui oleh :

Dosen Pembimbing I	Dosen Pembimbing II
Ir. Heru Nurwarsito, MKom.	**Suprapto, ST., MT.**
NIP. 19650402 199002 1 001	**NIP. 19710727 199603 1 001**

repository.ub.ac.id
UNIVERSITAS BRAWIJAYA

LEMBAR PENGESAHAN

PENGEMBANGAN PORTAL *MOBILE ONLINE SUPERMARKET* DENGAN TEKNOLOGI XHTML MP DAN WCSS

SKRIPSI

KONSENTRASI TEKNIK INFORMATIKA DAN KOMPUTER

Diajukan untuk memenuhi persyaratan
memperoleh gelar Sarjana Teknik

Disusun oleh :

JATI RAKHMAWATI
NIM. 0410633045-63

Skripsi ini telah diuji dan dinyatakan lulus pada
tanggal 11 Pebruari 2010

Majelis Penguji:

Ir. Muhammad Aswin, MT.
NIP. 19640626 199002 1 001

Waru Djuriatno, ST., MT.
NIP. 19690725 199702 1 001

Adharul Muttaqin, ST., MT.
NIP. 19760121 200501 1 001

Mengetahui:
Ketua Jurusan Teknik Elektro

Rudy Yuwono, ST., MSc.
NIP. 19710615 199802 1 003

repository.ub.ac.id

Teriring ucapan terima kasih kepada :
Bapak dan ibu tercinta

UNIVERSITAS BRAWIJAYA

KATA PENGANTAR

Alhamdulillah, segala puji bagi Allah SWT semata, sholawat serta salam semoga terlimpahkan untuk Nabi Muhammad SAW, beserta para sahabat yang setia mengikuti jejak beliau. Berkat hidayah dan inayah Allah SWT, skripsi berjudul "Pengembangan Portal *Mobile Online Supermarket* dengan Teknologi XHTML MP dan WCSS" dapat diselesaikan.

Ucapan terima kasih yang mendalam penulis sampaikan kepada pihak-pihak yang telah membantu secara langsung maupun tidak langsung, dan telah memberikan dukungan sehingga penyusunan skripsi ini dapat selesai pada waktunya. Atas dukungan dan bimbingannya, pada kesempatan yang berbahagia ini, penulis mengucapkan banyak terima kasih kepada :

1. Ayahanda Drs. Syawal dan Ibunda Dra. Sa'diyah tercinta yang telah memberikan doa, motivasi dan dukungan material, serta adikku tersayang Indah Kusuma Wardani.
2. Bapak Rudy Yuwono, ST, Msc., selaku Ketua Jurusan Teknik Elektro.
3. Bapak M. Azis Muslim, ST., MT.,Ph.D, selaku Sekretaris Jurusan Teknik Elektro.
4. Bapak Heru Nurwarsito, Ir., MKom., selaku Dosen Pembimbing I, Bapak Suprapto, ST., MT., selaku Dosen Pembimbing II, dan Bapak Waru Djuriatno, ST., MT., selaku Ketua Kelompok Dosen Keahlian (KKDK) Konsentrasi Teknik Informatika dan Komputer yang telah meluangkan waktu, tenaga, dan pikiran untuk membimbing penulis dalam penyusunan skripsi ini.
5. Ibu Endah Budi Purnomowati, Ir., MT., selaku Dosen Wali yang telah membimbing penulis.
6. Bapak Herman Tolle, ST., MT., yang telah memberikan judul dan bimbingan *online*.
7. Seluruh Dosen dan Staf Administrasi Jurusan Teknik Elektro.
8. Tulus, Prima, Riza, Mas Diriga, Ajenk, Dieu, Kiki atas bantuan, doa, dan semangat.
9. Semua teman Generator dan Teknik Elektro.
10. Semua pihak yang tidak dapat penulis sebutkan satu persatu, yang telah membantu dalam penyelesaian skripsi ini.

Penulis menyadari bahwa skripsi ini masih jauh dari sempurna. Oleh karenanya penulis sangat mengharapkan masukan berupa kritik dan saran yang membangun. Semoga skripsi ini memberikan manfaat bagi semua pihak sesuai dengan fungsinya.

Malang, 4 Pebruari 2010

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

ABSTRAK

JATI RAKHMAWATI. 2010. : Pengembangan Portal *Mobile Online Supermarket* dengan Teknologi XHTML MP dan WCSS. Skripsi Jurusan Teknik Elektro, Fakultas Teknik, Universitas Brawijaya. Dosen Pembimbing : Heru Nurwarsito, Ir., Mkom. dan Suprapto, ST., MT.

Supermarket merupakan tempat belanja untuk memenuhi kebutuhan sehari-hari. Banyaknya waktu yang dibutuhkan untuk menuju ke *supermarket* dan mengantri di loket kasir, sehingga tidak semua orang mempunyai waktu disela-sela kesibukannya untuk membeli barang kebutuhannya. Media pesan antar barang kebutuhan tanpa menghabiskan banyak waktu adalah dengan *portal mobile online supermarket*.

Proses pengembangan perangkat lunak ini menggunakan paradigma *object* oriented (OO) dengan pemodelan UML. Perangkat lunak ini dapat dibuka oleh *user* menggunakan *mobile browser* melalui perangkat *mobile* yang terkoneksi dengan GPRS kapanpun dan dimanapun, atau menggunakan *internet browser* melalui sebuah komputer yang terkoneksi dengan *internet*. Implementasi sistem menggunakan teknologi XHTML MP dan WCSS yang dapat dibaca melalui *internet browser* maupun *mobile browser*, PHP untuk melakukan komunikasi dengan basis data, dan MySQL untuk menyimpan data. PHP dipilih karena dapat membuat website menjadi lebih dinamis dan interaktif terhadap *user*. MySQL dipilih karena basis data ini cocok untuk sistem yang membutuhkan basis data besar, selain itu juga MySQL adalah suatu bahasa berbasis SQL yang dikembangkan secara *free* dan *open source*.

Perangkat lunak diuji dengan menggunakan pengujian unit, pengujian integrasi, pengujian validasi, dan pengujian waktu akses *query*. Hasil pengujian menunjukkan bahwa sistem yang dibangun adalah valid sesuai dengan kebutuhan dari *user*. Hasil pengujian waktu akses *query* menunjukkan perbedaan pengujian kecepatan waktu akses *query* yang dilakukan secara *offline* dan *online* ini terjadi karena faktor *web server* yang digunakan dan faktor jaringan.

Kata Kunci : *supermarket*, perangkat *mobile, mobile browser*, XHTML MP, WCSS, PHP, MySQL

BAB I
PENDAHULUAN

1.1. Latar Belakang

Supermarket merupakan tempat belanja untuk memenuhi kebutuhan sehari-hari, dari kebutuhan rumah tangga, sekolah, sampai kantor dimana pelayanan dilakukan sendiri oleh konsumen dan minimal memiliki 3 mesin *register*. *Supermarket* berukuran 1.000 m2 sampai dengan 4.999 m2. [AN1-08]. Saat berbelanja di *supermarket* konsumen dapat memilih, mengambil, membandingkan, dan membeli barang kebutuhan sesuai keinginannya sendiri. Setelah mendapatkan barang yang diinginkan konsumen menuju kasir untuk melakukan pembayaran. Namun sering terjadi antrian pada loket kasir, sehingga dapat menimbulkan ketidaknyamanan pada konsumen karena waktu yang terbuang untuk mengantri.

Banyaknya waktu yang dibutuhkan untuk menuju ke *supermarket* dan mengantri di loket kasir, sehingga tidak semua orang mempunyai waktu disela-sela kesibukannya untuk membeli barang kebutuhannya. Karena itu perlu dibuat suatu sarana pesan antar barang kebutuhan. Media pesan antar barang kebutuhan tanpa bergantung pada waktu adalah dengan *portal mobile online supermarket* menggunakan teknologi XHTML MP (*Extensible HyperText Markup Language Mobile Profile*) dan WCSS (*Wireless Aplication Protocol Cascading Style Sheets*).

Diperkirakan saat ini jumlah pelanggan telepon seluler telah melampaui jumlah pelanggan *internet*. Pesatnya pengguna telepon seluler menunjukkan betapa tingginya kebutuhan masyarakat terhadap perangkat komunikasi seluler. Ini berarti bahwa aplikasi *mobile online* di masa akan datang juga akan berkembang pesat. XHTML MP akan menjadi standar penulisan bahasa *markup* untuk *mobile internet*. Ditambahkan dengan WCSS yang memungkinkan sebuah situs dibangun dengan lebih fleksibel, karena dengan WCSS perubahan terhadap tata letak situs dapat dilakukan dengan mengubah *style sheet* tanpa mengubah isi dari situs tersebut. WCSS juga dapat dibaca melalui *internet browser* maupun *mobile browser* seperti pada XHTML MP. [JUS-07]

Berbelanja di *supermarket mobile online* ini konsumen tidak memerlukan banyak waktu, dapat dilakukan disela-sela kesibukan untuk membuka *website*-nya dengan menggunakan perangkat *mobile*. Untuk menjadi konsumen, terlebih dahulu

pengunjung melakukan registrasi yang kemudian divalidasi oleh *operator*. Setelah status *valid* konsumen dapat menggunakan *username* dan *password*-nya untuk login dan melakukan pemesanan barang. Setelah konsumen memesan barang, barang akan diantarkan ke alamat konsumen dan melakukan pembayaran saat itu juga.

Dengan *portal mobile online supermarket* yang menggunakan teknologi XHTML MP dan WCSS ini memberi alternatif belanja *online* yaitu konsumen dapat melihat dan memesan barang melalui *web* dengan perangkat-perangkat *mobile* tanpa menghabiskan banyak waktu.

1.2. Rumusan Masalah

Selain permasalahan yang telah diungkapkan sebelumnya, permasalahan dalam sistem *mobile online supermarket* dapat memberikan:

a. Bagaimana membuat aplikasi *web* dengan teknologi XHTML MP dan WCSS untuk sistem *mobile online supermarket*?
b. Bagaimana merancang dan mengimplementasikan basis data untuk sistem *mobile online supermarket*?
c. Bagaimana mengimplementasikan dan menguji sistem *mobile online supermarket*?

1.3. Batasan Masalah

Untuk memberi arah yang jelas dalam permasalahan ini maka diperlukan batasan masalah sebagai berikut:

a. Membuat aplikasi untuk mengakses menu-menu yang ada pada sistem *mobile online supermarket* ini. Halaman-halaman untuk konsumen disesuaikan untuk tampilan di *mobile browser*. Halaman-halaman untuk administrator yang disesuaikan untuk tampilan di *internet browser*, begitu juga dengan halaman-halaman untuk operator.
b. Aplikasi ini bagi *user* yang tidak *login* hanya menyediakan layanan melihat barang yang ditawarkan. Bagi konsumen hanya menyediakan fasilitas untuk pemesanan barang. Bagi administrator hanya menyediakan fasilitas untuk memanagemen barang, kategori, konsumen, pegawai, dan suplier. Bagi operator hanya menyediakan fasilitas untuk memanagemen validasi konsumen, pemesanan, dan mencetak nota.

c. Pemesanan barang di *mobile online supermarket* ini hanya ditujukan di satu kota, dalam skripsi ini dikhususkan untuk konsumen di kota Malang.
d. Sistem operasi yang digunakan Microsoft Windows XP *Professional Service Pack* 2.
e. Menggunakan *emulator* Nokia *Series 40 5th Edition SDK* dan *web server* Apache 2.2.9.
f. *Browser* yang digunakan *internet browser* dan *mobile browser*.
g. Bahasa pemrograman yang digunakan XHTML MP, WCSS, dan PHP 5.2.6. Untuk *database* menggunakan MySQL 5.0.67.
h. Bukti transaksi berupa nota yang dicetak.

1.4. Tujuan

Tujuan dari tugas akhir ini adalah mengembangkan *portal mobile online supermarket* untuk pesan antar barang dengan menggunakan teknologi XHTML MP dan WCSS.

1.5. Manfaat

Adapun manfaatnya dengan adanya *portal mobile online supermarket* yang menggunakan teknologi XHTML MP dan WCSS ini memberi alternatif belanja *online* yaitu konsumen dapat melihat, memesan, dan membeli barang melalui *web* dengan perangkat-perangkat *mobile* tanpa batasan waktu.

1.6. Sistematika Penulisan

Pembahasan tugas akhir ini terdiri dari 7 bab yang disusun dengan sistematika penulisan sebagai berikut:

Bab I Pendahuluan

Memuat latar belakang, rumusan masalah, batasan masalah, tujuan, manfaat serta sistematika penulisan.

Bab II Dasar Teori

Dalam bab ini akan menjelaskan tentang teori-teori yang menunjang penulisan laporan tugas akhir ini, seperti basis data dan bahasa pemrograman yang akan dipakai.

Bab III Metodologi

Membahas metodologi yang digunakan dalam penulisan yang terdiri dari studi literatur, analisis dan perancangan, implementasi, pengujian, pengambilan kesimpulan, dan penulisan laporan.

Bab IV Perancangan

Membahas analisis kebutuhan dan perancangan sistem yang sesuai dengan teori yang ada serta membahas pembuatan sistem yang dirancang.

Bab V Implementasi

Membahas lingkungan implementasi, batasan implementasi, *file-file* implementasi, algoritma operasi yang diimplementasikan.

Bab VI Pengujian

Membahas pengujian basis data dan sistem yang diimplementasikan.

Bab VII Penutup

Memuat kesimpulan yang diperoleh dari pembuatan dan pengujian program, serta saran-saran untuk pengembangan lebih lanjut**.**

BAB II
DASAR TEORI

Bab ini menjelaskan dasar teori yang digunakan untuk menunjang penulisan tugas akhir dan pengembangan perangkat lunak, yang meliputi *supermarket*, basis data, UML, MySQL, PHP, XHTML MP, WCSS, dan pengujian perangkat lunak.

2.1. Supermarket

Supermarket berukuran 1.000 m2 sampai dengan 4.999 m2. [AN1-08]. Merupakan tempat belanja untuk memenuhi kebutuhan sehari-hari, dari kebutuhan rumah tangga, sekolah, sampai kantor dimana pelayanan dilakukan sendiri oleh konsumen dan minimal memiliki 3 mesin *register*. Saat berbelanja di *supermarket* konsumen dapat memilih, mengambil, membandingkan, dan membeli barang kebutuhan sesuai keinginannya sendiri. Pelayanan dilakukan sendiri oleh konsumen karena tidak menyediakan pramuniaga, tidak ada tawar-menawar, tidak bisa saling tegur sapa, dan tidak ada informasi oleh pihak *supermarket* kalau ada barang baru. [JMD-03]

2.2. Basis Data

Basis data, menurut Stephens dan Plew (2000), adalah mekanisme yang digunakan untuk menyimpan informasi atau data. Dengan basis data dapat menyimpan data secara terorganisasi. Setelah data disimpan, informasi harus mudah diambil. Kriteria dapat digunakan untuk mengambil informasi. Cara data disimpan dalam basis data menentukan seberapa mudah mencari informasi berdasarkan banyak kriteria. Data pun harus mudah ditambahkan ke dalam basis data, dimodifikasi, dan dihapus. [SIM-06]

2.2.1. Sistem Manajemen Basis Data (DBMS)

Ramakrishnan dan Gehrke (2003) menyatakan Sistem Manajemen Basis Data (DBMS) adalah perangkat lunak yang didesain untuk membantu memelihara dan memanfaatkan kumpulan data yang besar. Alternatif penggunaan DBMS adalah menyimpan data dalam *file* dan menulis kode aplikasi tertentu untuk mengaturnya. Keuntungan DBMS memungkinkan perusahaan maupun seseorang untuk: [SIM-06]

a. Mengurangi pengulangan data

DBMS mengurangi jumlah total *file* dengan menghapus data yang terduplikasi diberbagai *file*. Data terduplikasi selebihnya dapat ditempatkan dalam satu *file*.

b. Mencapai independensi data

Spesifikasi data disimpan dalam skema pada tiap program aplikasi. Perubahan data dapat dibuat pada struktur data tanpa mempengaruhi program yang mengakses data.

c. Mengintegrasikan data beberapa *file*

Organisasi logis, pandangan pengguna, dan program aplikasi tidak harus tercermin pada media penyimpanan fisik.

d. Mengambil data dan informasi dengan cepat

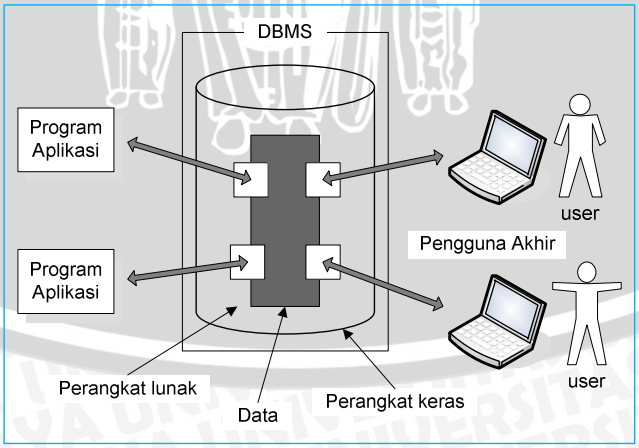
Hubungan-hubungan logis, bahasa manipulasi data, dan bahasa *query* memungkinkan pengguna mengambil data dalam hitungan detik atau menit.

e. Meningkatkan keamanan

DBMS mainframe maupun komputer mikro dapat menyertakan beberapa lapis keamanan seperti kata sandi (*password*), direktori pemakai, dan bahasa sandi (*encryption*) sehingga data yang dikelola akan lebih aman.

Komponen utama DBMS dapat dibagi menjadi 4 macam: [KAD-03]

1. perangkat keras,
2. data,
3. perangkat lunak, dan
4. pengguna.



Gambar 2. 1 Komponen-komponen utama sistem basis data
Sumber: [KAD-03]

2.2.2. Bahasa Basis Data

Sistem basis data menyediakan bahasa pendefinisian data (*Data Definition language-DDL*) untuk menentukan skema basis data dan bahasa manipulasi data (*Data Manipulation Language-DML*) untuk menyatakan *query* dan *update* basis data. [SIM-06]

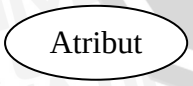
a. *Data Definition language* (DDL), skema basis data ditentukan sekumpulan definisi yang dinyatakan dengan bahasa tertentu. Sebagai contoh perintah pada SQL:
   - create : untuk membuat objek *database*.
   - alter : untuk memodifikasi objek *database*.
   - drop : untuk menghapus objek *database*.

b. *Data Manipulation Language* (DML), bahasa yang memungkinkan pengguna mengakses atau memanipulasi data seperti yang diatur oleh model data. Pada bahasa SQL terdapat perintah:
   - select : untuk mengambil data dari *database*.
   - update : untuk memodifikasi data pada *database*.
   - insert : untuk menambahkan data pada *database*.
   - delete : untuk menghapus data dari *database*.

2.2.3. *Entity Relationship Diagram* (ERD)

Entity relationship (ER) data model tersusun atas kumpulan kumpulan objek-objek dasar yang disebut entitas dan hubungan antarobjek. Skema *database* dapat ditunjukkan secara grafis dengan diagram ER yang dibentuk dari komponen-komponen berikut: [SIM-06]

persegi panjang mewakili kumpulan entitas



elips mewakili atribut

belah ketupat mewakili relasi

garis menghubungkan atribut dengan kumpulan entitas dan kumpulan entitas dengan relasi

2.2.4. Model Basis Data Relasional

Unit penyimpanan utama dalam basis data adalah tabel atau kelompok data yang saling berhubungan. Sebuah tabel terdiri atas baris dan kolom. Baris berhubungan dengan *record* dalam tabel dan kolom mengandung nilai semua baris yang berhubungan dengan *field* tertentu. Tabel dapat dihubungkan satu sama lain melalui nilai kolom yang disebut kunci (*key*). [SIM-06]

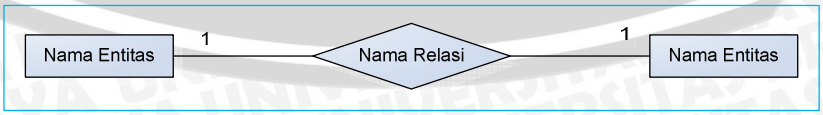
Ada 4 macam kunci (*key*) yang dapat diterapkan pada tabel dalam model basis data *relasional*, antara lain:

a. *Superkey*, merupakan satu atau lebih atribut yang dapat membedakan setiap baris data dalam sebuah tabel secara unik.
b. *Candidate-Key*, merupakan kumpulan atribut minimal yang dapat membedakan setiap baris data dalam sebuah tabel secara unik.
c. *Primary-Key*, merupakan *key* yang diambil dari beberapa *candidate-key*, dengan 3 kriteria sebagai berikut:
   - *Key* tersebut lebih natural untuk dijadikan acuan.
   - *Key* tersebut lebih sederhana.
   - *Key* tersebut cukup unik.
d. *Foreign-Key*, merupakan kunci tamu dalam suatu tabel dimana kunci ini juga merupakan *primary-key* dalam tabel lain yang berelasi.

2.2.5. Pemetaan Kardinalitas

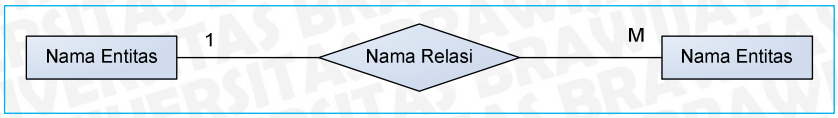
Pemetaan kardinalitas menyatakan jumlah entitas dimana entitas lain dapat dihubungkan ke entitas tersebut melalui sebuah himpunan relasi. Pemetaan kardinalitas yang dapat terjadi adalah: [SIM-06]

a. *One-to-One*, sebuah entitas pada A berhubungan dengan paling banyak satu entitas pada B dan sebuah entitas pada B berhubungan dengan paling banyak satu entitas pada A.



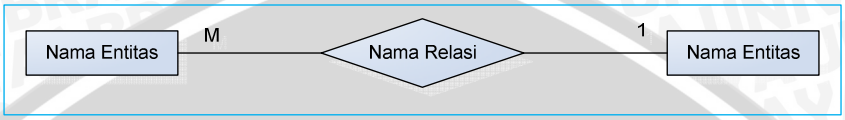
Gambar 2. 2 Simbol *one-to-one*
Sumber: [SIM-06]

b. *One-to-Many*, sebuah entitas pada A berhubungan dengan nol atau lebih entitas pada B. Sebuah entitas pada B dapat dihubungkan dengan paling banyak satu entitas pada A.



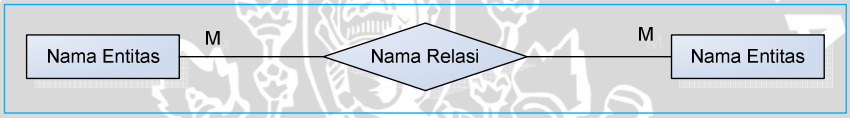
Gambar 2. 3 Simbol *one-to-many*
Sumber: [SIM-06]

c. *Many-to-One*, sebuah entitas pada A berhubungan dengan paling banyak satu entitas pada B. Sebuah entitas pada B dapat dihubungkan dengan nol atau lebih entitas pada A.



Gambar 2. 4 Simbol *many-to-one*
Sumber: [SIM-06]

d. *Many-to-Many*, sebuah entitas pada A berhubungan dengan dengan nol atau lebih entitas pada B dan sebuah entitas pada B dapat dihubungkan dengan nol atau lebih entitas pada A.



Gambar 2. 5 Simbol *many-to-many*
Sumber: [SIM-06]

2.3. *Unified Modelling Language* (UML)

UML adalah sistem notasi yang sudah dibakukan di dunia pengembangan sistem, hasil kerja bersama dari Grady Booch, James Rumbaugh dan Ivar Jacobson. UML yang terdiri dari serangkaian diagram memungkinkan bagi sistem analis untuk membuat cetak biru sistem yang komprehensif kepada klien, *programmer* dan tiap orang yang terlibat dalam proses pengembangan tersebut. Dengan UML akan bisa menceritakan apa yang seharusnya dilakukan oleh sebuah sistem bukan bagaimana yang seharusnya dilakukan oleh sebuah sistem. [MUN-05]

Diproyek pengembangan sistem apapun, fokus utama dalam analisis dan perancangan adalah model. Hal ini berlaku umum tidak hanya untuk perangkat lunak. Dengan model bisa merepresentasikan sesuatu karena:

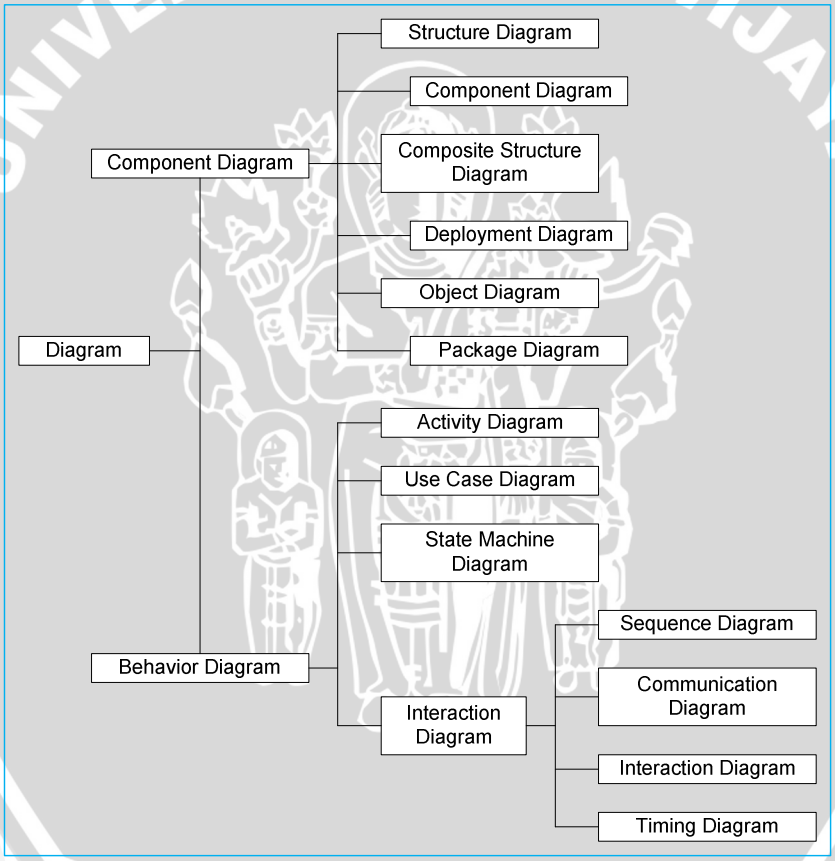
- Model mudah dan cepat untuk dibuat.
- Model bisa digunakan sebagai simulasi untuk mempelajari lebih detil tentang sesuatu.
- Model bisa dikembangkan sejalan dengan pemahaman kita tentang sesuatu.
- Dengan model bisa memberikan penjelasan lebih rinci tentang sesuatu.

- Model bisa mewakili sesuatu yang nyata maupun yang tidak nyata.

Disisi lain, ada alat bantu lain yang sangat sering dipakai oleh sistem analis dan perancang. Alat bantu tersebut adalah diagram. Diagram ini digunakan untuk:

- Mengkomunikasikan ide
- Melahirkan ide-ide baru dan peluang-peluang baru
- Menguji ide dan membuat prediksi
- Memahami struktur dan relasi-relasinya

Diagram menggambarkan atau mendokumentasikan beberapa aspek dari sebuah sistem. Sedangkan sebuah model menggambarkan pandangan yang lengkap tentang suatu sistem pada suatu tahapan tertentu dan dari perspektif tertentu. Sebuah model mungkin mengandung satu atau lebih diagram. [MUN-05]

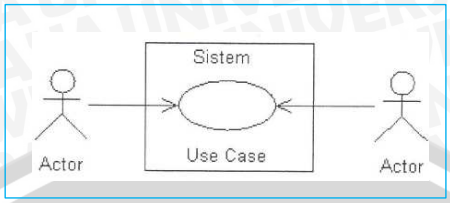


Gambar 2. 6 Klasifikasi diagram UML versi 2.0
Sumber: [MUN-05]

2.3.1. *Use Case Diagram*

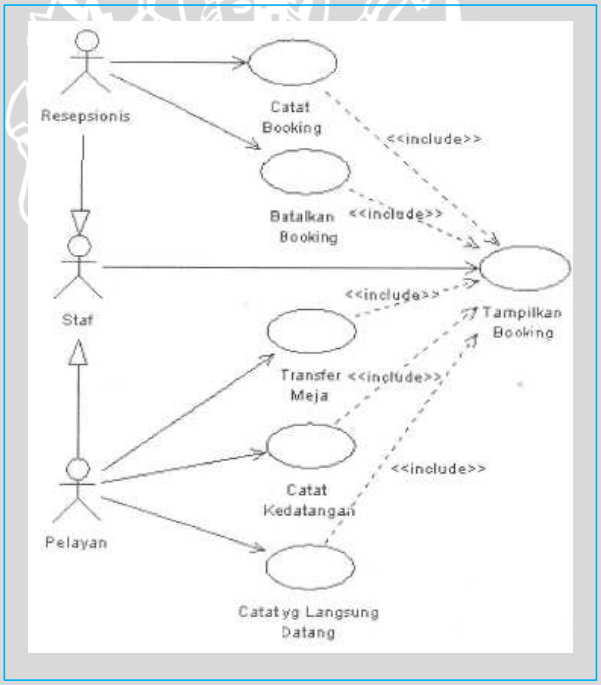
Use case adalah deskripsi fungsi dari sebuah sistem dari perspektif pengguna. *Use case* bekerja dengan cara mendeskripsikan tipikal interaksi antara *user* (pengguna) sebuah sistem dengan sistemnya sendiri melalui sebuah cerita bagaimana sebuah sistem dipakai. Urutan langkah-langkah yang menerangkan antara pengguna

dan sistem disebut *scenario*. Setiap *scenario* mendeskripsikan urutan kejadian. Setiap urutan diinisialisasi oleh orang, sistem yang lain, perangkat keras atau urutan waktu. Dalam *use case*, pengguna biasa disebut dengan *actor*. *Actor* adalah sebuah peran yang bisa dimainkan oleh pengguna dalam interaksinya dengan sistem. [MUN-05]



Gambar 2. 7 *Use case model*
Sumber: [MUN-05]

Di dalam *use case*, <<extend>> digunakan untuk menunjukkan bahwa satu *use case* yang lain jika kondisi atau syarat tertentu yang dipenuhi. Sedangkan <<include>> digunakan untuk menggambarkan bahwa suatu *use case* seluruhnya merupakan fungsionalitas dari *use case* lainnya. Biasanya <<include>> digunakan untuk menghindari peng-*copy*-an suatu *use case* karena sering dipakai.

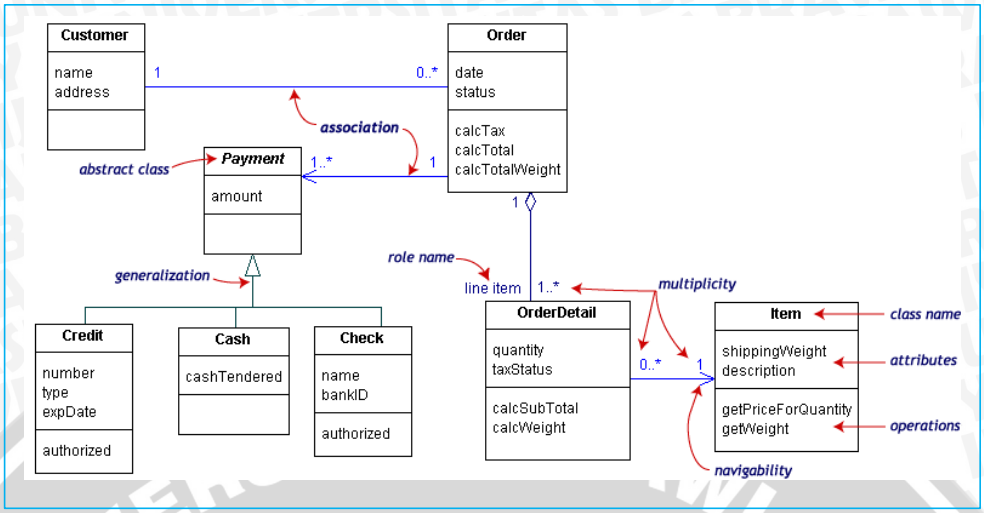


Gambar 2. 8 Contoh diagram *use case*
Sumber: [MUN-05]

2.3.2. *Class Diagram*

Class menggambarkan keadaan (atribut/properti) suatu sistem, sekaligus menawarkan layanan untuk memanipulasi keadaan tersebut (metoda/fungsi). *Class diagram* menggambarkan struktur dan deskripsi *class*, *package* dan objek beserta

hubungan satu sama lain seperti *containment*, pewarisan, asosiasi, dan lain-lain. [DHA-03]



Gambar 2. 9 Contoh diagram *class*
Sumber: [DHA-03]

Class memiliki tiga area pokok:

a. Nama (dan stereotype)
b. Atribut
c. Metoda

Atribut dan metoda dapat memiliki salah satu sifat berikut:

a. *Private*, tidak dapat dipanggil dari luar *class* yang bersangkutan
b. *Protected*, hanya dapat dipanggil oleh *class* yang bersangkutan dan anak-anak yang mewarisinya
c. *Public*, dapat dipanggil oleh siapa saja

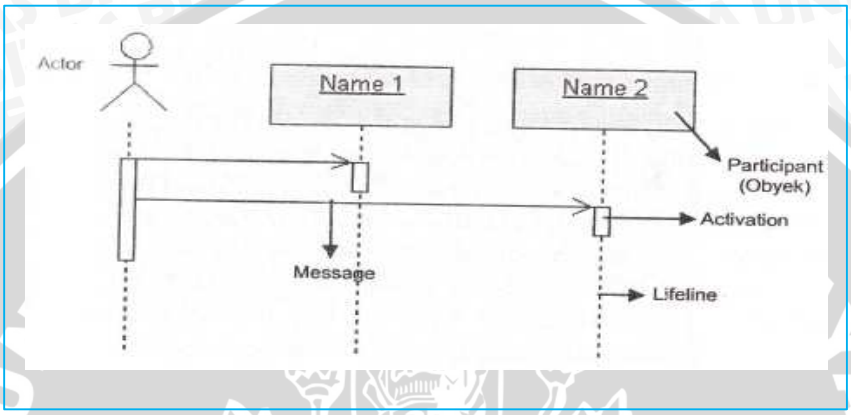
2.3.3. *Sequence Diagram*

Sequence diagram digunakan untuk menggambarkan perilaku pada sebuah *scenario*. Diagram ini menunjukkan sejumlah contoh obyek dan *message* (pesan) yang diletakkan diantara obyek-obyek ini di dalam *use case*. Komponen utama *sequence diagram* terdiri atas obyek yang dituliskan dengan kotak segiempat bernama. *Message* diwakili oleh garis dengan tanda panah dan waktu yang ditunjukkan dengan *progress vertical*.

Sequence diagram menambahkan dimensi waktu pada interaksi diantara obyek. Pada diagram ini *participant* diletakkan di atas dan waktu ditunjukkan dari atas ke bawah. *Life line participant* diurutkan dari setiap *participant*. Kotak kecil

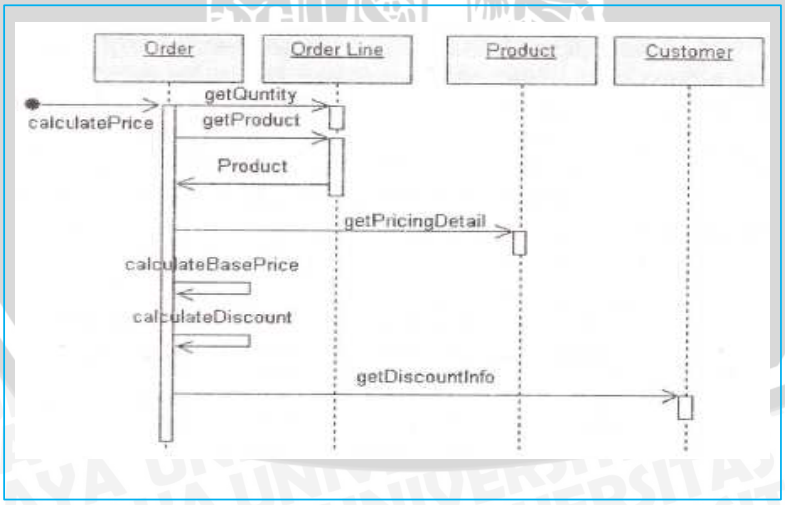
pada *life line* menyatakan *activation*, yaitu menjalankan salah satu *operation* dari *participant*. *State* bisa ditambahkan dengan menempatkannya sepanjang *life line*.

Message (sederhana, *synchronous* atau *asynchronous*) adalah tanda panah yang menghubungkan suatu *life line* ke *life line* yang lain. Lokasi *life line* dalam dimensi vertikal mewakili urutan waktu dalam *sequence diagram*. *Message* yang pertama terjadi adalah yang paling dekat dengan bagian atas diagram dan yang terjadi belakangan adalah yang dekat dengan bagian bawah.



Gambar 2. 10 Simbol-simbol yang ada pada *sequence diagram*
Sumber: [MUN-05]

Pada beberapa sistem, operasi bisa dilakukan kepada dirinya sendiri. Hal ini disebut dengan rekursif. Untuk melukiskannya digunakan anak panah dari *activation* kembali ke dirinya sendiri, dan sebuah kotak kecil diletakkan pada bagian atas dari *activation*. [MUN-05]



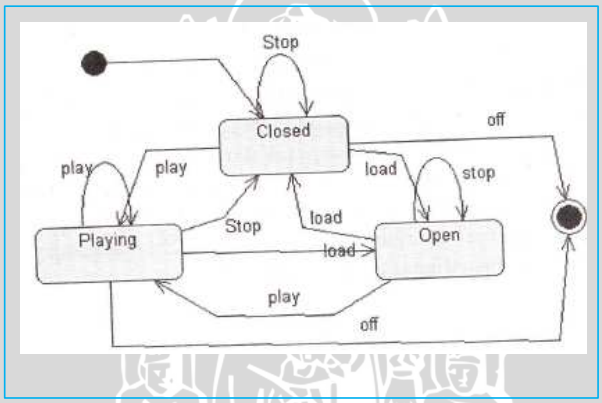
Gambar 2. 11 Contoh *sequence diagram*
Sumber: [MUN-05]

2.3.4. *State Diagram*

Statechart diagram menelusuri individu-individu obyek melalui keseluruhan daur hidupnya, menspesifikasikan semua urutan yang mungkin dari pesan-pesan yang akan diterima obyek tersebut, bersama-sama dengan tanggapan atas pesan-pesan tersebut. *State diagram* menyediakan variasi simbol dan sejumlah ide untuk pemodelan.

Simbol UML untuk *state transition diagram* adalah segiempat yang tiap pojoknya dibuat *rounded*. Titik awalnya menggunakan lingkaran solid yang diarsir dan diakhiri dengan mata. Berikut adalah simbol UML untuk *statechart*. [MUN-05]

Gambar 2. 12 Simbol *statechart diagram*
Sumber: [MUN-05]



Gambar 2. 13 Contoh *statechart diagram*
Sumber: [MUN-05]

2.4. MySQL

MySQL adalah sistem manajemen *database relasional*. Suatu *database relasional* menyimpan data dalam tabel-tabel terpisah. Hal ini memungkinkan kecepatan dan fleksibilitas. Tabel-tabel yang dihubungkan dengan relasi yang ditentukan membuatnya bisa mengkombinasikan data dari beberapa tabel pada suatu permintaan. [UTD-02]

Keistimewaan yang dimiliki MySQL: [PRA-03]

a. *Portability*, MySQL dapat berjalan stabil pada berbagai sistem operasi.
b. *Open Source*, MySQL didistribusikan secara gratis dibawah lisensi GPL (*General Public License*).

c. *Multiuser*, MySQL dapat digunakan oleh beberapa *user* dalam waktu yang bersamaan tanpa mengalami masalah.
d. *Performance Tuning*, MySQL memiliki kecepatan dalam *query* sederhana.
e. *Column Types*, MySQL memiliki tipe kolom yang sangat kompleks, seperti signed/unsigned integer, float, double, char, varchar, text, blob, date, time, datetime, timestamp, year, set, dan enum.
f. *Command* dan *Functions*, MySQL memiliki operator dan fungsi secara penuh yang mendukung perintah SELECT dan WHERE dalam *query*.
g. *Security*, MySQL memiliki beberapa lapisan sekuritas seperti level *subnetmask*, nama *host*, dan izin akses *user* dengan sistem *password*.
h. *Scalability* dan *Limits*, MySQL mampu menangani *database* dalam skala besar dengan jumlah *record* lebih dari 50 juta dan 60 ribu tabel serta 5 miliar baris.
i. *Connectivity*, MySQL dapat melakukan koneksi dengan *client* menggunakan *protocol* TCP/IP, Unix soket, atau Named Pipes (NT).
j. *Localisation*, MySQL dapat mendeteksi pesan kesalahan pada *client* dengan menggunakan lebih dari dua puluh bahasa.
k. *Interface*, MySQL memiliki *interface* terhadap berbagai aplikasi dan bahasa pemrograman dengan menggunakan fungsi API (*Aplication Programming Interface*).
l. *Clients* dan *Tools*, MySQL dilengkapi dengan berbagai *tool* yang dapat digunakan untuk administrasi *database*.
m. Struktur Tabel, MySQL memiliki struktur tabel yang lebih fleksibel dalam menangani ALTER TABLE.

2.4.1 Bahasa MySQL

a. CREATE DATABASE digunakan untuk membuat sebuah *database*.
   Sintaks: `CREATE DATABASE database_name`
b. DROP DATABASE digunakan untuk menghapus sebuah *database*.
   Sintaks: `DROP DATABASE database_name`
c. CREATE TABLE digunakan untuk membuat tabel baru.
   Sintaks: `CREATE TABLE table_name (create_definition, …)`
   Dimana `create_definition` berbentuk seperti:

```
nama_field tipe [NOT NULL | NULL] [DEFAULT nilai default] [AUTO_INCREMENT]
[PRIMARY_KEY]
```

atau `PRIMARY KEY (nama_field)`

atau `KEY [nama_index] (nama_field)`

atau `INDEX [nama_index] (nama_field)`

atau `UNIQUE [INDEX] [nama_index] (nama_field)`

d. ALTER TABLE digunakan untuk memodifikasi tabel yang pernah dibuat.

Sintaks:
```
ALTER [IGNORE] TABLE table_name
alter_spesification [,alter_spesification …]
```

Dimana `alter_spesification` berbentuk seperti:

`ADD [COLUMN] create_definition`

atau `CHANGE [COLUMN] old_column_name create_definition`

atau `ALTER [COLUMN] column_name {SET default | DROP DEFAULT}`

atau `DROP [COLUMN] column_name`

atau `DROP PRIMARY KEY`

atau `DROP INDEX key_name`

e. DROP TABLE digunakan untuk menghapus sebuah tabel.

Sintaks:
```
DROP TABLE table_name [table_name …]
```

f. INSERT digunakan untuk menyisipkan suatu data ke dalam tabel. Data yang disisipkan dapat berupa data yang diambil dari tabel lain, ataupun data yang berupa nilai-nilai tertentu yang disebutkan secara eksplisit.

Sintaks:
```
INSERT INTO table [(column_name, …)] VALUES (expression, …)
```
atau `INSERT INTO table [(column_name, …)] SELECT`

g. SELECT digunakan untuk mengambil data dari suatu tabel.

Sintaks:
```
SELECT {nama_field} FROM nama_tabel [INTO tabel_tujuan] [WHERE
kondisi]
```

h. DELETE digunakan untuk menghapus sebuah *record* dari tabel.

Sintaks:
```
DELETE FROM nama_tabel WHERE kondisi
```

i. UPDATE digunakan untuk memperbarui nilai suatu data.

Sintaks:
```
UPDATE nama_tabel SET criteria where kondisi
```

j. REPLACE hamper sama dengan INSERT, perbedaannya adalah jika *record* lama pada tabel memiliki nilai yang sama dengan *record* baru pada sebuah indeks unik, maka *record* lama akan dihapus dan diganti dengan *record* baru.

k. USE digunakan untuk memilih *database* yang akan digunakan.

Sintaks:
```
USE nama_tabel
```

l. SHOW digunakan untuk menampilkan informasi tentang *database* yang sedang digunakan.

Sintaks:

```
SHOW DATABASE [LIKE kondisi]
SHOW [OPEN] TABLES [FROM nama_database] [LIKE kondisi]
SHOW [FULL] COLUMNS FROM nama_tabel [FROM nama_database] [LIKE
kondisi]
```

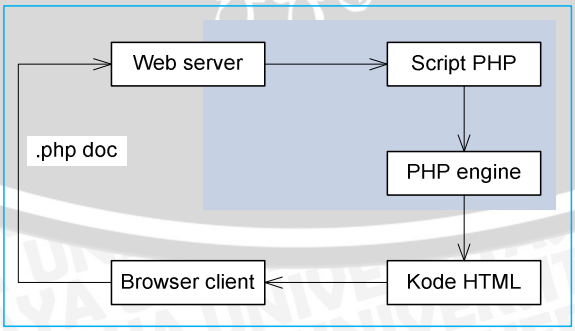
2.5. PHP

PHP (*Hypertext Prepocessor*) adalah skrip bersifat *server-side* yang ditambahkan ke dalam HTML. Skrip ini akan membuat suatu aplikasi dapat diintegrasikan ke dalam HTML sehingga suatu halaman *web* tidak lagi statis, namun bersifat dinamis. Sifat *server-side* berarti pengerjaan skrip akan dilakukan di *server*, baru kemudian hasilnya dikirimkan ke *browser*.

Keunggulan PHP adalah dapat mengambil nilai *form*, menghasilkan halaman *web* yang dinamis, serta menerima dan mengirim *cookie*. PHP juga dapat berkomunikasi dengan layanan-layanan yang menggunakan protokol IMAP, SNMP, NNTP, POP3, dan HTTP. Kemampuannya untuk melakukan koneksi dengan berbagai macam *database*, seperti: Interbase, PostgreSQL, dBase, FrontBase, mSQL, IBM DB2, MySQL, Unix dbm, semua *database* yang mempunyai *provider* ODBC, Ingres, dan Oracle. [KUR-02]

Berikut contoh skrip PHP sederhana:

```
<html>
    <head>
        <title> contoh skrip PHP </title>
    </head>
    <body>
    <?
        echo “PHP (Personal Home Page)”;
    ?>
    </body>
</html>
```



Gambar 2. 14 Skema PHP
Sumber: [KAD-02]

2.6. XHTML MP

Extensible Hypertext Markup Language Mobile Profile (XHTML MP) merupakan sebuah bahasa *markup* yang didefinisikan dalam *Wireless Aplication Protocol* 2.0, yaitu sebuah *protocol* komunikasi untuk aplikasi-aplikasi nirkabel yang dibuat oleh WAP forum. XHTML sendiri sebenarnya merupakan gabungan antara *Hypertext Markup Language* (HTML) yang telah umum digunakan sebagai bahasa pemrograman untuk membuat situs-situs *internet* dewasa ini dan *Extensible Markup Language* (XML). Penambahan istilah *Mobile Profile* berarti XHTML MP merupakan bahasa pemrograman yang dikhususkan untuk membangun aplikasi-aplikasi yang dapat dibaca melalui perangkat-perangkat *mobile*, seperti telepon seluler (ponsel), PDA ataupun *smartphone*. Karena XHTML MP merupakan bagian dari XHTML maka sintaks dan aturan-aturan dalam menulis aplikasi dengan menggunakan XHTML MP mengikuti sintaks dan aturan pada XHTML.

Sebelum adanya XHTML MP, semua aplikasi *mobile* dibangun dengan menggunakan bahasa pemrograman *Wireless Markup Language* (WML) dan WML*Script*. Aplikasi yang dibangun dengan menggunakan WML dan WML*Script* hanya dapat dibaca melalui *browser* yang ada didalam perangkat ponsel yang memiliki kemampuan menjalankan protokol WAP 1.2.1 dan versi-versi sebelumnya.

Sedangkan aplikasi *mobile* yang dibangun dengan menggunakan XHTML MP ini selain dapat dibaca melalui *browser* yang ada di dalam ponsel, juga dapat dibaca melalui *internet browser*. Karena tujuan utama XHTML MP adalah menggabungkan teknologi *browser* yang ada pada *mobile* dan *World Wide Web* (WWW). [JUS-07]

2.6.1. Struktur Dasar Dokumen XHTML MP

Struktur dokumen XHTML MP terdiri atas beberapa bagian penting seperti terlihat dibawah ini:

```
<?xml version="1.0"?>
<!DOCTYPE html PUBLIC "-//WAPFORUM//DTD XHTML Mobile 1.0//EN"
"http://www.wapforum.org/DTD/xhtml-mobile10.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>JUDUL</title>
    <meta name="author" content="Nama"/>
  </head>
  <body>
    <p>isi . . . </p>
  </body>
</html>
```

Dari *script* di atas secara umum dokumen XHTML MP terdiri atas tiga bagian utama yaitu:

- DOCTYPE,
- Head, dan
- Body

Baris pertama dari sebuah dokumen XHTML MP adalah deklarasi XML. Deklarasi ini dibutuhkan karena XHTML MP merupakan turunan dari XML. Seperti terlihat pada *script* diatas deklarasi *xml* diikuti dengan versi dari dokumen yaitu versi 1.0. dan juga jenis *encoding* karakter yang digunakan bisa dispesifikasikan dalam baris pertama seperti berikut:

```
<?xml version=”1.0” encoding=”UTF-8”?>
```

Apabila jenis *encoding* yang digunakan adalah UTF-8 atau UTF-16 maka *encoding* ini tidak harus dideklarasikan, karena secara *default* XHTML MP akan menggunakan jenis *encoding* ini.

Deklarasi xml bukan merupakan keharusan dalam XHTML MP. Akan tetapi apabila tidak dideklarasikan, maka kemungkinan akan muncul permasalahan untuk beberapa WAP *browser*, misalnya WAP *browser* Sony Ericsson menggunakan deklarasi ini untuk membedakan antara mana yang merupakan dokumen XHTML MP dengan CHTML bahasa *markup* buatan NTT Docomo Jepang atau *i-mode*.

Deklarasi DOCTYPE pada baris kedua mutlak harus ada dalam dokumen XHTML MP. Diletakkan diantara deklarasi *xml* dan elemen <html>. Deklarasi ini menspesifikasikan nama dari DTD (*Document Type Definision*) dan url dimana DTD tersebut berada. DTD menyimpan semua informasi tentang sintaks dari bahasa *markup* dan mendefinisikan elemen-elemen, atribut, dan aturan-aturan yang harus digunakan. Baris selanjutnya yaitu elemen <html>, <head>, <title>, dan <body> yang merupakan batang tubuh dari dokumen XHTML MP. Bagian ini sama dengan penulisan HTML kecuali adanya tambahan deklarasi *xmlns* pada XHTML.

a. Elemen <html>

   Elemen <html> merupakan elemen utama, elemen-elemen yang lain seperti <head>, <title>, dan <body> harus berada didalam tag <html> dan </html>. Secara umum semua dokumen HTML menggunakan elemen <html>, secara khusus XHTML MP menambahkan sebuah atribut *xmlns* dalam elemen <html>. Atribut ini berfungsi untuk mendefinisikan *namespace* bagi XHTML MP.

b. Elemen <head>

Pada elemen ini tag <head> digunakan sebagai pembuka dan ditutup dengan tag </head>. Elemen <head> berfungsi untuk menyimpan semua informasi tentang dokumen XHTML MP, misalnya judul dokumen dan *link* menuju ke CSS. Judul dokumen dituliskan dengan meletakkan diantara tag <title> dan </title>. Untuk menghubungkan dokumen XHTML MP dengan CSS eksternal, dapat digunakan elemen <link> seperti berikut ini:

```
<head>
   <title> judul </title>
   <link href="home.css">
         rel="stylesheet" type="teks/css"/>
</head>
```

c. Elemen <body>

Elemen <body> berisi semua elemen-elemen yang akan ditampilkan dalam halaman situs XHTML MP. Dalam sebuah halaman hanya diperbolehkan menggunakan sebuah elemen <body>. Selain itu teks yang ada didalam elemen <body> harus berada didalam elemen-elemen lain seperti elemen paragraf … , *heading* <h1> … </h1>, dan elemen *list* … .

d. Elemen

Elemen digunakan untuk menampung sebuah paragraf dalam halaman XHTML MP. Jadi setiap WAP *browser* membaca elemen , maka akan memulai *text* tersebut pada baris yang baru. Tetapi, XHTML MP tidak memiliki elemen untuk membuat paragraf rata kanan, kiri, atau tengah.

e. Metadata

Metadata dalam XHTML MP diletakkan dalam sebuah tag <meta />, tag ini harus dituliskan di dalam tag <head>...</head>. Keuntungannya, metadata tidak akan ditampilkan, karena itu kehadiran metadata tidak akan mempengaruhi tampilan dari dokumen XHTML MP dan WAP *browser* secara otomatis akan menghiraukan metadata apabila *browser* tersebut tidak memahami isi dari metadata sehingga dokumen yang memuat metadata dapat di*upload* dari berbagai macam perangkat *mobile* tanpa mengganggu tampilan. Dalam XHTML MP metadata berfungsi untuk:

- Menjelaskan pemilik dokumen

```
<html xmlns="http://www.w3.org/1999/xhtml">
   <head>
      <title> skripsi </title>
      <meta name="author" content="jati"/>
   </head>
```

- Menjelaskan deskripsi dokumen

```
<html xmlns=”http://www.w3.org/1999/xhtml”>
    <head>
        <title> skripsi </title>
        <meta name=”description” content=”coba-coba”/>
    </head>
```

- Menjelaskan kata kunci dokumen

```
<html xmlns=”http://www.w3.org/1999/xhtml”>
    <head>
        <title> skripsi </title>
        <meta name=”description” content=”coba-coba”/>
    </head>
```

- Menentukan kapan sebuah dokumen akan di*refresh*

```
<?xml version=”1.0”?><!-- File: skripsi.xhtml -->
<!DOCTYPE html PUBLIC “-//WAPFORUM//DTD XHTML Mobile 1.0//EN”
“http://www.wapforum.org/DTD/xhtml-mobile10.dtd”>
<html xmlns=”http://www.w3.org/1999/xhtml”>
    <head>
        <title> skripsi </title>
        <meta http-equiv=”refresh” content=”8”/>
    </head>
    <body>
        <p> akan direfresh setiap 8 detik </p>
    </body>
</html>
```

2.6.2. Keuntungan menggunakan XHTML MP sebagai bahasa pemrograman untuk semua aplikasi *mobile internet*

a. Sebuah situs XHTML dapat dibaca melalui *mobile browser* maupun *internet browser*, hal ini berarti tidak diperlukan teknologi berbeda untuk membangun aplikasi yang dapat diakses melalui WWW atau WAP 2.0.

b. Sebuah situs dapat diakses melalui dua media sekaligus maka hanya dibutuhkan sebuah *tool* untuk membangun situs yang dapat diakses melalui WWW dan WAP 2.0.

c. Pada dasarnya XHTML adalah HTML dengan aturan-aturan ketat XML sehingga tidak sulit untuk membangun situs *internet* dengan XHTML.

d. Kehadiran WAP CSS (WCSS) pada XHTML MP sangat memudahkan pengaturan tata letak dari sebuah situs. Karena WCSS memungkinkan adanya pemisahan antara isi dari sebuah situs dengan pengaturan tata letaknya.

2.6.3. Perbedaan XHTML MP dan HTML

a. Elemen-elemen dalam XHTML MP harus tersarang dengan benar .

`<b><i> hello world! </i></b>`, bukan

`<b><i> hello world! </b></i>`

b. Elemen-elemen dalam XHTML MP harus tertutup.

`<p> hello world! </p>`

Elemen-elemen lain dalam XHTML MP yang membutuhkan penutupan secara benar adalah elemen-elemen kosong seperti berikut ini:

Elemen untuk pindah baris:

c. Elemen-elemen dalam XHTML MP harus tertulis dengan huruf kecil.

```
<p> hello world! </p>, bukan
<P> hello world! </P>
```

d. Nilai dari setiap atribut pada XHTML MP harus ditulis dalam tanda petik.

```
<p id="par1"> ini paragraph 1 </p>, atau
<p id='par1'> ini paragraph 1 </p>, bukan
<p id=par1> ini paragraph 1 </p>
```

e. XHTML MP tidak mengijinkan adanya minimasi atribut.

Contoh:

```
<input type="radio" value="1" checked="checked"> XHTML MP 1 <br />
<input type="radio" value="2"> XHTML MP 2 <br />, bukan
<input type="radio" value="1" checked> XHTML MP 1 <br />
<input type="radio" value="2"> XHTML MP 2 <br />
```

XHTML MP adalah bahasa pemrograman untuk membuat desain halaman *web* dan merupakan penyempurnaan bahasa HTML. Tujuan XHTML MP adalah agar tampilan bisa lebih konsisten disemua *browser*. Dokumen XHTML MP dapat divalidasi dengan mudah. Pada HTML tidak semua *browser* mendukung sintaks HTML yang sama. HTML merupakan *tag* untuk memformat tampilan atau menitikberatkan pada unsur presentasi.

2.6.4. Perbedaan XHTML MP dan WML

a. XHTML MP tidak memiliki *deck* dan *card*.

Deck dan *card* adalah fitur utama pada WML. Sebuah *file* yang di*download* dapat mengandung beberapa *card*. Dan beberapa *card* tersebut membentuk sebuah *deck*.

b. XHTML MP tidak mendukung *client-side scripting*.

Client-side scripting dengan menggunakan WML*scripti* dapat dilakukan. Akan tetapi hal semacam ini tidak disediakan oleh XHTML MP. Sebagai gantinya XHTML mendukung adanya *server-side scripting*.

c. XHTML MP tidak mendukung *variabel*.

Pada WML diijinkan untuk membuat *variabel* dan memasukkan sebuah nilai ke dalamnya. Hal ini sangat berguna apabila sebuah nilai diinginkan untuk

digunakan berulang-ulang dari satu *card* ke *card* berikutnya. Sebaliknya pada XHTML MP seluruh proses disimpan dan dilakukan oleh *server*.

d. XHTML MP tidak mendukung *event*.

WML mendukung adanya empat buah *event*, yaitu: *ontimer, onenterbackward, onenterforward, onpick.*

e. XHTML MP tidak mendukung *timer*.

Event timer tidak didukung oleh XHTML MP, sehingga untuk mendapat fungsi yang sama digunakan HTTP *refresh*.

f. XHTML MP tidak mendukung pemrograman *softkey*.

Pemrograman *softkey* pada WML adalah penggunaan sebuah tombol pada *keypad* ponsel untuk mengarahkan situs melompat ketujuan tertentu. Misalnya tombol panah atas dapat diprogram untuk mengembalikan ke posisi *home*. XHTML tidak memiliki fitur pemrograman *softkey*. Sebagai gantinya XHTML menggunakan sebuah atribut *accesskey* yang memiliki fungsi mirip seperti pada pemrograman *softkey*.

WML adalah bahasa pemrograman untuk membuat desain WAP untuk *mobile content*. Tujuan WML adalah untuk menampilkan halaman *web* pada perangkat bergerak yang memiliki sumber daya atau kemampuan terbatas melalui WAP (*Wireless Application Protocol*) *browser*. WML mampu menyembunyikan *script* halaman WML dan layanan untuk validasi waktu tertentu untuk memperkecil *routing* pada *web server*. Elemen WML dapat dengan mudah diimplementasikan dengan menggunakan *keyboard* kecil (*keypad ponsel*). WML juga tidak memiliki kemampuan WCSS.

2.6.5. Tipe MIME (*Multipurpose Internet Mail Extensions*)

Pada saat *browser* disisi pengguna melakukan permintaan dokumen *web* kepada *web server*, maka *web server* akan memberitahukan jenis dokumen *web* apa yang dimiliki. Jenis dokumen *web* ini didefinisikan dari tipe MIME.

Tabel 2. 1 Tipe MIME

Tipe dokumen	Tipe MIME	Ekstensi *file*
HTML	text/html	.html
XHTML Basic	application/xhtml+xml	.xhtml
XHTML MP	application/vnd.wap.xhtml+xml	.xhtml

Sumber: [JUS-07]

Sangat disarankan agar ketiga tipe MIME tersebut ditambahkan pada *web server* agar dokumen XHTML MP dapat diakses melalui *browser*.

2.7. WCSS

WCSS (WAP *Cascading Style Sheet*) adalah versi *mobile* dari CSS. Ini adalah sub bagian dari CSS2 (bahasa CSS dari *World Wide Web*) ditambah beberapa ekstensi spesifik WAP. Fitur CSS2 dan kelengkapannya yang tidak banyak berguna untuk aplikasi *internet mobile* tidak termasuk dalam WAP CSS. WAP CSS adalah kerabat dari XHTML *Mobile Profile* (XHTML MP). Keduanya didefinisikan dalam spesifikasi WAP 2.0 yang dibuat oleh forum pembuat WAP (saat ini adalah *Open Mobile* Alliance atau OMA). Ada beberapa WAP 2.0 yang memungkinkan digunakan untuk telepon selular saat ini.

Fitur ini sangat berguna dalam dunia tanpa kabel yang mana bermacam-macam peralatan *mobile* seperti telepon selular yang mempunyai karakteristik yang bervariasi seperti ukuran layar. Dengan WCSS dapat mengendalikan tampilan dari halaman yang ada pada peralatan *mobile* yang berbeda dengan menggunakan pemisah dari lembar pola CSS dan tidak harus merubah isi *file*.

Tipe MIME (*Multipurpose Internet Mail Extentions*) dan ekstensi *file* dari WCSS adalah teks/CSS dan ekstensi *file*nya adalah ".css" . Tipe ini sama dengan CSS *web*. [ANO-08]

Berikut ini adalah beberapa keuntungan penggunaan lembaran-lembaran model WAP CSS pada situs-situs *internet mobile*:

a. Dengan adanya WAP 2.0 (XHTML MP / WAP CSS), pemrogaman *web* dan WAP digabungkan. Para pengembang *web* dapat terus memanfaatkan alat-alat pengaturan *web* yang telah dikenal dan PC *web browser* untuk membuat situs-situs *internet mobile*. Ini merupakan salah satu kelebihan utama dari XHTML MP/WCSS melebihi WML.
b. Penggunaan WAP CSS memiliki keuntungan yaitu isi dan tampilannya dapat dipisahkan. Ini berarti anda dapat:
   - Mencocokkan layout dan model dari isi dan karakteristik yang sama dari peralatan *wireless* dengan mudah. Misalnya, jika menginginkan *layout* situs *internet mobile* yang berbeda pada *mobile phone* yakni pada perbedaan ukuran layarnya, maka dapat mendesain beberapa versi lembaran model WAP CSS dan masing-masing versi diperuntukkan untuk satu ukuran *layer* tertentu. *File* yang ada tidak perlu dimodifikasi.
   - Mencocokkan *layout* dan model isi yang sama untuk *user* yang berbeda dengan mudah. Misalnya, jika menginginkan *layout* situs *internet mobile*

berbeda pada sebuah PC, PDA dan *mobile phone*, maka dapat mendesain beberapa versi lembaran model WAP CSS dan masing-masing versi digunakan untuk satu tipe klien tertentu. *File-file* yang ada tidak perlu dimodifikasi.

- Meminimalisir usaha untuk memelihara sebuah situs WAP. Ketika model-model *mobile phone* yang baru mulai diperkenalkan di pasaran, dapat dibuat lembaran model WCSS baru untuk menoptimalkan *layout* dari situs WAP pada *mobile phone* yang baru ini. *File-file* yang telah ada tidak perlu dimodifikasi.
- Menggunakan lembaran model WAP CSS tunggal untuk melipat gandakan halaman WAP. Kemudian jika ingin merubah tampilan keseluruhan situs WAP, hanya perlu untuk memodifikasi lembaran model WAP CSS.
- Menggunakan kembali kode model dalam melipat gandakan proyeksi/rancangan.
- Mengembangkan pembagian kerja. Misalnya, salah satu anggota tim dapat fokus terhadap tampilan dan sentuhan situs WAP dan yang lainnya fokus terhadap isi. Dengan pemisahan tugas tersebut, akan mempermudah dalam menjaga konsistensi tampilan dan sentuhan dari keseluruhan situs.

c. Memiliki kontrol yang lebih besar pada tampilan halaman WAP dengan WCSS dari pada dengan WML. Misalnya, dapat menentukan warna, *font*, *background*, *border*, *margin*, dan *padding* berbagai macam elemen dengan WCSS.

d. Jika menggunakan model *cascading* tunggal pada semua situs *internet mobile*, sebuah perlengkapan *mobile* akan men*download* lembaran model *cascading* hanya sekali pada kunjungan pertama situs *internet mobile* tersebut. Lembaran model *cascading* kemudian akan disimpan pada tempat yang tersembunyi dan dapat diakses kembali tanpa harus terhubung dengan *server*.

e. Ukuran *file* dokumen XHTML MP dapat diperkecil jika *layout* dan informasi penyusunan dipindah pada sebuah lembaran model WAP CSS eksternal. Ukuran *file* yang lebih kecil memiliki keuntungan berupa waktu *download* yang lebih singkat. (walaupun demikian, cara ini tidak dapat digunakan untuk kunjungan halaman WAP pertama oleh seorang *user* karena lembaran model

eksternal harus di*download* dengan permintaan HTTP secara terpisah, yang berarti waktu *download* total tidak mungkin lebih singkat seperti dua *round trip* yang berlangsung).

Sintaks dari WAP CSS sebagai berikut:

```
selector {property: property_value}
```

contoh:

```
p {text-align: right}
```

Dengan banyak properti dalam satu pernyataan, antara properti dipisahkan dengan tanda *semicolon* (;) maka sintaks:

```
selector {property1: property_value1; property2: property_value2; ... propertyN: property_valueN}
```

contoh:

```
p {text-align: right; color: blue}
```

Jika dalam pernyataan terdapat lebih dari satu *selector*, antara *selector* yang satu dengan yang lain dipisahkan dengan tanda koma, sintaksnya adalah:

```
selector1, selector2, ... selectorN {property1: property_value1}
```

contoh:

```
p, h1 {text-align: right; color: blue}
```

2.8. Pengujian Perangkat Lunak

Pengujian perangkat lunak adalah elemen kritis dari jaminan kualitas perangkat lunak dan merepresentasikan spesifikasi, desain dan pengkodean.

2.8.1. Teknik Pengujian

Pengujian perangkat lunak memerlukan perancangan kasus uji (*test case*) agar dapat menemukan kesalahan dalam waktu singkat dan usaha minimum. Metode perancangan kasus uji menyediakan mekanisme penting yang dapat membantu menjamin kompleksitas pengujian dan keandalan yang tinggi untuk mengatasi kesalahan. Perangkat lunak dapat diuji melalui dua cara yaitu *white box testing* dan *black box testing*.

2.8.1.1. White Box Testing

Uji coba *white box* adalah metode perancangan *test case* yang menggunakan struktur kontrol dari perancangan prosedural untuk mendapatkan *test case*. Dengan menggunakan metode *white box*, analis sistem akan dapat memperoleh *test case* yang: [AN1-09]

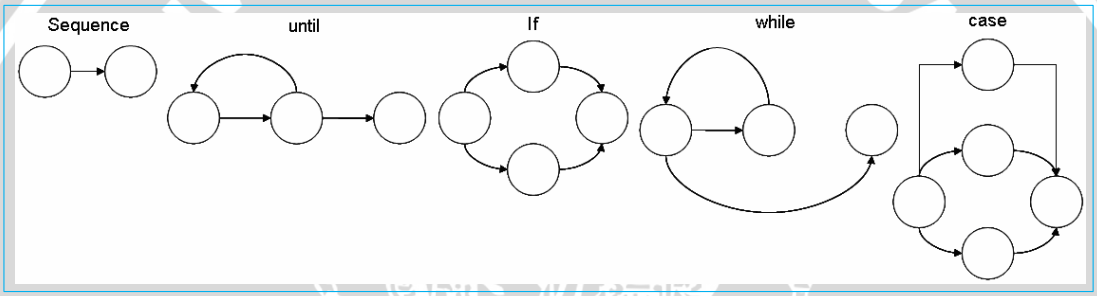
- Menjamin seluruh *independent path* di dalam modul yang dikerjakan sekurang-kurangnya sekali.

- Mengerjakan seluruh keputusan logikal.
- Mengerjakan seluruh *loop* yang sesuai dengan batasannya.
- Mengerjakan seluruh struktur data internal yang menjamin validitas.

Ada dua jenis pengujian pada *white box testing* yaitu pengujian *basis path* dan pengujian *loop.* [AN1-09]

a. Pengujian *Basis Path*

Merupakan teknik uji coba *white box* yang diusulkan Tom McCabe. Metode ini memungkinkan perancang *test case* mendapatkan ukuran kekompleksan *logical* dari perancangan prosedural dan menggunakan ukuran ini sebagai petunjuk untuk mendefinisikan basis set dari jalur pengerjaan. *Test case* yang didapat digunakan untuk mengerjakan basis set yang menjamin pengerjaan setiap perintah minimal satu kali selama uji coba.



Gambar 2. 15 Notasi diagram alir
Sumber: [AN1-09]

Pada diagram alir lingkaran *(node)*, menggambarkan satu atau lebih perintah prosedural. Urutan proses dan keputusan dapat dipetakan dalam satu *node*. Tanda panah *(edge)*, menggambarkan aliran kontrol. Setiap *node* harus mempunyai tujuan *node*. *Region* adalah daerah yang dibatasi oleh *edge* dan *node*. Termasuk daerah diluar diagram alir. Nomor pada *pseudo code* berhubungan dengan nomor *node*. Apabila ditemukan kondisi majemuk (*compound condition*) pada *pseudo code* pembuatan diagram alir menjadi rumit. Kondisi majemuk mungkin terjadi pada operator Boolean (`AND, OR, NAND, NOR`) yang dipakai pada perintah `if`.

Cyclomatic complexity adalah metrik *software* yang menyediakan ukuran kuantitatif dari kekompleksan logikal program. Apabila digunakan dalam konteks metode uji coba *basis path*, nilai yang dihitung untuk *cyclomatic complexity* menentukan jumlah jalur independen dalam basis set suatu program dan memberi batas atas untuk jumlah uji coba yang harus dikerjakan untuk menjamin bahwa seluruh perintah sekurang-kurangnya telah dikerjakan sekali. Jalur independen

adalah jalur yang melintasi atau melalui program dimana sekurang-kurangnya terdapat proses perintah yang baru atau kondisi yang baru.

Cyclomatic complexity digunakan untuk mencari jumlah *path* dalam satu *flowgraph*. Dapat dipergunakan rumusan sebagai berikut : [AN1-09]

1. Jumlah region grafik alir sesuai dengan *cyclomatic complexity*.
2. *Cyclomatix complexity* V(G) untuk grafik alir dihitung dengan rumus:

   V(G) = E - N + 2

   Dimana :

   E = jumlah *edge* pada grafik alir

   N = jumlah *node* pada grafik alir
3. *Cyclomatix complexity* V(G) juga dapat dihitung dengan rumus :

   V(G) = P + 1

   Dimana P = jumlah *predicate node* pada grafik alir

b. Pengujian *Loop*

Loop merupakan kendala yang sering muncul untuk menerapkan algoritma dengan tepat. Uji coba *loop* merupakan teknik pengujian *white box* yang fokusnya pada validitas dari *loop*. Kelas *loop* yaitu *loop* sederhana, *loop* tersarang, *loop* terangkai, dan *loop* tidak terstruktur.

2.8.1.2. Black Box Testing

Pengujian *black box* berfokus pada persyaratan fungsional perangkat lunak. Pengujian ini memungkinkan analis sistem memperoleh kumpulan kondisi input yang akan mengerjakan seluruh keperluan fungsional program. Tujuan metode ini mencari kesalahan pada : [AN1-09]

- Fungsi yang salah atau hilang.
- Kesalahan pada *interface*.
- Kesalahan pada struktur data atau akses *database*.
- Kesalahan performansi.
- Kesalahan inisialisasi dan tujuan akhir.

2.8.2. Strategi Pengujian

Strategi uji coba perangkat lunak memudahkan para perancang untuk menentukan keberhasilan sistem yang telah dikerjakan. Hal yang harus diperhatikan

adalah langkah-langkah perencanaan dan pelaksanaan harus direncanakan dengan baik dan berapa lama waktu, upaya dan sumber daya yang diperlukan. [AN1-09]

2.8.2.1. Pengujian Unit

Uji coba unit fokusnya pada usaha verifikasi pada unit terkecil dari desain perangkat lunak, yakni modul. Uji coba unit selalu berorientasi pada *white box testing* dan dapat dikerjakan paralel atau beruntun dengan modul lainnya.

2.8.2.2. Pengujian Integrasi

Pengujian terintegrasi adalah teknik yang sistematis untuk penyusunan struktur program, pada saat bersamaan dikerjakan uji coba untuk memeriksa kesalahan yang nantinya digabungkan dengan *interface*. Program sumber yang telah dikembangkan, ditunjang kembali dan diverifikasi untuk sintaksnya, maka perancangan *test case* dimulai. Peninjauan kembali perancangan informasi akan menyediakan petunjuk untuk menentukan *test case*. Karena modul bukan program yang berdiri sendiri maka *driver* (pengendali) dan atau *stub* perangkat lunak harus dikembangkan untuk pengujian unit.

Driver adalah program yang menerima data untuk *test case* dan menyalurkan ke modul yang diuji dan mencetak hasilnya. *Stub* melayani pemindahan modul yang akan dipanggil untuk diuji. Terdapat dua metode pengujian yaitu *top down integration* dan *bottom up integration*.

Top down integration merupakan pendekatan inkremental untuk penyusunan struktur program. Modul dipadukan dengan bergerak ke bawah melalui kontrol hirarki dimulai dari modul utama. Modul subordinat ke modul kontrol utama digabungkan ke dalam struktur baik menurut *depth first* atau *breadth first*. Proses integrasi pada top down integration : [AN1-09]

- Modul utama digunakan sebagai tes *driver* dan *stub* yang menggantikan seluruh modul yang secara langsung berada di bawah modul kontrol utama.
- Tergantung pada pendekatan perpaduan yang dipilih (*depth / breadth*).
- Uji coba dilakukan selama masing-masing modul dipadukan.
- Pada penyelesaian masing-masing uji coba stub yang lain dipindahkan dengan modul sebenarnya.
- Uji coba regression yaitu pengulangan pengujian untuk mencari kesalahan lain yang mungkin muncul.

Bottom up integration dinyatakan dengan penyusunan yang dimulai dan diujicobakan dengan *atomic* modul (modul tingkat paling bawah pada struktur program). Karena modul dipadukan dari bawah ke atas, proses yang diperlukan untuk modul subordinat yang selalu diberikan harus ada dan diperlukan untuk *stub* yang akan dihilangkan. Proses integrasi pada *bottom up integration* : [AN1-09]

- Modul tingkat bawah digabungkan ke dalam *cluster* yang memperlihatkan subfungsi perangkat lunak.
- *Driver* (program kontrol pengujian) ditulis untuk mengatur *input test case* dan *output*.
- *Cluster* diuji.
- *Driver* diganti dan *cluster* yang dikombinasikan dipindahkan ke atas pada struktur program.

2.8.2.3. Uji Coba Validasi

Setelah semua kesalahan diperbaiki maka langkah selanjutnya adalah *validasi testing*. Pengujian validasi dikatakan berhasil bila fungsi yang ada pada perangkat lunak sesuai dengan yang diharapkan pemakai. Validasi perangkat lunak merupakan kumpulan seri uji coba *black box* yang menunjukkan sesuai dengan yang diperlukan. Terdapat dua macam pengujian yaitu pengujian *alpha* dan pengujian *beta*.

BAB III
METODE PENELITIAN

Dalam penyusunan skripsi ini diperlukan data dan metode dalam penyelesaiannya. Langkah-langkah yang perlu dilakukan untuk membuat perangkat lunak adalah studi literatur, perancangan, implementasi, pengujian, pengambilan kesimpulan dan saran, dan penulisan laporan.

3.1. Studi Literatur

Melakukan studi literatur yaitu dengan pengumpulan dan pendalaman bahan pustaka yang dibutuhkan dalam penyelesaian tugas akhir ini. Bahan pustaka tersebut meliputi *supermarket*, basis data, UML, MySQL, PHP, XHTML MP, WCSS, dan pengujian perangkat lunak.

3.2. Perancangan

Pada perancangan terdiri dari dua tahap yaitu analisis kebutuhan dan perancangan perangkat lunak. Pada tahap analisis kebutuhan mencakup analisis sistem yang telah ada dan analisis kebutuhan sistem yang akan dirancang, yang terdiri dari daftar kebutuhan dan *use case diagram*.

Sedangkan pada tahap perancangan perangkat lunak akan dimulai dengan melakukan pembuatan *class diagram*, *sequence diagram*, *state diagram* dan perancangan basis data yang menggunakan *ER Diagram*, *Conceptual Data Model*, dan *Physical Data Model*.

3.3. Implementasi

Pada proses implementasi akan dilakukan pembuatan lingkungan implementasi untuk melakukan spesifikasi sistem. Kegiatan yang dilakukan adalah menerjemahkan dari bentuk perancangan yang ada dalam bentuk program ke dalam bahasa pemrograman XHTML MP, PHP, dan MySQL sebagai penyimpan data.

3.4. Pengujian

Pengujian dilakukan untuk mengetahui apakah sistem ini dapat berjalan dengan baik dan menemukan kesalahan aplikasi yang dibuat. Sehingga sebelum

aplikasi yang dibuat digunakan oleh pengguna, aplikasi ini diharapkan dapat sedikit mungkin atau bahkan tidak ada kesalahan pada saat digunakan nantinya. Proses pengujian yang dilakukan meliputi tiga tahapan yaitu pengujian unit, pengujian integrasi, dan pengujian validasi. Pengujian kecepatan *query* dalam mengakses *database* akan dilakukan terpisah.

3.5. Pengambilan Kesimpulan dan Saran

Berisi kesimpulan-kesimpulan yang diperoleh dari hasil pengujian dan saran yang sangat diperlukan untuk memperbaiki kesalahan-kesalahan yang terjadi untuk menyempurnakan tulisan.

3.6. Penulisan Laporan

Merupakan kegiatan penulisan laporan tugas akhir yang dibuat dan diharapkan dapat bermanfaat bagi pihak-pihak lain dan tidak menutup kemungkinan dapat dilakukan perbaikan dari saran-saran yang diajukan.

BAB IV
PERANCANGAN

Bab ini menjelaskan tentang analisis kebutuhan dan perancangan sistem Pengembangan *Portal Mobile Online Supermarket* dengan Teknologi XHTML MP dan WCSS. Diagram pohon analisis dan perancangan sistem *portal mobile online supermarket* ditampilkan pada Gambar 4.1.

Gambar 4. 1 Diagram pohon analisis dan perancangan sistem
Sumber: [Perancangan]

4.1. Analisis Kebutuhan

Analisis kebutuhan perangkat lunak adalah aktifitas rekayasa perangkat lunak yang menjembatani antara kebutuhan ditingkat sistem yang sudah ada dengan yang dirancang pada perancangan perangkat lunak [PRE-02]. Analisis kebutuhan perangkat lunak dapat juga diartikan proses yang digunakan untuk mendapatkan, menganalisis, dan memvalidasi kebutuhan-kebutuhan sistem [SOM-02].

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 yang Telah Ada

Tujuan dari pengembangan sebuah sistem adalah menghasilkan sesuatu yang dapat memenuhi kebutuhan dari pengguna yang akan menggunakannya. Tujuan ini dapat dipenuhi dalam suatu pengembangan sistem dengan cara melakukan analisis kebutuhan pada sistem tersebut.

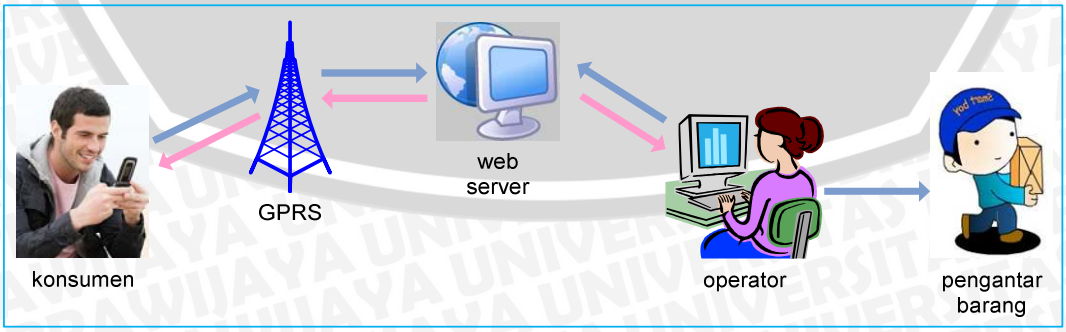
Proses pembelian barang di *supermarket* yang telah ada adalah konsumen datang ke *supermarket*, memilih barang dan membayar dikasir. Proses pembelian barang di *supermarket* ditunjukkan pada Gambar 4.2.

Gambar 4. 2 Pembelian barang di *supermarket*
Sumber: [Perancangan]

Proses pembelian barang di *mobile online supermarket* adalah konsumen dengan menggunakan perangkat *mobile* dapat melihat, memesan dan mendapatkan barang tanpa membutuhkan banyak waktu. Kelebihan yang dimiliki pada pembelian di *mobile online supermarket* adalah sebagai berikut:

a. Konsumen dapat memesan barang tanpa menghabiskan banyak waktu.
b. Konsumen tidak memerlukan banyak tenaga untuk mendapatkan barang.
c. Konsumen bisa menghemat uang karena pembayaran dilakukan pada saat barang diantar (*cash on delivery*).

Proses pembelian barang di *mobile online supermarket* ditunjukkan Gambar 4.3.



Gambar 4. 3 Pembelian barang di *mobile online supermarket*
Sumber: [Perancangan]

4.1.2. Analisis Kebutuhan Perangkat Lunak

Pengembangan sebuah perangkat lunak bertujuan untuk menghasilkan perangkat lunak yang dapat memenuhi kebutuhan *user*. Setiap pengembangan sebuah sistem perangkat lunak memerlukan adanya dokumentasi terhadap kebutuhan-kebutuhan *user* agar tujuan tersebut tercapai.

4.1.2.1. Daftar Kebutuhan

Daftar kebutuhan merupakan daftar yang menguraikan kebutuhan-kebutuhan pengguna 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 *mobile online supermarket*.

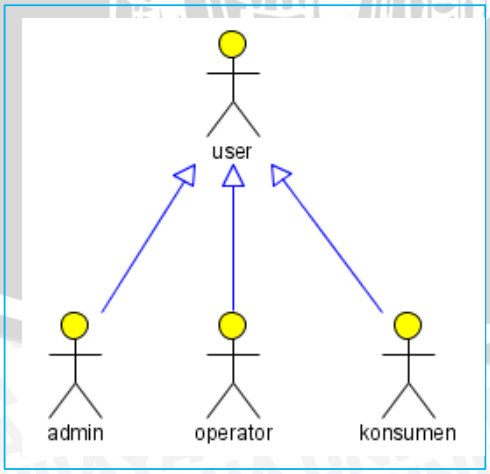
Tabel 4. 1 Deskripsi aktor

No.	Aktor	Keterangan
1	User	Aktor yang telah melakukan login. Setelah melakukan login aktor akan memiliki hak-hak akses yang sesuai pada sistem ini.
2	Konsumen	Aktor yang telah registrasi. Setelah melakukan login aktor dapat memesan barang.
3	Operator	Aktor yang bertugas memvalidasi konsumen dan melihat daftar pesanan barang.
4	Administrator	Aktor yang bertugas mengolah data barang dan data user.

Sumber: [Perancangan]

Operator dan administrator adalah pegawai dari *supermarket*.

Penyusunan daftar kebutuhan fungsional dari sistem *mobile online supermarket* dilakukan setelah identifikasi aktor. Daftar kebutuhan fungsional sistem disertai dengan nama *use case* yang merepresentasikan fungsionalitas dari kebutuhan tersebut.



Gambar 4. 4 Generalisasi aktor-aktor pengguna sistem *mobile online supermarket*
Sumber: [Perancangan]

Penyusunan daftar kebutuhan fungsional dari sistem *mobile online supermarket* dilakukan setelah identifikasi aktor. Daftar kebutuhan fungsional sistem disertai dengan nama *use case* yang merepresentasikan fungsionalitas dari kebutuhan tersebut. Daftar kebutuhan fungsional tersebut ditunjukkan pada Tabel 4. 2.

Tabel 4. 2 Daftar kebutuhan fungsional

No.	*Use Case*	Kebutuhan
1.	Login	Sistem harus memberikan fasilitas *login*, sehingga hanya aktor tertentu yang dapat mengakses fasilitas tertentu.
2.	Logout	Sistem harus menyediakan fasilitas *logout*, agar aktor yang telah *login* dapat keluar dari sistem.
3.	Melihat daftar pegawai	Sistem harus menyediakan fasilitas untuk melihat semua pegawai yang ada di dalam sistem. Data yang dapat dilihat adalah kode, nama, dan tugas.
4.	Melihat data pegawai	Sistem harus menyediakan fasilitas untuk melihat data pegawai tertentu. Data yang dapat dilihat kode, *username*, nama, *sex*, tempat lahir, tanggal lahir, alamat, no. telepon, *email*, dan tugas.
5.	Menambah pegawai	Sistem harus menyediakan fasilitas untuk menambah pegawai. Data yang dicatat adalah kode, nama, *username*, *password*, *sex*, tempat lahir, tanggal lahir, alamat, no. telepon, *email*, dan tugas.
6.	Mengubah data pegawai	Sistem harus menyediakan fasilitas untuk mengubah data pegawai tertentu. Data yang dapat diubah adalah kode, nama, *username*, *password*, tempat lahir, tanggal lahir, alamat, no. telepon, dan *email*.
7.	Menghapus pegawai	Sistem harus menyediakan fasilitas untuk menghapus data pegawai tertentu.
8.	Mencari pegawai	Sistem harus menyediakan fasilitas untuk mencari pegawai tertentu. Pencarian pegawai berdasarkan nama pegawai.
9.	Melihat daftar barang	Sistem harus menyediakan fasilitas untuk melihat daftar semua barang yang ada pada sistem. Data yang dapat dilihat adalah kode, nama, harga, dan stok.
10.	Melihat data barang	Sistem harus menyediakan fasilitas untuk melihat data sebuah barang. Data yang dapat dilihat kode, nama, kategori, gambar, harga, suplier, deskripsi, dan stok.
11.	Menambah barang	Sistem harus menyediakan fasilitas untuk menambah barang. Data yang dicatat pada penambahan barang adalah kode, nama, kategori, gambar, harga, suplier, deskripsi, dan stok.
12.	Menambah stok barang	Sistem harus menyediakan fasilitas untuk menambah stok barang. Data yang dicatat pada penambahan stok barang adalah stok tambah.
13.	Mengubah data barang	Sistem harus menyediakan fasilitas untuk mengubah data suatu barang. Data yang dapat diubah adalah kode, nama, kategori, gambar, harga, suplier, deskripsi, dan stok.
14.	Menghapus barang	Sistem harus menyediakan fasilitas untuk menghapus data barang tertentu.
15.	Mencari barang	Sistem harus menyediakan fasilitas untuk mencari barang tertentu. Pencarian barang berdasarkan nama barang.
16.	Melihat daftar kategori barang	Sistem harus menyediakan fasilitas untuk melihat daftar semua kategori barang yang ada pada sistem. Data yang dapat dilihat adalah kode dan nama.
17.	Melihat data kategori barang	Sistem harus menyediakan fasilitas untuk melihat data suatu kategori barang. Data yang dapat dilihat kode, nama, dan deskripsi.
18.	Menambah kategori barang	Sistem harus menyediakan fasilitas untuk menambah kategori. Data yang harus dicatat pada penambahan kategori barang adalah kode, nama, dan deskripsi.
19.	Mengubah data kategori barang	Sistem harus menyediakan fasilitas untuk mengubah data kategori barang. Data yang dapat diubah adalah kode, nama, dan deskripsi.
20.	Menghapus kategori barang	Sistem harus menyediakan fasilitas untuk menghapus kategori barang tertentu.
21.	Mencari kategori barang	Sistem harus menyediakan fasilitas untuk mencari kategori barang tertentu. Pencarian kategori barang berdasarkan nama kategori barang.
22.	Melihat daftar subkategori barang	Sistem harus menyediakan fasilitas untuk melihat daftar subkategori per kategori barang yang ada pada sistem. Data yang dapat dilihat adalah kode dan nama.
23.	Menambah subkategori barang	Sistem harus menyediakan fasilitas untuk menambah subkategori. Data yang harus dicatat pada penambahan subkategori barang adalah kode dan nama.

24.	Mengubah data subkategori barang	Sistem harus menyediakan fasilitas untuk mengubah data subkategori barang. Data yang dapat diubah adalah kode dan nama.
25.	Menghapus subkategori barang	Sistem harus menyediakan fasilitas untuk menghapus subkategori barang tertentu.
26.	Melihat daftar suplier	Sistem harus menyediakan fasilitas untuk melihat daftar supplier yang ada di dalam sistem. Data yang dapat dilihat adalah kode, nama, dan no. telp.
27.	Melihat data suplier	Sistem harus menyediakan fasilitas untuk melihat data suplier tertentu. Data yang dapat dilihat kode, nama, alamat, dan no. telp.
28.	Menambah suplier	Sistem harus menyediakan fasilitas untuk menambah suplier. Data yang harus dicatat adalah kode, nama, alamat, dan no. telp.
29.	Mengubah data suplier	Sistem harus menyediakan fasilitas untuk mengubah data suplier tertentu. Data yang dapat diubah adalah kode, nama, alamat, dan no. telp.
30.	Menghapus suplier	Sistem harus menyediakan fasilitas untuk menghapus suplier tertentu.
31.	Mencari suplier	Sistem harus menyediakan fasilitas untuk mencari suplier tertentu. Pencarian suplier berdasarkan nama suplier.
32.	Melihat daftar konsumen	Sistem harus menyediakan fasilitas untuk melihat daftar konsumen yang ada di dalam sistem. Data yang dapat dilihat adalah nama dan status validasi.
33.	Menghapus konsumen	Sistem harus menyediakan fasilitas untuk menghapus konsumen tertentu.
34.	Mencari konsumen	Sistem harus menyediakan fasilitas untuk mencari konsumen tertentu. Pencarian konsumen berdasarkan status validasi.
35.	Melihat daftar validasi konsumen	Sistem harus menyediakan fasilitas untuk melihat daftar validasi konsumen yang telah registrasi. Data yang dapat dilihat adalah nama, *no. handphone*, dan status validasi.
36.	Mengubah data konsumen	Sistem harus menyediakan fasilitas untuk mengubah data konsumen tertentu. Data yang diubah adalah status validasi.
37.	Memesan barang	Sistem harus menyediakan fasilitas untuk melakukan pemesanan barang.
38.	Melihat keranjang belanja	Sistem harus menyediakan fasilitas untuk melihat daftar barang yang dipesan. Data yang dapat dilihat adalah nama, harga, jumlah, sub total, dan total belanja.
39.	Mengubah data keranjang belanja	Sistem harus menyediakan fasilitas untuk mengubah data barang yang dipesan. Data yang dapat diubah adalah jumlah barang.
40.	Menghapus keranjang belanja	Sistem harus menyediakan fasilitas untuk menghapus barang tertentu.
41.	Melihat detail pesanan	Sistem harus menyediakan fasilitas untuk melihat data dan barang yang dipesan.
42.	Check out	Sistem harus menyediakan fasilitas untuk menampilkan data transaksi.
43.	Melihat daftar pesanan	Sistem harus menyediakan fasilitas untuk melihat daftar pesanan konsumen. Data yang dapat dilihat adalah kode transaksi, waktu pesan, dan status kirim.
44.	Mencetak nota	Sistem harus menyediakan fasilitas untuk mencetak nota sebagai bukti transaksi.

Sumber: [Perancangan]

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

1. Modul pendukung sistem
2. Modul penyediaan barang
3. Modul konsumen
4. Modul pendukung pemesanan barang

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.	Pendukung sistem	1. Login
		2. Logout
		3. Melihat daftar pegawai
		4. Melihat data pegawai
		5. Menambah pegawai
		6. Mengubah data pegawai
		7. Menghapus pegawai
		8. Mencari pegawai
2.	Penyediaan barang	1. Melihat daftar barang
		2. Melihat data barang
		3. Menambah barang
		4. Menambah stok barang
		5. Mengubah data barang
		6. Menghapus barang
		7. Mencari barang
		8. Melihat daftar kategori barang
		9. Melihat data kategori barang
		10. Menambah kategori barang
		11. Mengubah data kategori barang
		12. Menghapus kategori barang
		13. Mencari kategori barang
		14. Melihat daftar subkategori barang
		15. Menambah subkategori barang
		16. Mengubah data subkategori barang
		17. Menghapus subkategori barang
		18. Melihat daftar suplier
		19. Melihat data suplier
		20. Menambah suplier
		21. Mengubah data suplier
		22. Menghapus suplier
		23. Mencari suplier
3.	Konsumen	1. Melihat daftar konsumen
		2. Menghapus konsumen
		3. Mencari konsumen
		4. Melihat daftar validasi konsumen
		5. Mengubah data konsumen
4.	Pendukung pemesanan barang	1. Memesan barang
		2. Melihat keranjang belanja
		3. Mengubah data keranjang belanja
		4. Menghapus keranjang belanja
		5. Melihat detail pesanan
		6. Check out
		7. Melihat daftar pesanan
		8. Mencetak nota

Sumber: [Perancangan]

Daftar kebutuhan non fungsional sistem *mobile online supermarket* ditunjukkan pada Tabel 4.4.

Tabel 4. 4 Daftar kebutuhan non fungsional sistem *mobile online supermarket*

No.	Kebutuhan non fungsional
1.	Sistem dikembangkan dengan menggunakan XHTML MP, WCSS, dan PHP
2.	Sistem harus dapat diakses dengan *web browser* dan *mobile browser*

Sumber: [Perancangan]

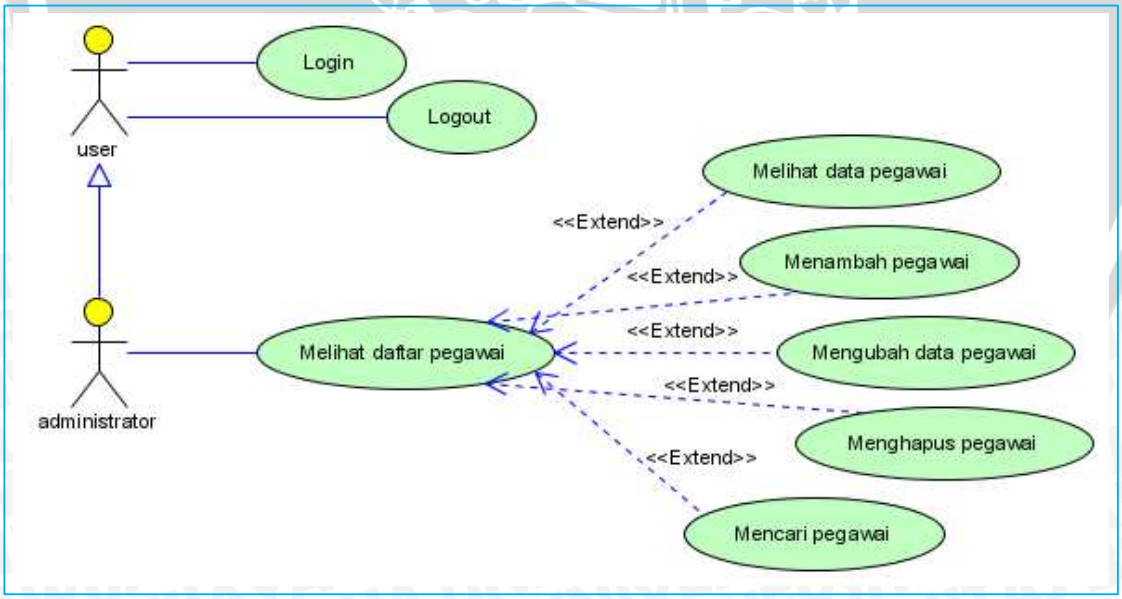
4.1.2.2. *Use Case Diagram*

Use case adalah deskripsi fungsi dari sebuah sistem dari perspektif pengguna. *Use case* bekerja dengan cara mendeskripsikan tipikal interaksi antara *user* (pengguna) sebuah sistem dengan sistemnya sendiri melalui sebuah cerita bagaimana sebuah sistem dipakai. [MUN-05]

Pemodelan dalam *use case diagram* yang menggambarkan fungsionalitas yang disediakan oleh sistem *mobile online supermarket* dibagi menjadi empat diagram yang bersesuaian dengan modul dalam sistem *mobile online supermarket*.

4.1.2.2.1. *Use Case Diagram* untuk Modul Pendukung Sistem *Supermarket Online*

Use case diagram untuk modul pendukung sistem *mobile online supermarket* ditunjukkan pada Gambar 4.5. Diagram *use case* ini melibatkan 2 aktor dan 8 *use case*. Kedelapan *use case* yang termasuk dalam modul pendukung sistem *mobile online supermarket* dapat dilihat pada Tabel 4.3. Kedua aktor yang terlibat dalam modul pendukung sistem *mobile online supermarket* adalah *user* dan administrator.



Gambar 4. 5 *Use case diagram* untuk modul pendukung sistem *mobile online supermarket*
Sumber: [Perancangan]

1. ***Use Case Spesification* Login**

Tabel 4. 5 *Use case specification* Login

Nama *use case*	Login
Aktor	*User*

Deskripsi	Sistem harus memberikan fasilitas *login*, sehingga hanya aktor tertentu yang dapat mengakses fasilitas tertentu.
Kondisi awal	Aktor menjalankan sistem pada halaman pertama.
Kondisi akhir	User telah *login* sebagai konsumen, operator atau administrator.
Aliran Utama	
Aksi dari Aktor	**Tanggapan dari Sistem**
1. User memasukkan *username*, *password*, dan menekan tombol "Login".	2. Sistem memvalidasi *username* dan *password*.
Aliran Alternatif 1: Pasangan *username* dan *password* tidak ada di dalam sistem. (User pada langkah nomer 1 aliran utama tidak memasukkan *username* atau *password* yang sesuai)	
Aksi dari Aktor	**Tanggapan dari Sistem**
	1. Sistem kembali pada langkah nomer 1 aliran utama.

Sumber: [Perancangan]

2. *Use Case Spesification* Logout

Tabel 4. 6 *Use case specification* Logout

Nama *use case*	Logout
Aktor	*User*
Deskripsi	Sistem harus menyediakan fasilitas *logout*, agar aktor yang telah *login* dapat keluar dari sistem.
Kondisi awal	User telah *login*.
Kondisi akhir	User telah *logout*.
Aliran Utama	
Aksi dari Aktor	**Tanggapan dari Sistem**
1. Aktor menekan tombol "Logout".	2. Sistem melakukan *logout* untuk user tersebut.

Sumber: [Perancangan]

3. *Use Case Spesification* Melihat Daftar Pegawai

Tabel 4. 7 *Use case specification* Melihat daftar pegawai

Nama *use case*	Melihat daftar pegawai
Aktor	Administrator
Deskripsi	Sistem harus menyediakan fasilitas untuk melihat semua pegawai yang ada di dalam sistem.
Kondisi awal	Sistem telah berjalan dan *user* telah *login* sebagai administrator.
Kondisi akhir	Sistem menampilkan daftar pegawai.
Aliran Utama	
Aksi dari Aktor	**Tanggapan dari Sistem**
1. Administrator memilih menu "Pegawai".	2. Sistem menampilkan daftar pegawai yang ada di dalam sistem. Data yang ditampilkan adalah kode, nama, dan tugas.

Sumber: [Perancangan]

4. *Use Case Spesification* Melihat Data Pegawai

Tabel 4. 8 *Use case specification* Melihat data pegawai

Nama *use case*	Melihat data pegawai
Aktor	Administrator
Deskripsi	Sistem harus menyediakan fasilitas untuk melihat data pegawai tertentu yang ada di dalam sistem.
Kondisi awal	Administrator telah menjalankan *use case* Melihat daftar pegawai.
Kondisi akhir	Sistem menampilkan data pegawai tertentu.
Aliran Utama	
Aksi dari Aktor	**Tanggapan dari Sistem**
1. Administrator memilih 'kode' salah satu baris dari daftar pegawai yang ditampilkan dari *use case* Melihat daftar pegawai.	2. Sistem menampilkan data pegawai yang ada di dalam sistem. Data yang ditampilkan adalah kode, *username*, nama, *sex*, tempat

	lahir, tanggal lahir, alamat, no. telepon, *email*, dan tugas.

Sumber: [Perancangan]

5. *Use Case Spesification* Menambah Pegawai

Tabel 4. 9 *Use case specification* Menambah pegawai

Nama *use case*	Menambah pegawai
Aktor	Administrator
Deskripsi	Sistem harus menyediakan fasilitas untuk menambah pegawai. Data yang harus dicatat adalah kode, nama, *username*, *password*, *sex*, tempat lahir, tanggal lahir, alamat, no. telepon, *email*, dan tugas.
Kondisi awal	Administrator telah menjalankan *use case* Melihat daftar pegawai.
Kondisi akhir	Pegawai telah ditambahkan ke dalam sistem.
Aliran Utama	
Aksi dari Aktor	**Tanggapan dari Sistem**
1. Administrator memilih menu "Tambah Pegawai".	2. Sistem menampilkan *form* masukkan yang terdiri dari kode, nama, *username*, *password*, *sex*, tempat lahir, tanggal lahir, alamat, no. telepon, *email*, dan tugas.
3. Administrator memasukkan kode, nama, *username*, *password*, *sex*, tempat lahir, tanggal lahir, alamat, no. telepon, *email*, dan tugas.	4. Sistem memvalidasi masukkan. Kode, nama, *username*, *password*, *sex*, tempat lahir, alamat, no. telepon, *email*, tugas tidak boleh kosong dan kode atau *username* tersebut belum ada dalam sistem.
	5. Sistem menampilkan pesan bahwa data berhasil ditambahkan.
Aliran Alternatif 1: Masukkan kode, nama, *username*, *password*, *sex*, tempat lahir, alamat, no. telepon, *email*, atau tugas kosong. (Administrator pada langkah nomer 3 aliran utama tidak memasukkan kode, nama, *username*, *password*, *sex*, tempat lahir, alamat, no. telepon, *email*, atau tugas)	
Aksi dari Aktor	**Tanggapan dari Sistem**
	1. Sistem menampilkan pesan kesalahan bahwa kode, nama, *username*, *password*, *sex*, tempat lahir, alamat, no. telepon, *email*, atau tugas tidak boleh kosong.
	2. Sistem kembali pada langkah nomer 2 aliran utama.
Aliran Alternatif 2: Kode, *username*, atau *email* telah ada dalam sistem (Administrator pada langkah nomer 3 aliran utama memasukkan kode, *username*, atau *email* yang telah ada dalam sistem)	
Aksi dari Aktor	**Tanggapan dari Sistem**
	1. Sistem menampilkan pesan kesalahan bahwa kode, *username*, atau *email* tersebut telah ada dalam sistem.
	2. Sistem kembali pada langkah nomer 2 aliran utama.
Aturan khusus 1. Masukkan kode, nama, *username*, *password*, *sex*, tempat lahir, alamat, no. telepon, *email*, dan tugas tidak boleh kosong. 2. Masukkan kode, *username*, dan *email* belum ada dalam sistem.	

Sumber: [Perancangan]

6. *Use Case Spesification* Mengubah Data Pegawai

Tabel 4. 10 *Use case specification* Mengubah data pegawai

Nama *use case*	Mengubah data pegawai
Aktor	Administrator
Deskripsi	Sistem harus menyediakan fasilitas untuk mengubah data pegawai tertentu. Data yang dapat diubah adalah kode, nama, tempat lahir, tanggal lahir,

	alamat, no. telepon, dan *email*.
Kondisi awal	Administrator telah menjalankan *use case* Melihat daftar pegawai.
Kondisi akhir	Data pegawai yang telah diubah disimpan ke dalam sistem.

Aliran Utama	
Aksi dari Aktor	**Tanggapan dari Sistem**
1. Administrator memilih menu "Ubah" salah satu baris dari daftar pegawai yang ditampilkan dari *use case* Melihat daftar pegawai.	2. Sistem menampilkan *form* masukkan kode, nama, tempat lahir, tanggal lahir, alamat, no. telepon, dan *email* dari baris pegawai yang telah dipilih administrator.
3. Administrator mengubah data untuk kode, nama, tempat lahir, tanggal lahir, alamat, no. telepon, dan *email.*	4. Sistem memvalidasi masukkan. kode, nama, tempat lahir, alamat, no. telepon, dan *email* tidak boleh kosong.
	5. Sistem menampilkan pesan bahwa data berhasil diubah.

Aliran Alternatif 1: Masukkan kode, nama, tempat lahir, alamat, no. telepon, atau *email* kosong. (Administrator pada langkah nomer 3 aliran utama tidak memasukkan kode, nama, tempat lahir, alamat, no. telepon, atau *email*)	
Aksi dari Aktor	**Tanggapan dari Sistem**
	1. Sistem menampilkan pesan kesalahan bahwa kode, nama, tempat lahir, alamat, no. telepon, atau *email* tidak boleh kosong.
	2. Sistem kembali pada langkah nomer 2 aliran utama.

Aliran Alternatif 2: Kode, *username,* atau *email* telah ada dalam sistem (Administrator pada langkah nomer 3 aliran utama memasukkan kode, *username,* atau *email* yang telah ada dalam sistem)	
Aksi dari Aktor	**Tanggapan dari Sistem**
	1. Sistem menampilkan pesan kesalahan bahwa kode, *username*, atau *email* tersebut telah ada dalam sistem.
	2. Sistem kembali pada langkah nomer 2 aliran utama.

Aturan khusus
1. Masukkan kode, nama, tempat lahir, alamat, no. telepon, dan *email* tidak boleh kosong.
2. Masukkan kode, *username*, dan *email* belum ada dalam sistem.

Sumber: [Perancangan]

7. *Use Case Spesification* Menghapus Pegawai

Tabel 4. 11 *Use case specification* Menghapus pegawai

Nama *use case*	Menghapus pegawai
Aktor	Administrator
Deskripsi	Sistem harus menyediakan fasilitas untuk menghapus pegawai tertentu.
Kondisi awal	Administrator telah menjalankan *use case* Melihat daftar pegawai.
Kondisi akhir	Data pegawai telah dihapus dari dalam sistem.

Aliran Utama	
Aksi dari Aktor	**Tanggapan dari Sistem**
1. Administrator memilih menu "Hapus" salah satu baris dari daftar pegawai yang ditampilkan dari *use case* Melihat daftar pegawai.	2. Sistem menampilkan pesan bahwa data berhasil dihapus.

Sumber: [Perancangan]

8. *Use Case Spesification* Mencari Pegawai

Tabel 4. 12 *Use case specification* Mencari pegawai

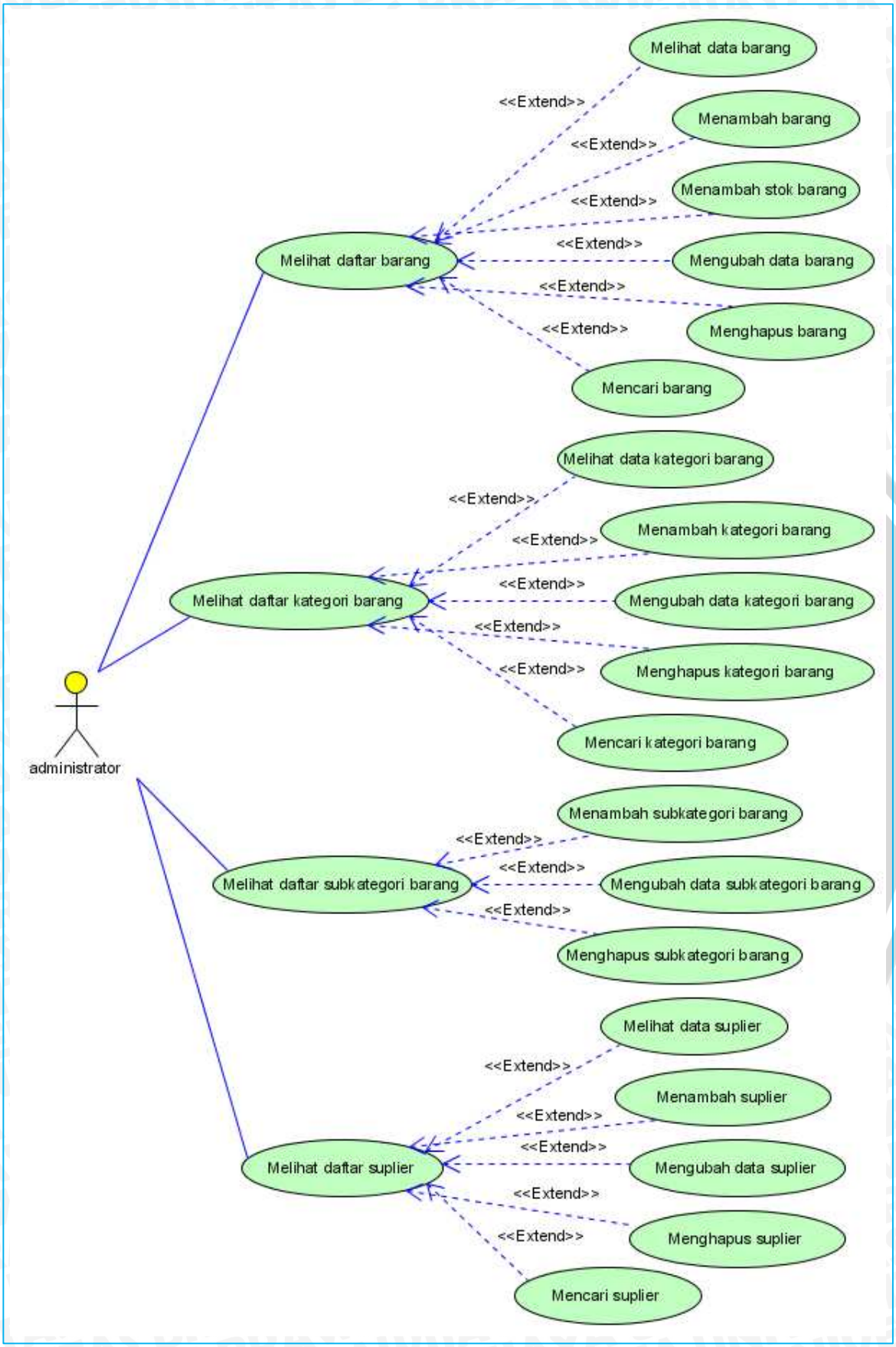
Nama *use case*	Mencari pegawai
Aktor	Administrator
Deskripsi	Sistem harus menyediakan fasilitas untuk mencari pegawai tertentu. Pencarian pegawai berdasarkan nama pegawai.

Kondisi awal	User telah *login* sebagai administrator.
Kondisi akhir	Sistem menampilkan pegawai yang dicari oleh administrator.
Aliran Utama	
Aksi dari Aktor	**Tanggapan dari Sistem**
1. Administrator memilih menu "Pegawai".	2. Sistem menampilkan semua daftar pegawai yang ada di dalam sistem dan *form* pencarian.
3. Administrator memasukkan nama sebagai *keyword* dan menekan tombol "Cari".	4. Sistem menampilkan pegawai yang dicari.
Aliran Alternatif 1: Masukkan *keyword* kosong (Administrator pada langkah nomer 3 aliran utama tidak memasukkan *keyword*)	
Aksi dari Aktor	**Tanggapan dari Sistem**
	1. Sistem kembali ke langkah nomer 2 aliran utama.
Aturan khusus	
1. Masukkan *keyword* tidak boleh kosong.	

Sumber: [Perancangan]

4.1.2.2.2. *Use Case Diagram* untuk Modul Penyediaan Barang

Use case diagram untuk modul penyediaan barang *mobile online supermarket* ditunjukkan pada Gambar 4.6. Diagram *use case* ini melibatkan 1 aktor dan 23 *use case*. Kedua puluh tiga *use case* yang termasuk dalam modul penyediaan barang *mobile online supermarket* dapat dilihat pada Tabel 4.3. Aktor yang terlibat dalam modul penyediaan barang *mobile online supermarket* adalah administrator.



Gambar 4. 6 *Use case diagram* untuk modul penyediaan barang *mobile online supermarket*
Sumber: [Perancangan]

1. *Use Case Spesification* Melihat Daftar Barang

Tabel 4. 13 *Use case specification* Melihat daftar barang

Nama *use case*	Melihat daftar barang
Aktor	Administrator
Deskripsi	Sistem harus menyediakan fasilitas untuk melihat daftar semua barang yang ada pada sistem.
Kondisi awal	Sistem telah berjalan dan *user* telah *login* sebagai administrator.
Kondisi akhir	Sistem menampilkan daftar barang.
Aliran Utama	
Aksi dari Aktor	**Tanggapan dari Sistem**
1. Administrator memilih menu “Barang”.	2. Sistem menampilkan daftar semua barang.

Sumber: [Perancangan]

2. *Use Case Spesification* Melihat Data Barang

Tabel 4. 14 *Use case specification* Melihat data barang

Nama *use case*	Melihat data barang
Aktor	Administrator
Deskripsi	Sistem harus menyediakan fasilitas untuk melihat data barang tertentu yang ada di dalam sistem.
Kondisi awal	Administrator telah menjalankan *use case* Melihat daftar barang.
Kondisi akhir	Sistem menampilkan data barang tertentu.
Aliran Utama	
Aksi dari Aktor	**Tanggapan dari Sistem**
1. Administrator memilih ‘kode’ salah satu baris dari daftar barang yang ditampilkan dari *use case* Melihat daftar barang.	2. Sistem menampilkan data barang yang ada di dalam sistem. Data yang ditampilkan adalah kode, nama, kategori, gambar, harga, suplier, deskripsi, dan stok .

Sumber: [Perancangan]

3. *Use Case Spesification* Menambah Barang

Tabel 4. 15 *Use case specification* Menambah barang

Nama *use case*	Menambah barang
Aktor	Administrator
Deskripsi	Sistem harus menyediakan fasilitas untuk menambah data suatu barang. Data yang harus dicatat pada penambahan data barang adalah kode, nama, kategori, gambar, harga, suplier, deskripsi, dan stok.
Kondisi awal	Administrator telah menjalankan *use case* Melihat daftar barang.
Kondisi akhir	Barang telah ditambahkan ke dalam sistem.
Aliran Utama	
Aksi dari Aktor	**Tanggapan dari Sistem**
1. Administrator memilih link “Tambah Barang”.	2. Sistem menampilkan *form* masukkan yang terdiri dari kode, nama, kategori, gambar, harga, suplier, deskripsi, dan stok.
3. Administrator memasukkan kode, nama, kategori, gambar, harga, suplier, deskripsi, dan stok.	4. Sistem memvalidasi masukkan. Kode, nama, dan harga tidak boleh kosong.
	5. Sistem menampilkan pesan bahwa data berhasil ditambahkan.
Aliran Alternatif 1: Masukkan kode, nama, atau harga kosong. (Administrator pada langkah nomer 3 aliran utama tidak memasukkan kode, nama, atau harga)	
Aksi dari Aktor	**Tanggapan dari Sistem**
	1. Sistem menampilkan pesan kesalahan bahwa kode, nama, atau harga tidak boleh kosong.
	2. Sistem kembali pada langkah nomer 2 aliran utama.

Aliran Alternatif 2: Kode telah ada dalam sistem (Administrator pada langkah nomer 3 aliran utama memasukkan kode telah ada dalam sistem)	
Aksi dari Aktor	**Tanggapan dari Sistem**
	1. Sistem menampilkan pesan kesalahan bahwa kode tersebut telah ada dalam sistem.
	2. Sistem kembali pada langkah nomer 2 aliran utama.
Aturan khusus 1. Masukkan kode, nama, dan harga tidak boleh kosong. 2. Masukkan kode belum ada dalam sistem.	

Sumber: [Perancangan]

4. *Use Case Spesification* Menambah Stok Barang

Tabel 4. 16 *Use case specification* Menambah stok barang

Nama *use case*	Menambah stok barang	
Aktor	Administrator	
Deskripsi	Sistem harus menyediakan fasilitas untuk menambah stok suatu barang. Data yang harus dicatat pada penambahan data barang adalah stok tambah.	
Kondisi awal	Administrator telah menjalankan *use case* Melihat daftar barang.	
Kondisi akhir	Stok barang telah ditambahkan ke dalam sistem.	
Aliran Utama		
Aksi dari Aktor		**Tanggapan dari Sistem**
1. Administrator memilih link "Stok" salah satu baris dari data barang yang ditampilkan dari *use case* Melihat daftar barang.		2. Sistem menampilkan nama barang, stok awal, dan *form* masukkan yaitu stok tambah.
3. Administrator memasukkan stok tambah.		4. Sistem menampilkan pesan bahwa stok barang berhasil ditambahkan.

Sumber: [Perancangan]

5. *Use Case Spesification* Mengubah Data Barang

Tabel 4. 17 *Use case specification* Mengubah data barang

Nama *use case*	Mengubah data barang	
Aktor	Administrator	
Deskripsi	Sistem harus menyediakan fasilitas untuk mengubah data suatu barang. Data yang dapat diubah adalah kode, nama, kategori, gambar, harga, suplier, deskripsi, dan stok.	
Kondisi awal	Administrator telah menjalankan *use case* Melihat daftar barang.	
Kondisi akhir	Data barang yang telah diubah disimpan ke dalam sistem.	
Aliran Utama		
Aksi dari Aktor		**Tanggapan dari Sistem**
1. Administrator menekan tombol "Ubah" salah satu baris dari data barang yang ditampilkan dari *use case* Melihat daftar barang.		2. Sistem menampilkan kode, nama, kategori, gambar, harga, suplier, deskripsi dan stok dari baris barang yang telah dipilih administrator.
3. Administrator mengubah data untuk kode, nama, kategori, gambar, harga, suplier, deskripsi, dan stok.		4. Sistem memvalidasi masukkan. Kode, nama, dan harga tidak boleh kosong.
		5. Sistem menampilkan pesan bahwa data berhasil diubah.
Aliran Alternatif 1: Masukkan kode, nama, atau harga kosong. (Administrator pada langkah nomer 3 aliran utama tidak memasukkan kode, nama, atau harga)		
Aksi dari Aktor		**Tanggapan dari Sistem**
		1. Sistem menampilkan pesan kesalahan bahwa kode, nama, atau harga tidak boleh kosong.
		2. Sistem kembali pada langkah nomer 2 aliran utama.

Aliran Alternatif 2: Kode telah ada dalam sistem (Administrator pada langkah nomer 3 aliran utama memasukkan kode telah ada dalam sistem)	
Aksi dari Aktor	**Tanggapan dari Sistem**
	1. Sistem menampilkan pesan kesalahan bahwa kode tersebut telah ada dalam sistem.
	2. Sistem kembali pada langkah nomer 2 aliran utama.
Aturan khusus	
1. Masukkan kode, nama, dan harga tidak boleh kosong. 2. Masukkan kode belum ada dalam sistem.	

Sumber: [Perancangan]

6. *Use Case Spesification* Menghapus Barang

Tabel 4. 18 *Use case specification* Menghapus barang

Nama *use case*	Menghapus barang
Aktor	Administrator
Deskripsi	Sistem harus menyediakan fasilitas untuk menghapus barang tertentu.
Kondisi awal	Administrator telah menjalankan *use case* Melihat daftar barang.
Kondisi akhir	Barang telah dihapus dari dalam sistem.
Aliran Utama	
Aksi dari Aktor	**Tanggapan dari Sistem**
1. Administrator memilih menu "Hapus" salah satu baris dari daftar barang yang ditampilkan dari *use case* Melihat daftar barang.	2. Sistem menampilkan pesan bahwa data berhasil dihapus.

Sumber: [Perancangan]

7. *Use Case Spesification* Mencari Barang

Tabel 4. 19 *Use case specification* Mencari barang

Nama *use case*	Mencari barang
Aktor	Administrator
Deskripsi	Sistem harus menyediakan fasilitas untuk mencari suatu barang. Pencarian barang berdasarkan nama barang.
Kondisi awal	User telah *login* sebagai administrator.
Kondisi akhir	Sistem menampilkan data barang yang dicari oleh administrator.
Aliran Utama	
Aksi dari Aktor	**Tanggapan dari Sistem**
1. Administrator memilih menu "Barang".	2. Sistem menampilkan semua daftar barang yang ada di dalam sistem dan *form* pencarian.
3. Administrator memasukkan nama barang sebagai *keyword* dan menekan tombol "Cari".	4. Sistem menampilkan barang yang dicari.
Aliran Alternatif 1: Masukkan *keyword* kosong (Administrator pada langkah nomer 3 aliran utama tidak memasukkan *keyword*)	
Aksi dari Aktor	**Tanggapan dari Sistem**
	1. Sistem kembali ke langkah nomer 2 aliran utama.
Aturan khusus	
1. Masukkan *keyword* tidak boleh kosong.	

Sumber: [Perancangan]

8. *Use Case Spesification* Melihat Daftar Kategori Barang

Tabel 4. 20 *Use case specification* Melihat daftar kategori barang

Nama *use case*	Melihat daftar kategori barang
Aktor	Administrator
Deskripsi	Sistem harus menyediakan fasilitas untuk melihat daftar semua kategori barang yang ada pada sistem. Data yang dapat dilihat adalah kode dan nama.
Kondisi awal	Administrator telah menjalankan *use case* Melihat daftar barang.
Kondisi akhir	Sistem menampilkan daftar kategori barang.

Aliran Utama	
Aksi dari Aktor	**Tanggapan dari Sistem**
1. Administrator memilih menu “Kategori”.	2. Sistem menampilkan daftar kategori barang yang ada di dalam sistem. Data yang ditampilkan adalah kode dan nama.

Sumber: [Perancangan]

9. *Use Case Spesification* Melihat Data Kategori Barang

Tabel 4. 21 *Use case specification* Melihat data kategori barang

Nama *use case*	Melihat data kategori barang
Aktor	Administrator
Deskripsi	Sistem harus menyediakan fasilitas untuk melihat data kategori barang tertentu yang ada di dalam sistem.
Kondisi awal	Administrator telah menjalankan *use case* Melihat daftar kategori barang.
Kondisi akhir	Sistem menampilkan data kategori barang tertentu.
Aliran Utama	
Aksi dari Aktor	**Tanggapan dari Sistem**
1. Administrator memilih ‘kode’ salah satu baris dari daftar kategori barang yang ditampilkan dari *use case* Melihat daftar kategori barang.	2. Sistem menampilkan data kategori barang yang ada di dalam sistem. Data yang ditampilkan adalah kode, nama, dan deskripsi.

Sumber: [Perancangan]

10. *Use Case Spesification* Menambah Kategori Barang

Tabel 4. 22 *Use case specification* Menambah kategori barang

Nama *use case*	Menambah kategori barang
Aktor	Administrator
Deskripsi	Sistem harus menyediakan fasilitas untuk menambah kategori barang. Data yang harus dicatat pada penambahan kategori barang adalah kode, nama, dan deskripsi.
Kondisi awal	Administrator telah menjalankan *use case* Melihat daftar kategori barang.
Kondisi akhir	Kategori barang telah ditambahkan ke dalam sistem.
Aliran Utama	
Aksi dari Aktor	**Tanggapan dari Sistem**
1. Administrator memilih link “Tambah Kategori”.	2. Sistem menampilkan *form* masukkan yang terdiri dari kode, nama, dan deskripsi.
3. Administrator memasukkan kode, nama, dan deskripsi.	4. Sistem memvalidasi masukkan kode dan nama tidak boleh kosong.
	5. Sistem menampilkan pesan bahwa data berhasil ditambahkan.
Aliran Alternatif 1: Masukkan kode atau nama kosong. (Administrator pada langkah nomer 3 aliran utama tidak memasukkan kode atau nama)	
Aksi dari Aktor	**Tanggapan dari Sistem**
	1. Sistem menampilkan pesan kesalahan bahwa kode atau nama tidak boleh kosong.
	2. Sistem kembali pada langkah nomer 2 aliran utama.
Aliran Alternatif 2: Kode telah ada dalam sistem (Administrator pada langkah nomer 3 aliran utama memasukkan kode yang telah ada dalam sistem)	
Aksi dari Aktor	**Tanggapan dari Sistem**
	1. Sistem menampilkan pesan kesalahan bahwa kode tersebut telah ada dalam sistem.
	2. Sistem kembali pada langkah nomer 2 aliran utama.
Aturan khusus 1. Masukkan kode dan nama tidak boleh kosong. 2. Masukkan kode belum ada dalam sistem.	

Sumber: [Perancangan]

11. *Use Case Spesification* Mengubah Data Kategori Barang

Tabel 4. 23 *Use case specification* Mengubah data kategori barang

Nama *use case*	Mengubah data kategori barang
Aktor	Administrator
Deskripsi	Sistem harus menyediakan fasilitas untuk mengubah data kategori suatu barang. Data yang dapat diubah adalah kode, nama, dan deskripsi.
Kondisi awal	Administrator telah menjalankan *use case* Melihat daftar kategori barang.
Kondisi akhir	Data kategori barang yang telah diubah disimpan ke dalam sistem.
Aliran Utama	
Aksi dari Aktor	**Tanggapan dari Sistem**
1. Administrator menekan tombol "Ubah" salah satu baris dari data kategori barang yang ditampilkan dari *use case* Melihat daftar kategori barang.	2. Sistem menampilkan kode, nama, dan deskripsi dari baris kategori barang yang telah dipilih administrator.
3. Administrator mengubah data untuk kode, nama, dan deskripsi.	4. Sistem memvalidasi masukkan kode dan nama tidak boleh kosong.
	5. Sistem menampilkan pesan bahwa data berhasil diubah.
Aliran Alternatif 1: Masukkan kode atau nama kosong. (Administrator pada langkah nomer 3 aliran utama tidak memasukkan kode atau nama)	
Aksi dari Aktor	**Tanggapan dari Sistem**
	1. Sistem menampilkan pesan kesalahan bahwa kode atau nama tidak boleh kosong.
	2. Sistem kembali pada langkah nomer 2 aliran utama.
Aliran Alternatif 2: kode telah ada dalam sistem (Administrator pada langkah nomer 3 aliran utama memasukkan kode yang telah ada dalam sistem)	
Aksi dari Aktor	**Tanggapan dari Sistem**
	1. Sistem menampilkan pesan kesalahan bahwa kode tersebut telah ada dalam sistem.
	2. Sistem kembali pada langkah nomer 2 aliran utama.
Aturan khusus 1. Masukkan kode dan nama tidak boleh kosong. 2. Masukkan kode belum ada dalam sistem.	

Sumber: [Perancangan]

12. *Use Case Spesification* Menghapus Kategori Barang

Tabel 4. 24 *Use case specification* Menghapus kategori barang

Nama *use case*	Menghapus kategori barang
Aktor	Administrator
Deskripsi	Sistem harus menyediakan fasilitas untuk menghapus kategori barang tertentu.
Kondisi awal	Administrator telah menjalankan *use case* Melihat daftar kategori barang.
Kondisi akhir	Kategori barang telah dihapus dari dalam sistem
Aliran Utama	
Aksi dari Aktor	**Tanggapan dari Sistem**
1. Administrator memilih menu "Hapus" salah satu baris dari daftar kategori barang yang ditampilkan dari *use case* Melihat daftar kategori barang.	2. Sistem menampilkan pesan bahwa data berhasil dihapus.

Sumber: [Perancangan]

13. *Use Case Spesification* Mencari Kategori Barang

Tabel 4. 25 *Use case specification* Mencari kategori barang

Nama *use case*	Mencari kategori barang
Aktor	Administrator

Deskripsi	Sistem harus menyediakan fasilitas untuk mencari suatu kategori barang. Pencarian kategori barang berdasarkan nama kategori barang.
Kondisi awal	Administrator telah menjalankan *use case* Melihat daftar kategori barang.
Kondisi akhir	Sistem menampilkan kategori barang yang dicari oleh administrator.

Aliran Utama	
Aksi dari Aktor	**Tanggapan dari Sistem**
1. Administrator memilih menu "Kategori".	2. Sistem menampilkan semua daftar kategori barang yang ada di dalam sistem dan *form* pencarian.
3. Administrator memasukkan nama kategori sebagai *keyword* dan menekan tombol "Cari".	4. Sistem menampilkan kategori barang yang dicari.
Aliran Alternatif 1: Masukkan *keyword* kosong (Administrator pada langkah nomer 3 aliran utama tidak memasukkan *keyword*)	
Aksi dari Aktor	**Tanggapan dari Sistem**
	1. Sistem kembali ke langkah nomer 2 aliran utama.
Aturan khusus	
1. Masukkan *keyword* tidak boleh kosong.	

Sumber: [Perancangan]

14. *Use Case Spesification* Melihat Daftar Subkategori Barang

Tabel 4. 26 *Use case specification* Melihat daftar subkategori barang

Nama *use case*	Melihat daftar subkategori barang
Aktor	Administrator
Deskripsi	Sistem harus menyediakan fasilitas untuk melihat daftar subkategori per kategori barang yang ada pada sistem. Data yang dapat dilihat adalah kode dan nama.
Kondisi awal	Administrator telah menjalankan *use case* Melihat data kategori barang.
Kondisi akhir	Sistem menampilkan daftar subkategori barang.

Aliran Utama	
Aksi dari Aktor	**Tanggapan dari Sistem**
1. Administrator memilih salah satu kategori dari daftar "Kategori".	2. Sistem menampilkan daftar subkategori barang per kategori yang ada di dalam sistem. Data yang ditampilkan adalah kode dan nama.

Sumber: [Perancangan]

15. *Use Case Spesification* Menambah Subkategori Barang

Tabel 4. 27 *Use case specification* Menambah subkategori barang

Nama *use case*	Menambah subkategori barang
Aktor	Administrator
Deskripsi	Sistem harus menyediakan fasilitas untuk menambah subkategori barang. Data yang harus dicatat pada penambahan subkategori barang adalah kode dan nama.
Kondisi awal	Administrator telah menjalankan *use case* Melihat data kategori barang.
Kondisi akhir	Subkategori barang telah ditambahkan ke dalam sistem.

Aliran Utama	
Aksi dari Aktor	**Tanggapan dari Sistem**
1. Administrator memilih link "Tambah Subkategori".	2. Sistem menampilkan *form* masukkan yang terdiri dari kode dan nama.
3. Administrator memasukkan kode dan nama.	4. Sistem memvalidasi masukkan kode dan nama tidak boleh kosong.
	5. Sistem menampilkan pesan bahwa data berhasil ditambahkan.
Aliran Alternatif 1: Masukkan kode atau nama kosong. (Administrator pada langkah nomer 3 aliran utama tidak memasukkan kode atau nama)	
Aksi dari Aktor	**Tanggapan dari Sistem**
	1. Sistem menampilkan pesan kesalahan bahwa

	kode atau nama tidak boleh kosong.
	2. Sistem kembali pada langkah nomer 2 aliran utama.
Aliran Alternatif 2: Kode telah ada dalam sistem (Administrator pada langkah nomer 3 aliran utama memasukkan kode yang telah ada dalam sistem)	
Aksi dari Aktor	**Tanggapan dari Sistem**
	1. Sistem menampilkan pesan kesalahan bahwa kode tersebut telah ada dalam sistem.
	2. Sistem kembali pada langkah nomer 2 aliran utama.
Aturan khusus 1. Masukkan kode dan nama tidak boleh kosong. 2. Masukkan kode belum ada dalam sistem.	

Sumber: [Perancangan]

16. *Use Case Spesification* Mengubah Data Subkategori Barang

Tabel 4. 28 *Use case specification* Mengubah data subkategori barang

Nama *use case*	Mengubah data subkategori barang
Aktor	Administrator
Deskripsi	Sistem harus menyediakan fasilitas untuk mengubah data subkategori suatu barang. Data yang dapat diubah adalah kode dan nama.
Kondisi awal	Administrator telah menjalankan *use case* Melihat data kategori barang.
Kondisi akhir	Data subkategori barang yang telah diubah disimpan ke dalam sistem.
Aliran Utama	
Aksi dari Aktor	**Tanggapan dari Sistem**
1. Administrator menekan tombol "Ubah" salah satu baris dari daftar subkategori barang yang ditampilkan dari *use case* Melihat data kategori barang.	2. Sistem menampilkan kode dan nama dari baris subkategori barang yang telah dipilih administrator.
3. Administrator mengubah data untuk kode dan nama.	4. Sistem memvalidasi masukkan kode dan nama tidak boleh kosong.
	5. Sistem menampilkan pesan bahwa data berhasil diubah.
Aliran Alternatif 1: Masukkan kode atau nama kosong. (Administrator pada langkah nomer 3 aliran utama tidak memasukkan kode atau nama)	
Aksi dari Aktor	**Tanggapan dari Sistem**
	1. Sistem menampilkan pesan kesalahan bahwa kode atau nama tidak boleh kosong.
	2. Sistem kembali pada langkah nomer 2 aliran utama.
Aliran Alternatif 2: kode telah ada dalam sistem (Administrator pada langkah nomer 3 aliran utama memasukkan kode yang telah ada dalam sistem)	
Aksi dari Aktor	**Tanggapan dari Sistem**
	1. Sistem menampilkan pesan kesalahan bahwa kode tersebut telah ada dalam sistem.
	2. Sistem kembali pada langkah nomer 2 aliran utama.
Aturan khusus 1. Masukkan kode dan nama tidak boleh kosong. 2. Masukkan kode belum ada dalam sistem.	

Sumber: [Perancangan]

17. *Use Case Spesification* Menghapus Subkategori Barang

Tabel 4. 29 *Use case specification* Menghapus subkategori barang

Nama *use case*	Menghapus subkategori barang
Aktor	Administrator
Deskripsi	Sistem harus menyediakan fasilitas untuk menghapus subkategori barang

	tertentu.
Kondisi awal	Administrator telah menjalankan *use case* Melihat data kategori barang.
Kondisi akhir	Subkategori barang telah dihapus dari dalam sistem
Aliran Utama	
Aksi dari Aktor	**Tanggapan dari Sistem**
1. Administrator memilih menu "Hapus" salah satu baris dari daftar subkategori barang yang ditampilkan dari *use case* Melihat data kategori barang.	2. Sistem menampilkan pesan bahwa data berhasil dihapus.

Sumber: [Perancangan]

18. *Use Case Spesification* Melihat Daftar Suplier

Tabel 4. 30 *Use case specification* Melihat daftar suplier

Nama *use case*	Melihat daftar suplier
Aktor	Administrator
Deskripsi	Sistem harus menyediakan fasilitas untuk melihat daftar suplier yang ada di dalam sistem. Data yang dapat dilihat adalah kode, nama, dan no. telp.
Kondisi awal	Sistem telah berjalan dan user telah *login* sebagai administrator.
Kondisi akhir	Sistem menampilkan daftar suplier.
Aliran Utama	
Aksi dari Aktor	**Tanggapan dari Sistem**
1. Administrator memilih menu "Suplier".	2. Sistem menampilkan daftar suplier yang ada di dalam sistem. Data yang ditampilkan adalah kode, nama, dan no. telp.

Sumber: [Perancangan]

19. *Use Case Spesification* Melihat Data Suplier

Tabel 4. 31 *Use case specification* Melihat data suplier

Nama *use case*	Melihat data suplier
Aktor	Administrator
Deskripsi	Sistem harus menyediakan fasilitas untuk melihat data suplier tertentu yang ada di dalam sistem.
Kondisi awal	Administrator telah menjalankan *use case* Melihat daftar suplier.
Kondisi akhir	Sistem menampilkan data suplier tertentu.
Aliran Utama	
Aksi dari Aktor	**Tanggapan dari Sistem**
1. Administrator memilih 'kode' salah satu baris dari daftar suplier yang ditampilkan dari *use case* Melihat daftar suplier.	2. Sistem menampilkan data suplier tertentu yang ada di dalam sistem. Data yang ditampilkan adalah kode, nama, alamat, dan no. telp.

Sumber: [Perancangan]

20. *Use Case Spesification* Menambah Suplier

Tabel 4. 32 *Use case specification* Menambah suplier

Nama *use case*	Menambah suplier
Aktor	Administrator
Deskripsi	Sistem harus menyediakan fasilitas untuk menambah suplier. Data yang harus dicatat adalah kode, nama, alamat, dan no. telp.
Kondisi awal	Administrator telah menjalankan *use case* Melihat daftar suplier.
Kondisi akhir	Suplier telah ditambahkan ke dalam sistem.
Aliran Utama	
Aksi dari Aktor	**Tanggapan dari Sistem**
1. Administrator memilih menu "Tambah Suplier".	2. Sistem menampilkan *form* masukkan yang terdiri dari kode, nama, alamat, dan no. telp.
3. Administrator memasukkan kode, nama, alamat, dan no. telp.	4. Sistem memvalidasi masukkan. Kode, nama, alamat, dan no. telp tidak boleh kosong.

	5. Sistem menampilkan pesan bahwa data berhasil ditambahkan.
Aliran Alternatif 1: Masukkan kode, nama, alamat, atau no. telp kosong. (Administrator pada langkah nomer 3 aliran utama tidak memasukkan kode, nama, alamat, atau no. telp)	
Aksi dari Aktor	**Tanggapan dari Sistem**
	1. Sistem menampilkan pesan kesalahan bahwa kode, nama, alamat, atau no. telp tidak boleh kosong.
	2. Sistem kembali pada langkah nomer 2 aliran utama.
Aliran Alternatif 2: kode telah ada dalam sistem (Administrator pada langkah nomer 3 aliran utama memasukkan kode yang telah ada dalam sistem)	
Aksi dari Aktor	**Tanggapan dari Sistem**
	1. Sistem menampilkan pesan kesalahan bahwa kode tersebut telah ada dalam sistem.
	2. Sistem kembali pada langkah nomer 2 aliran utama.
Aturan khusus 1. Masukkan kode, nama, alamat, dan no. telp tidak boleh kosong. 2. Masukkan kode belum ada dalam sistem.	

Sumber: [Perancangan]

21. *Use Case Spesification* Mengubah Data Suplier

Tabel 4. 33 *Use case specification* Mengubah data suplier

Nama *use case*	Mengubah data suplier
Aktor	Administrator
Deskripsi	Sistem harus menyediakan fasilitas untuk mengubah data suplier tertentu. Data yang dapat diubah adalah kode, nama, alamat, dan no. telp.
Kondisi awal	Administrator telah menjalankan *use case* Melihat daftar suplier.
Kondisi akhir	Data suplier yang telah diubah disimpan ke dalam sistem.
Aliran Utama	
Aksi dari Aktor	**Tanggapan dari Sistem**
1. Administrator memilih menu "Ubah" salah satu baris dari daftar suplier yang ditampilkan dari *use case* Melihat daftar suplier.	2. Sistem menampilkan kode, nama, alamat, dan no. telp dari baris suplier yang telah dipilih administrator.
3. Administrator mengubah data untuk kode, nama, alamat, dan no. telp.	4. Sistem memvalidasi masukkan. Kode, nama, alamat, dan no. telp tidak boleh kosong.
	5. Sistem menampilkan pesan bahwa data suplier berhasil diubah.
Aliran Alternatif 1: Masukkan kode, nama, alamat, atau no. telp kosong. (Administrator pada langkah nomer 3 aliran utama tidak memasukkan kode, nama, alamat, atau no. telp)	
Aksi dari Aktor	**Tanggapan dari Sistem**
	1. Sistem menampilkan pesan kesalahan bahwa kode, nama, alamat, atau no. telp tidak boleh kosong.
	2. Sistem kembali pada langkah nomer 2 aliran utama.
Aliran Alternatif 2: kode telah ada dalam sistem (Administrator pada langkah nomer 3 aliran utama memasukkan kode yang telah ada dalam sistem)	
Aksi dari Aktor	**Tanggapan dari Sistem**
	1. Sistem menampilkan pesan kesalahan bahwa kode tersebut telah ada dalam sistem.
	2. Sistem kembali pada langkah nomer 2 aliran utama.
Aturan khusus 1. Masukkan kode, nama, alamat, atau no. telp tidak boleh kosong. 2. Masukkan kode belum ada dalam sistem.	

Sumber: [Perancangan]

22. *Use Case Spesification* Menghapus Suplier

Tabel 4. 34 *Use case specification* Menghapus suplier

Nama *use case*	Menghapus suplier
Aktor	Administrator
Deskripsi	Sistem harus menyediakan fasilitas untuk menghapus suplier tertentu.
Kondisi awal	Administrator telah menjalankan *use case* Melihat daftar suplier.
Kondisi akhir	Suplier telah dihapus dari dalam sistem.
Aliran Utama	
Aksi dari Aktor	**Tanggapan dari Sistem**
1. Administrator memilih menu "Hapus" salah satu baris dari daftar suplier yang ditampilkan dari *use case* Melihat daftar suplier.	2. Sistem menampilkan pesan bahwa data berhasil dihapus.

Sumber: [Perancangan]

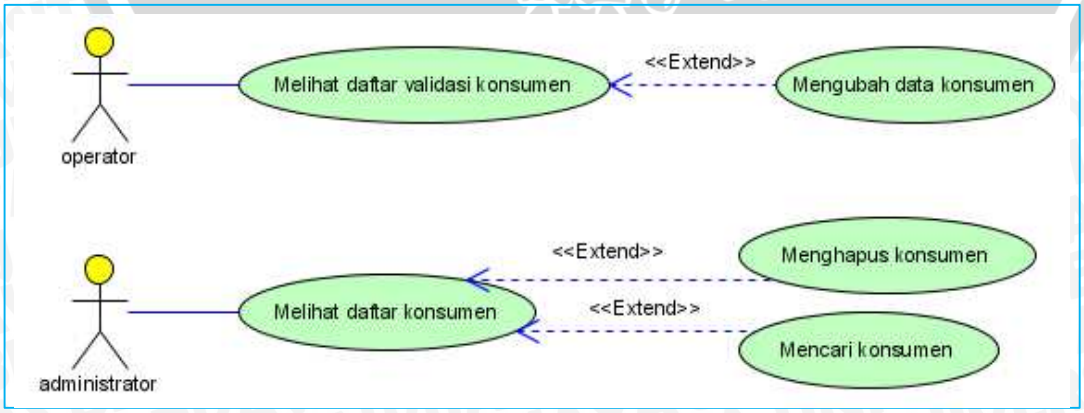
23. *Use Case Spesification* Mencari Suplier

Tabel 4. 35 *Use case specification* Mencari suplier

Nama *use case*	Mencari suplier
Aktor	Administrator
Deskripsi	Sistem harus menyediakan fasilitas untuk mencari suplier tertentu. Pencarian suplier berdasarkan nama suplier.
Kondisi awal	User telah *login* sebagai administrator.
Kondisi akhir	Sistem menampilkan data suplier yang dicari oleh administrator.
Aliran Utama	
Aksi dari Aktor	**Tanggapan dari Sistem**
1. Administrator memilih menu "Suplier".	2. Sistem menampilkan semua daftar suplier yang ada di dalam sistem dan *form* pencarian.
3. Administrator memasukkan nama suplier sebagai *keyword* dan menekan tombol "Cari".	4. Sistem menampilkan suplier yang dicari.
Aliran Alternatif 1: Masukkan *keyword* kosong (Administrator pada langkah nomer 3 aliran utama tidak memasukkan *keyword*)	
Aksi dari Aktor	**Tanggapan dari Sistem**
	1. Sistem kembali ke langkah nomer 2 aliran utama.
Aturan khusus	
1. Masukkan *keyword* tidak boleh kosong.	

Sumber: [Perancangan]

4.1.2.2.3. *Use Case Diagram* untuk Modul Konsumen



Gambar 4. 7 *Use case diagram* untuk modul konsumen *mobile online supermarket*
Sumber: [Perancangan]

Gambar 4.7 menunjukkan *use case diagram* untuk modul konsumen *mobile online supermarket*. Diagram *use case* ini melibatkan 2 aktor dan 5 *use case*. Kelima *use case* yang termasuk dalam modul konsumen *mobile online supermarket* dapat dilihat pada Tabel 4.3. Kedua aktor yang terlibat dalam modul konsumen *mobile online supermarket* adalah operator dan administrator.

1. *Use Case Spesification* Melihat Daftar Validasi Konsumen

Tabel 4. 36 *Use case specification* Melihat daftar validasi konsumen

Nama *use case*	Melihat daftar validasi konsumen
Aktor	Operator
Deskripsi	Sistem harus menyediakan fasilitas untuk melihat daftar validasi konsumen yang telah registrasi. Data yang dapat dilihat adalah nama, *no. handphone*, dan status validasi.
Kondisi awal	Sistem telah berjalan dan user telah melakukan *login* sebagai operator.
Kondisi akhir	Status validasi konsumen akan disimpan pada sistem.
Aliran Utama	
Aksi dari Aktor	**Tanggapan dari Sistem**
1. Operator memilih menu "Validasi Konsumen".	2. Sistem menampilkan daftar konsumen yang baru tercatat di dalam sistem. Data yang ditampilkan adalah nama, *no. handphone*, dan status validasi.

Sumber: [Perancangan]

2. *Use Case Spesification* Mengubah Data Konsumen

Tabel 4. 37 *Use case specification* Mengubah data konsumen

Nama *use case*	Mengubah data konsumen
Aktor	Operator
Deskripsi	Sistem harus menyediakan fasilitas untuk mengubah data konsumen tertentu. Data yang dapat diubah adalah status validasi.
Kondisi awal	Administrator telah menjalankan *use case* Melihat daftar konsumen.
Kondisi akhir	Data konsumen yang telah diubah disimpan ke dalam sistem.
Aliran Utama	
Aksi dari Aktor	**Tanggapan dari Sistem**
1. Operator memilih menu "Status Validasi" salah satu baris dari daftar konsumen yang ditampilkan dari *use case* Melihat daftar konsumen.	2. Sistem menampilkan *username*, nama, alamat, no. handphone, email, dan status validasi yang telah dipilih operator.
3. Operator mengubah data status validasi.	4. Sistem memvalidasi data dan status validasi tidak boleh kosong.
	5. Sistem menampilkan pesan bahwa data berhasil diubah.

Sumber: [Perancangan]

3. *Use Case Spesification* Melihat Daftar Konsumen

Tabel 4. 38 *Use case specification* Melihat daftar konsumen

Nama *use case*	Melihat daftar konsumen
Aktor	Administrator
Deskripsi	Sistem harus menyediakan fasilitas untuk melihat daftar konsumen yang ada di dalam sistem. Data yang dapat dilihat adalah nama dan status validasi.
Kondisi awal	Sistem telah berjalan dan user telah *login* sebagai administrator.
Kondisi akhir	Sistem menampilkan daftar konsumen.
Aliran Utama	

Aksi dari Aktor	Tanggapan dari Sistem
1. Administrator memilih menu "Konsumen".	2. Sistem menampilkan daftar konsumen yang ada di dalam sistem. Data yang ditampilkan adalah nama dan status validasi.

Sumber: [Perancangan]

4. *Use Case Spesification* Menghapus Konsumen

Tabel 4. 39 *Use case specification* Menghapus konsumen

Nama *use case*	Menghapus konsumen
Aktor	Administrator
Deskripsi	Sistem harus menyediakan fasilitas untuk menghapus konsumen tertentu.
Kondisi awal	Administrator telah menjalankan *use case* Melihat daftar konsumen.
Kondisi akhir	Konsumen telah dihapus dari dalam sistem.
Aliran Utama	
Aksi dari Aktor	**Tanggapan dari Sistem**
1. Administrator memilih menu "Hapus" salah satu baris dari daftar konsumen yang ditampilkan dari *use case* Melihat daftar konsumen.	2. Sistem menampilkan pesan bahwa data berhasil dihapus.

Sumber: [Perancangan]

5. *Use Case Spesification* Mencari Konsumen

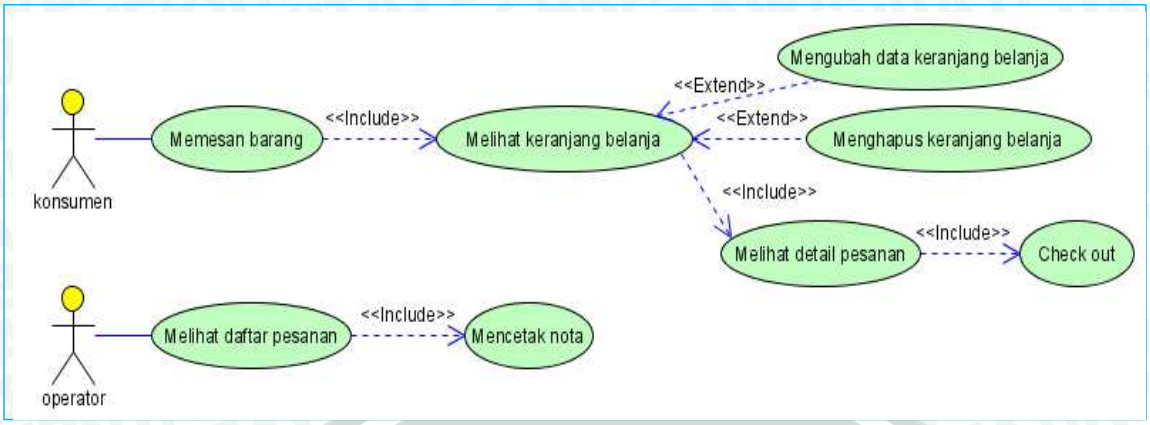
Tabel 4. 40 *Use case specification* Mencari konsumen

Nama *use case*	Mencari konsumen
Aktor	Administrator
Deskripsi	Sistem harus menyediakan fasilitas untuk mencari konsumen. Pencarian konsumen berdasarkan status validasi.
Kondisi awal	User telah *login* sebagai administrator.
Kondisi akhir	Sistem menampilkan daftar konsumen yang dicari oleh administrator.
Aliran Utama	
Aksi dari Aktor	**Tanggapan dari Sistem**
1. Administrator memilih menu "Konsumen".	2. Sistem menampilkan semua daftar konsumen yang ada di dalam sistem dan *form* pencarian.
3. Administrator memasukkan status validasi sebagai *keyword* dan menekan tombol "Cari".	4. Sistem menampilkan daftar konsumen yang dicari.
Aliran Alternatif 1: Masukkan *keyword* kosong (Administrator pada langkah nomer 3 aliran utama tidak memasukkan *keyword*)	
Aksi dari Aktor	**Tanggapan dari Sistem**
	1. Sistem kembali ke langkah nomer 2 aliran utama.
Aturan khusus	
1. Masukkan *keyword* tidak boleh kosong.	

Sumber: [Perancangan]

4.1.2.2.4. *Use Case Diagram* untuk Modul Pendukung Pemesanan Barang

Use case diagram untuk modul pendukung pemesanan barang *mobile online supermarket* ditunjukkan pada Gambar 4.8. Diagram *use case* ini melibatkan 2 aktor dan 8 *use case*. Kedelapan *use case* yang termasuk dalam modul pendukung pemesanan barang *mobile online supermarket* dapat dilihat pada Tabel 4.3. Kedua aktor yang terlibat dalam modul pendukung pemesanan barang *mobile online supermarket* adalah konsumen dan operator.



Gambar 4. 8 *Use case diagram* untuk modul pendukung pemesanan barang *mobile online supermarket*
Sumber: [Perancangan]

1. *Use Case Spesification* Memesan Barang

Tabel 4. 41 *Use case specification* Memesan barang

Nama *use case*	Memesan barang
Aktor	Konsumen
Deskripsi	Sistem harus menyediakan fasilitas untuk melakukan pemesanan barang.
Kondisi awal	Sistem telah berjalan dan user telah melakukan *login* sebagai konsumen.
Kondisi akhir	Sistem menjalankan *use case* Melihat keranjang belanja.

Aliran Utama	
Aksi dari Aktor	**Tanggapan dari Sistem**
1. Konsumen memilih menu "Kategori Barang" yang diinginkan.	2. Sistem menampilkan daftar kategori barang.
3. Konsumen memilih kategori barang dari daftar yang ditampilkan.	4. Sistem menampilkan daftar barang per kategori.
5. Konsumen memasukkan angka yang sesuai pada *form input* jumlah barang dan menekan tombol "Pesan" pada barang yang diinginkan.	6. Sistem menyimpan data barang yang dipesan dalam keranjang belanja.

Sumber: [Perancangan]

2. *Use Case Spesification* Melihat Keranjang Belanja

Tabel 4. 42 *Use case specification* Melihat keranjang belanja

Nama *use case*	Melihat keranjang belanja
Aktor	Konsumen
Deskripsi	Sistem harus menyediakan fasilitas untuk melihat data barang yang dipesan di keranjang belanja. Data yang dapat dilihat adalah nama barang, harga, jumlah, sub total, dan total belanja.
Kondisi awal	Konsumen telah menjalankan *use case* Memesan barang.
Kondisi akhir	Sistem menampilkan daftar barang dalam keranjang belanja.

Aliran Utama	
Aksi dari Aktor	**Tanggapan dari Sistem**
1. Konsumen memilih menu "Keranjang Belanja"	2. Sistem menampilkan daftar belanja yang terdiri dari nama barang, harga, jumlah, sub total, dan total belanja.

Sumber: [Perancangan]

3. *Use Case Spesification* Mengubah Data Keranjang Belanja

Tabel 4. 43 *Use case specification* Mengubah data keranjang belanja

Nama *use case*	Mengubah data keranjang belanja
Aktor	Konsumen
Deskripsi	Sistem harus menyediakan fasilitas untuk mengubah data barang yang

	dipesan di keranjang belanja. Data yang dapat diubah adalah jumlah.
Kondisi awal	Konsumen telah menjalankan *use case* Melihat keranjang belanja.
Kondisi akhir	Jumlah barang yang telah diubah disimpan dalam keranjang belanja.
Aliran Utama	
Aksi dari Aktor	**Tanggapan dari Sistem**
1. Konsumen memasukkan angka yang sesuai pada *form input* jumlah barang.	2. Sistem menyimpan data yang diubah.

Sumber: [Perancangan]

4. *Use Case Spesification* Menghapus Keranjang Belanja

Tabel 4. 44 *Use case specification* Menghapus keranjang belanja

Nama *use case*	Menghapus keranjang belanja
Aktor	Konsumen
Deskripsi	Sistem harus menyediakan fasilitas untuk menghapus barang tertentu di keranjang belanja.
Kondisi awal	Konsumen telah menjalankan *use case* Melihat keranjang belanja.
Kondisi akhir	Data barang telah dihapus dari dalam keranjang belanja.
Aliran Utama	
Aksi dari Aktor	**Tanggapan dari Sistem**
1. Konsumen memilih menu “Hapus” salah satu baris dari daftar barang yang ditampilkan dari *use case* Melihat keranjang belanja.	2. Sistem menampilkan pesan bahwa data berhasil dihapus.

Sumber: [Perancangan]

5. *Use Case Specification* Melihat Detail Pesanan

Tabel 4. 45 *Use case specification* Melihat detail pesanan

Nama *use case*	Melihat detail pesanan
Aktor	Konsumen
Deskripsi	Sistem harus menyediakan fasilitas untuk melihat data dan detail barang yang dipesan di keranjang belanja. Data yang dapat dilihat adalah nama konsumen, alamat, *no. handphone*, nama barang, harga, jumlah, sub total, dan total belanja.
Kondisi awal	Konsumen telah menjalankan *use case* Melihat keranjang belanja.
Kondisi akhir	Sistem menampilkan data dan detail barang yang dipesan.
Aliran Utama	
Aksi dari Aktor	**Tanggapan dari Sistem**
1. Konsumen memilih menu “Detail Pesanan”	2. Sistem menampilkan data nama konsumen, alamat, *no. handphone*, dan detail belanja yang terdiri dari nama barang, harga, jumlah, sub total, dan total belanja.

Sumber: [Perancangan]

6. *Use Case Spesification* Check Out

Tabel 4. 46 *Use case specification* Check out

Nama *use case*	Check out
Aktor	Konsumen
Deskripsi	Sistem harus menyediakan fasilitas untuk menampilkan data transaksi.
Kondisi awal	Konsumen telah menjalankan *use case* Melihat detail pesanan.
Kondisi akhir	Sistem menampilkan data transaksi
Aliran Utama	
Aksi dari Aktor	**Tanggapan dari Sistem**
1. Konsumen menekan tombol “Check Out”.	2. Sistem menampilkan data transaksi yang terdiri dari kode transaksi, waktu pesan, nama, alamat, *no. handphone*, total belanja, dan nama operator.

Sumber: [Perancangan]

7. *Use Case Spesification* Melihat Daftar Pesanan

Tabel 4. 47 *Use case specification* Melihat daftar pesanan

Nama *use case*	Melihat daftar pesanan
Aktor	Operator
Deskripsi	Sistem harus menyediakan fasilitas untuk melihat daftar pesanan konsumen. Data yang dapat dilihat adalah kode transaksi, waktu pesan, dan status kirim.
Kondisi awal	User telah *login* sebagai operator.
Kondisi akhir	Sistem menampilkan daftar pesanan konsumen.
Aliran Utama	
Aksi dari Aktor	**Tanggapan dari Sistem**
1. Operator memilih menu "Daftar Pesanan".	2. Sistem menampilkan daftar pesanan yang terdiri dari kode transaksi, waktu pesan, dan status kirim.

Sumber: [Perancangan]

8. *Use Case Spesification* Mencetak Nota

Tabel 4. 48 *Use case specification* Mencetak nota

Nama *use case*	Mencetak nota
Aktor	Operator
Deskripsi	Sistem harus menyediakan fasilitas untuk mencetak nota sebagai bukti pemesanan.
Kondisi awal	User telah *login* sebagai operator dan telah menjalankan *use case* Melihat daftar pesanan.
Kondisi akhir	Sistem menampilkan informasi nota.
Aliran Utama	
Aksi dari Aktor	**Tanggapan dari Sistem**
1. Operator menekan tombol "Cetak Nota".	2. Sistem akan mencetak nota.

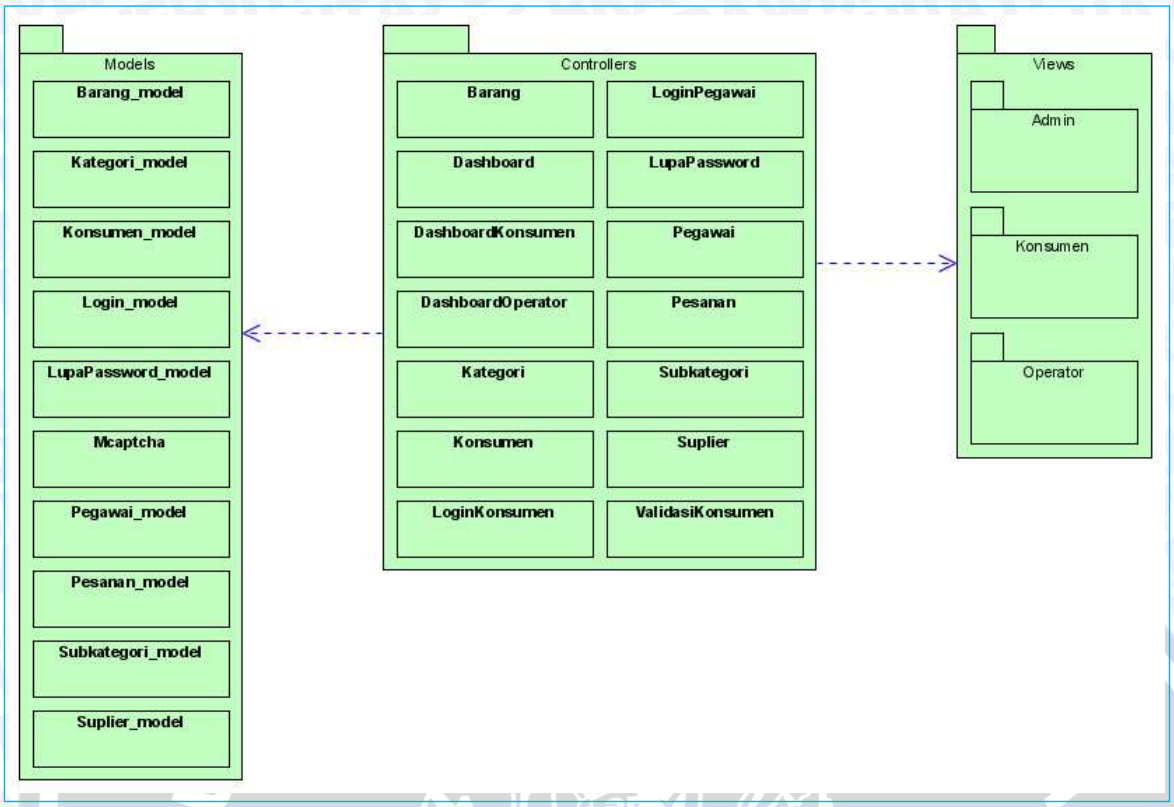
Sumber: [Perancangan]

4.2. Perancangan Perangkat Lunak

Use case diagram dan *use case spesification* yang telah dibuat pada tahap analisis kebutuhan memberikan sebuah pandangan terhadap kebutuhan fungsional dari user yang harus disediakan oleh sistem perangkat lunak.

Selain *use case diagram* dan *use case specification*, perancangan pada tugas akhir ini akan dimodelkan dengan beberapa diagram yaitu *class diagram, sequence diagram,* dan *state diagram*. Perancangan umum dari aplikasi *mobile online supermarket* ini menggambarkan relasi antar *class* pada tiga *package* yang berbeda. Ketiga *package* itu adalah *models* yang berisi fungsi-fungsi yang membantu mencari, memasukkan, dan memperbarui informasi dalam *database*, *views* adalah informasi yang ditampilkan kepada *user*, dan *controllers* sebagai perantara yang menghubungkan antara *models*, *views*, dan program lainnya yang dibutuhkan untuk membangun sebuah halaman *web*.

Ditunjukkan pada Gambar 4.9 ketiga *package* saling terhubung, dimana *package class controllers* tergantung pada *package class models* dan tergantung pada *package views* yang mempunyai *sub package* admin, konsumen, dan operator yang masing-masing didalamnya terdapat *file-file* .php sebagai tampilan halaman *web*.



Gambar 4. 9 Relasi antar *package diagram*
Sumber: [Perancangan]

Tabel 4. 49 Daftar *file-file* .php dari *package views*

Package Views		
***Package* Admin**	***Package* Konsumen**	***Package* Operator**
adminHome.php	barangKonsumen.php	cariGagal.php
adminUbah.php	barangKonsumenCari.php	cetak.php
barangCari.php	barangUser.php	loginPegawai.php
barangDaftar.php	barangUserCari.php	lupaPassword.php
barangLihat.php	cariGagalKonsumen.php	nota.php
barangTambah.php	cariGagalUser.php	operatorHome.php
barangUbah.php	checkOut.php	operatorUbah.php
cariGagal.php	detailBarangKonsumen.php	pesananCari.php
emailSama.php	detailBarangUser.php	pesananDaftar.php
kategoriCari.php	detailPesanan.php	pesananKodeCari.php
kategoriDaftar.php	emailSama.php	pesananLihat.php
kategoriLihat.php	emailSamaReg.php	suksesEmail.php
kategoriTambah.php	kategoriKonsumen.php	suksesProses.php
kategoriUbah.php	kategoriUser.php	suksesUbah.php
kodeSama.php	keranjangBelanja.php	suksesUbahDataku.php
konsumenCari.php	konsumenHome.php	userSama.php
konsumenDaftar.php	loginKonsumen.php	validasiKonsumenCari.php
konsumenLihat.php	lupaPassword.php	validasiKonsumenDaftar.php
lihatBarang.php	registrasi.php	validasiKonsumenUbah.php
pegawaiCari.php	salahChaptcha.php	
pegawaiDaftar.php	stokHabis.php	
pegawaiLihat.php	subkategoriKonsumen.php	
pegawaiTambah.php	subkategoriUser.php	
pegawaiUbah.php	suksesEmail.php	
stokTambah.php	suksesHapus.php	
subkategoriTambah.php	suksesPesan.php	
subkategoriUbah.php	suksesRegistrasi.php	

suksesHapus.php suksesTambah.php suksesUbah.php suksesUbahDataku.php suplierCari.php suplierDaftar.php suplierLihat.php suplierTambah.php suplierUbah.php userSama.php	suksesUbah.php suksesUbahUserPass.php ubahDataku.php ubahUserPass.php userSama.php userSamaReg.php	

Sumber: [Perancangan]

4.2.1. *Class Diagram*

Class diagram menggambarkan struktur dan deskripsi *class*, *package* dan objek beserta hubungan satu sama lain seperti pewarisan, asosiasi, dan lain-lain. [DHA-03]

Gambar 4. 10 *Class-class diagram* dari *models*
Sumber: [Perancangan]

Gambar 4. 11 *Class-class diagram* dari *controller*
Sumber: [Perancangan]

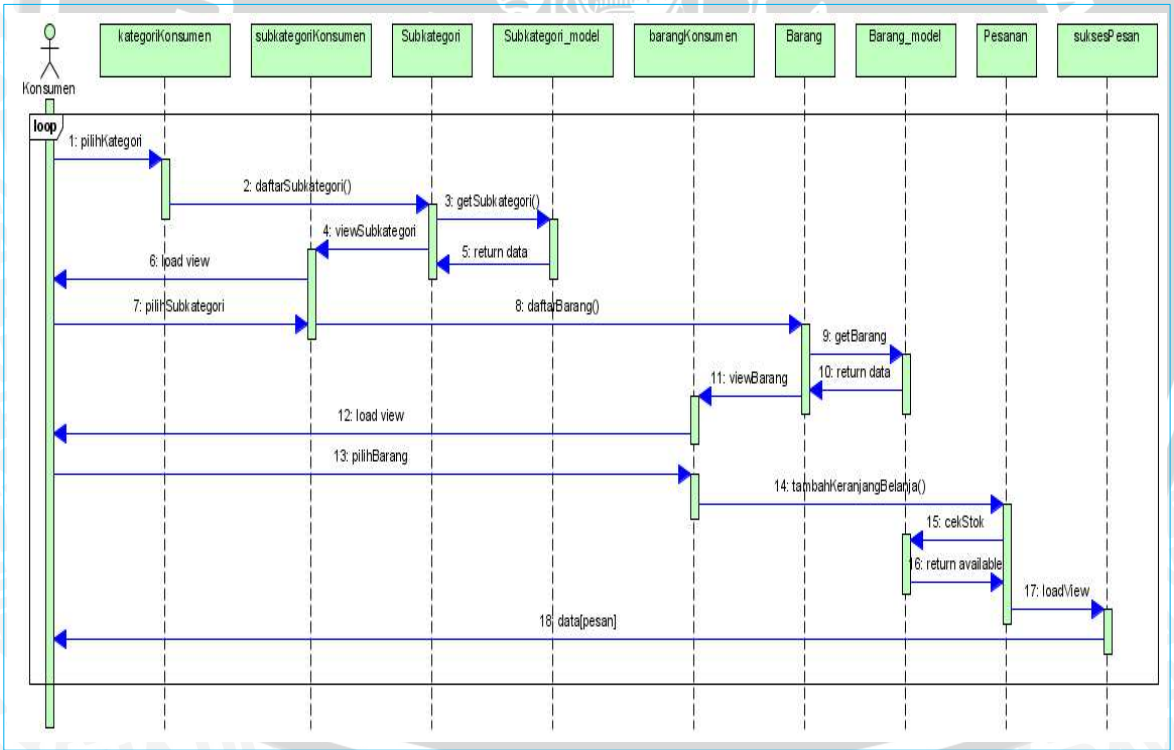
4.2.2. *Sequence Diagram*

Sequence diagram digunakan untuk mengetahui perilaku beberapa obyek pada *use case*. *Sequence diagram* menambahkan dimensi waktu pada interaksi diantara obyek. Pada sub bab 4.2.2. ini tidak memodelkan keseluruhan *sequence diagram* untuk tiap *use case*, akan tetapi diambil beberapa *sequence diagram* untuk *use case* tertentu. Sub bab 4.2.2. ini memodelkan *sequence diagram* untuk *use case-use case* berikut:

1. *Use case* Memesan barang.
2. *Use case* Melihat keranjang belanja.
3. *Use case* Melihat detail pesanan.
4. *Use case* Check out.

4.2.2.1. *Sequence Diagram* untuk *Use Case* Memesan barang

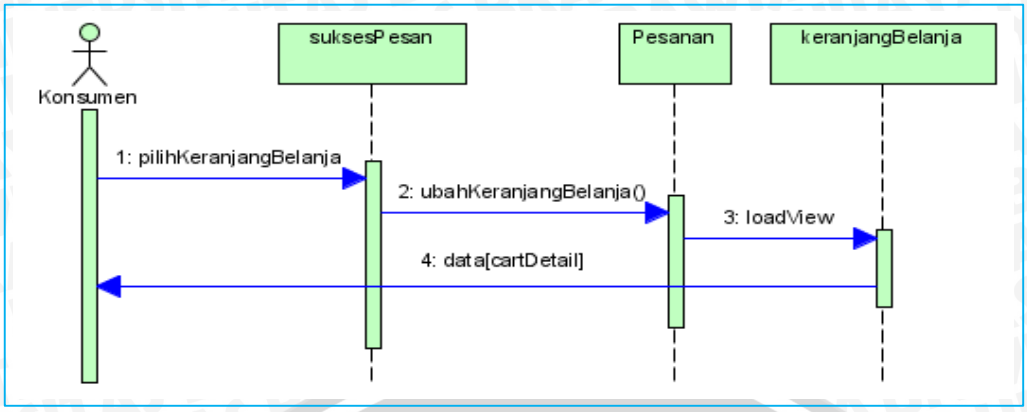
Gambar 4.12 adalah *sequence diagram* untuk *use case* memesan barang. *Use case* ini dipergunakan untuk menampilkan daftar barang per kategori barang.



Gambar 4. 12 *Sequence diagram* untuk *use case* memesan barang
Sumber: [Perancangan]

4.2.2.2. *Sequence Diagram* untuk *Use Case* Melihat keranjang belanja

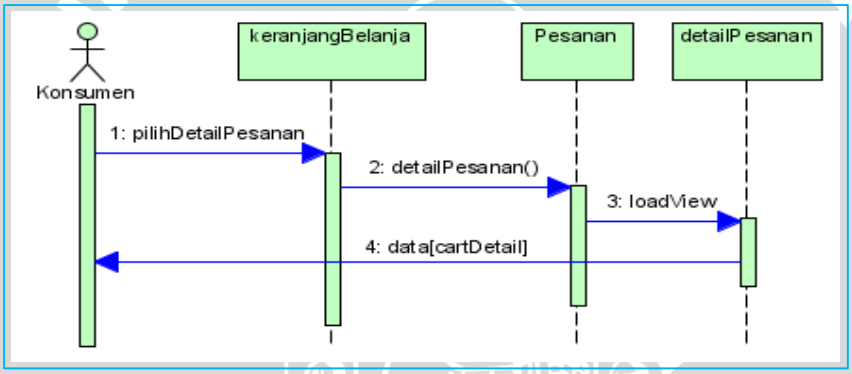
Gambar 4.13 adalah *sequence diagram* untuk *use case* melihat keranjang belanja. *Use case* ini dipergunakan untuk menampilkan daftar barang di keranjang belanja.



Gambar 4. 13 *Sequence diagram* untuk *use case* melihat keranjang belanja
Sumber: [Perancangan]

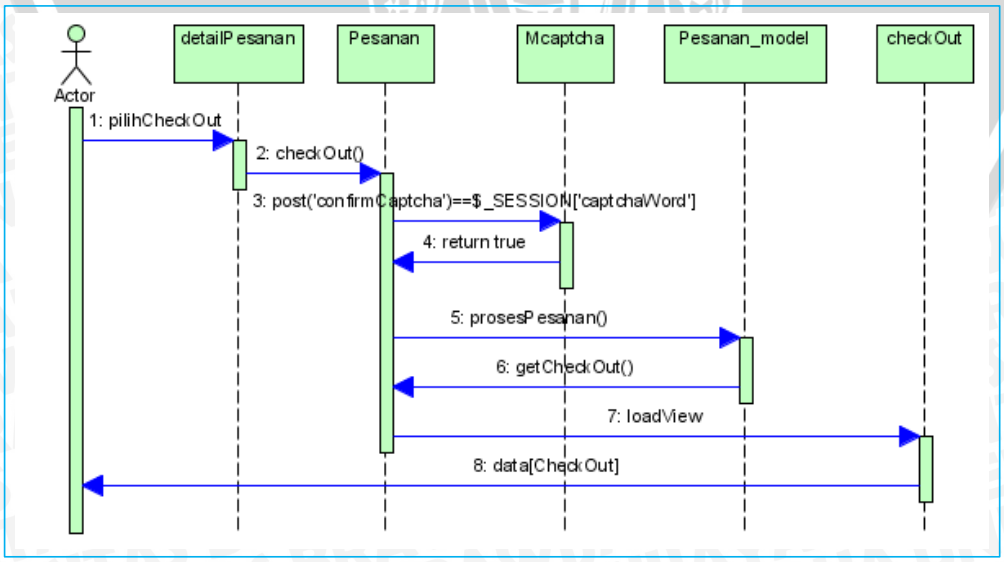
4.2.2.3. *Sequence Diagram* untuk *Use Case* Melihat detail pesanan

Gambar 4.14 adalah *sequence diagram* untuk *use case* melihat detail pesanan. *Use case* ini dipergunakan untuk menampilkan daftar barang yang dipesan.



Gambar 4. 14 *Sequence diagram* untuk *use case* melihat detail pesanan
Sumber: [Perancangan]

4.2.2.4. *Sequence Diagram* untuk *Use Case* Check out



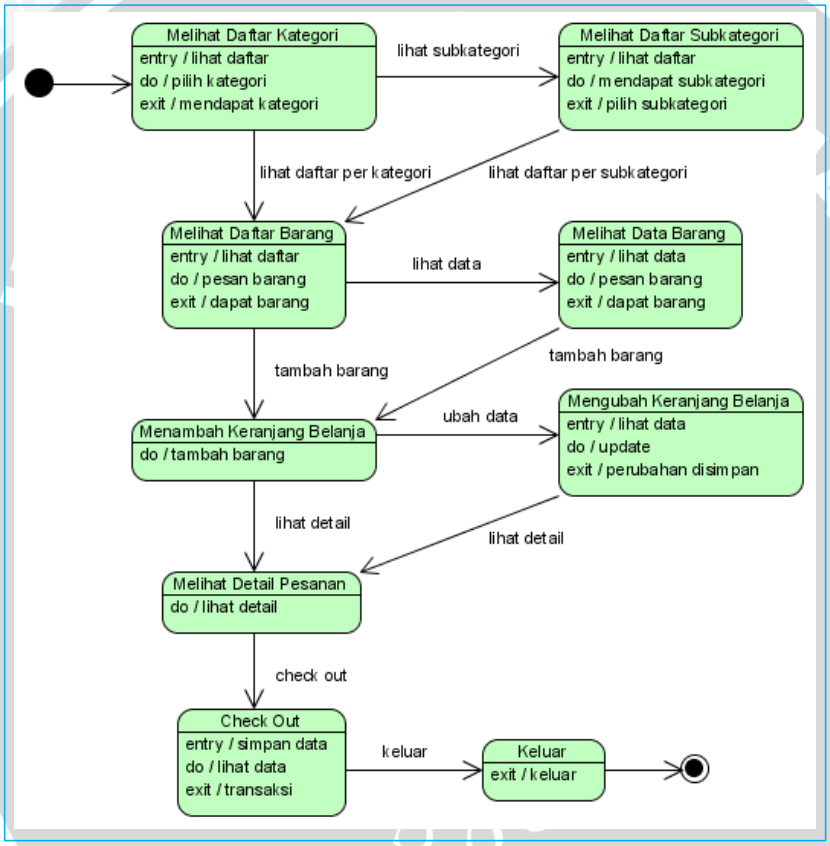
Gambar 4. 15 *Sequence diagram* untuk *use case* check out
Sumber: [Perancangan]

Gambar 4.15 adalah *sequence diagram* untuk *use case* check out. *Use case* ini dipergunakan untuk menampilkan data transaksi.

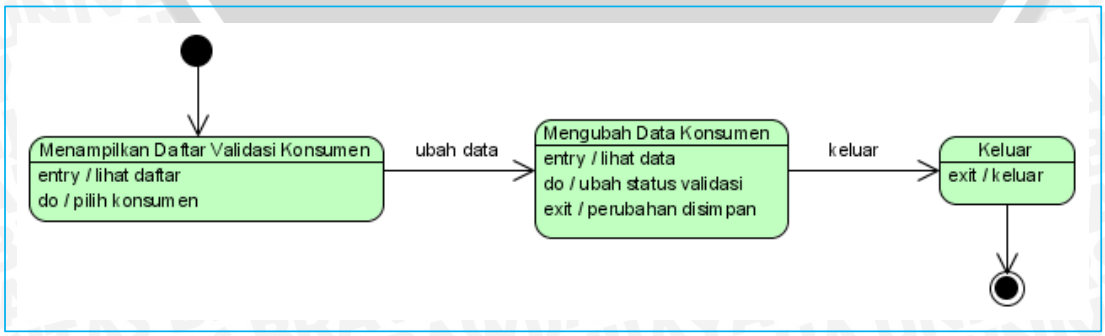
4.2.3. *State Diagram*

State Diagram memperlihatkan urutan *state* yang dilalui sebuah obyek, kejadian yang menyebabkan sebuah transisi dari suatu *state* atau aktivitas ke *state* atau aktivitas yang lain, dan aksi yang menyebabkan perubahan *state* atau aktivitas.

Gambar 4.16 menunjukkan *state diagram* pesanan dan Gambar 4.17 menunjukkan *state diagram* validasi konsumen.



Gambar 4. 16 *State diagram* pesanan
Sumber: [Perancangan]



Gambar 4. 17 *State diagram* validasi konsumen
Sumber: [Perancangan]

4.2.4. Perancangan Basis Data

Aplikasi *mobile online supermarket* ini memerlukan basis data untuk menyimpan informasi atau data. Perancangan basis data diawali dengan *Entity Relationship Diagram* pada Gambar 4.18, *Conceptual Data Model* pada Gambar 4.19, *Physical Data Model* pada Gambar 4.20, dan *Data Object Description*.

4.2.4.1. *Entity Relationship Diagram*

Entity Relationship Diagram merupakan alat pemodelan data yang membantu mengorganisasi data ke dalam entitas-entitas dan menentukan hubungan antarentitas.

Pada Gambar 4.18 ditunjukkan hubungan entitas-entitas yang menyusun basis data *mobile online supermarket* ini, terdiri dari 8 entitas yaitu kategori, subkategori, barang, suplier, detail_pesanan, transaksi, konsumen, dan pegawai.

Kedelapan entitas tersebut masing-masing memiliki atribut dan relasi antar entitas.

Entitas kategori memiliki atribut idkategori, kode_kategori, nama_kategori, dan deskripsi_kategori. Entitas kategori memiliki relasi mempunyai dengan entitas subkategori.

Entitas subkategori memiliki atribut idsubkategori, kode_subkategori, dan nama_subkategori. Entitas subkategori memiliki relasi mempunyai dengan entitas barang.

Entitas barang memiliki atribut idbarang, kode_barang, nama_barang, gambar, harga, deskripsi_barang, stok, idkategori, dan idsuplier. Selain memiliki relasi mempunyai dengan entitas kategori, entitas barang juga memiliki relasi menyuplai dengan entitas suplier dan relasi terdapat pada dengan entitas detail_pesanan.

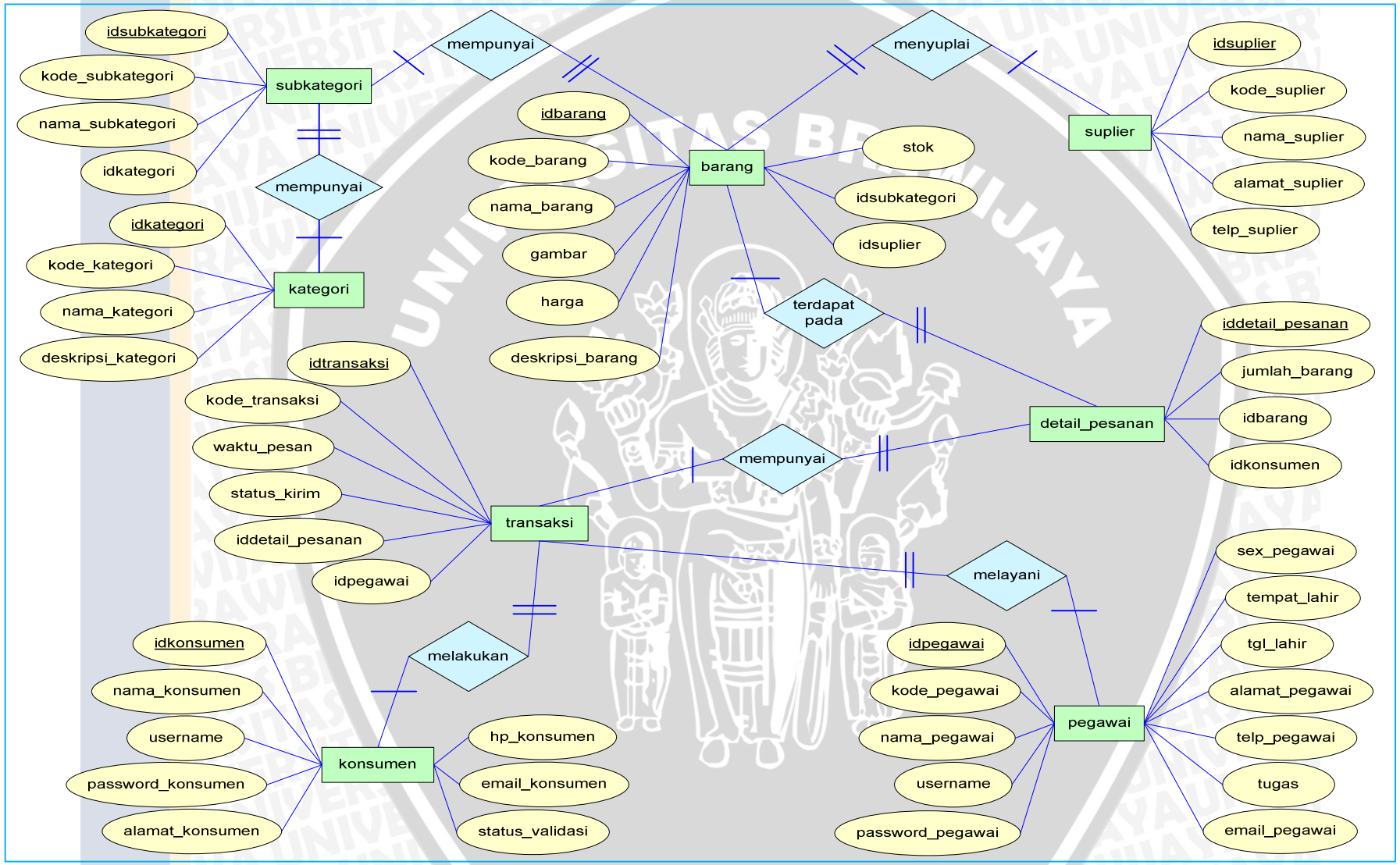
Entitas suplier memiliki atribut idsuplier, kode_suplier, nama_suplier, alamat_suplier, dan telp_suplier. Entitas suplier memiliki relasi menyuplai dengan entitas barang.

Entitas detail_pesanan memiliki atribut iddetail_pesanan, jumlah_barang, idbarang, dan idtransaksi. Entitas detail_pesanan memiliki dua relasi yaitu relasi terdapat pada dengan entitas barang dan relasi mempunyai dengan entitas transaksi.

Entitas transaksi memiliki atribut idtransaksi, kode_transaksi, waktu_pesan, status_kirim, idkonsumen, dan idpegawai. Selain memiliki relasi mempunyai dengan detail_pesanan, entitas transaksi memiliki relasi melayani dengan entitas pegawai, dan relasi melakukan dengan entitas konsumen.

Entitas konsumen memiliki atribut idkonsumen, nama_konsumen, username, password_konsumen, alamat_konsumen, hp_konsumen, email_konsumen, dan status_validasi. Entitas konsumen memiliki relasi melakukan dengan entitas transaksi.

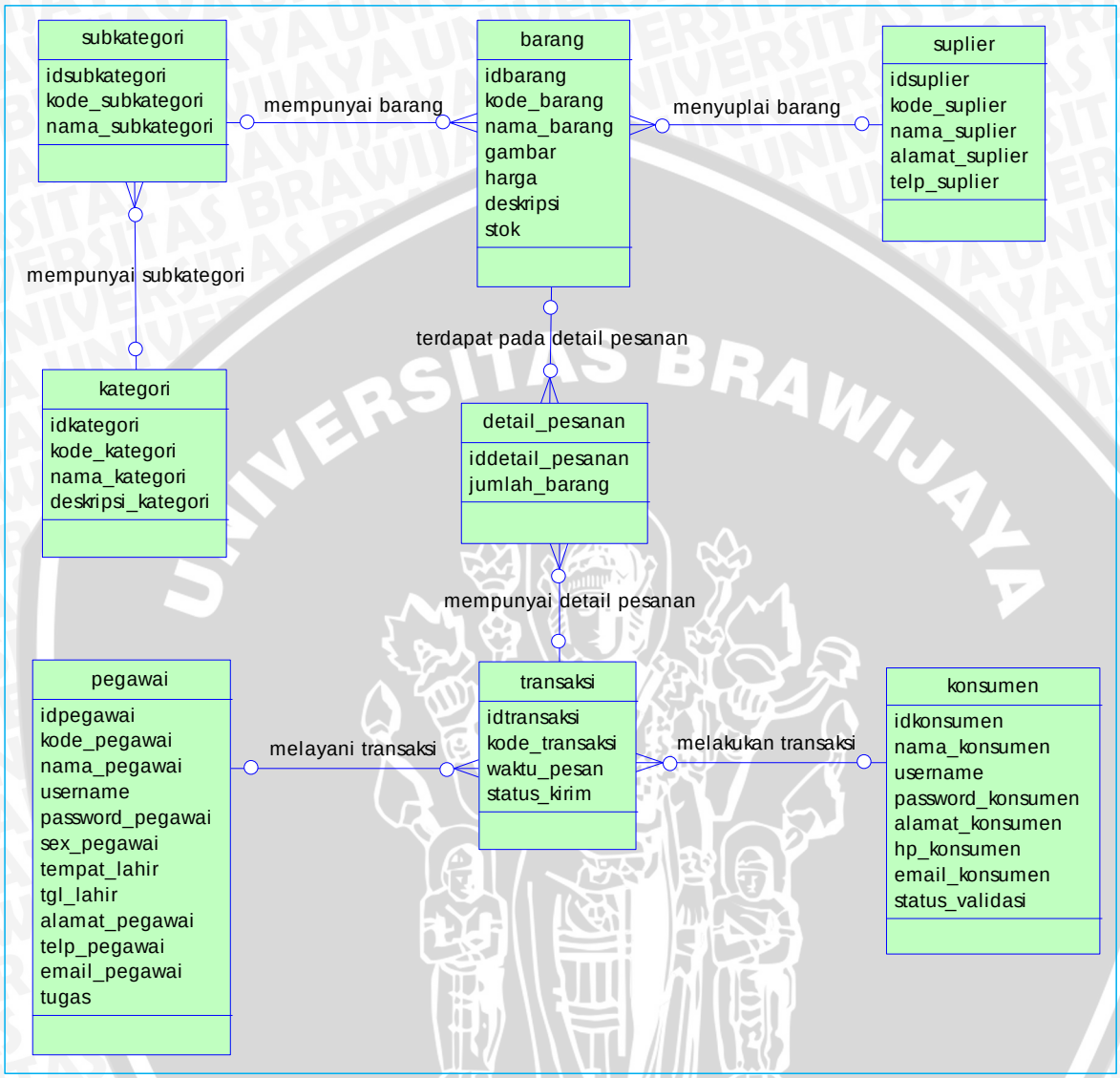
Entitas pegawai memiliki atribut idpegawai, kode_pegawai, nama_pegawai, username, password_pegawai, sex_pegawai, tempat_lahir, tgl_lahir, alamat_pegawai, telp_pegawai, email_pegawai, dan tugas. Entitas pegawai memiliki relasi melayani dengan entitas transaksi.



Gambar 4. 18 *Entity Relationship Diagram* sistem *mobile online supermarket*
Sumber: [Perancangan]

4.2.4.2. *Conceptual Data Model*

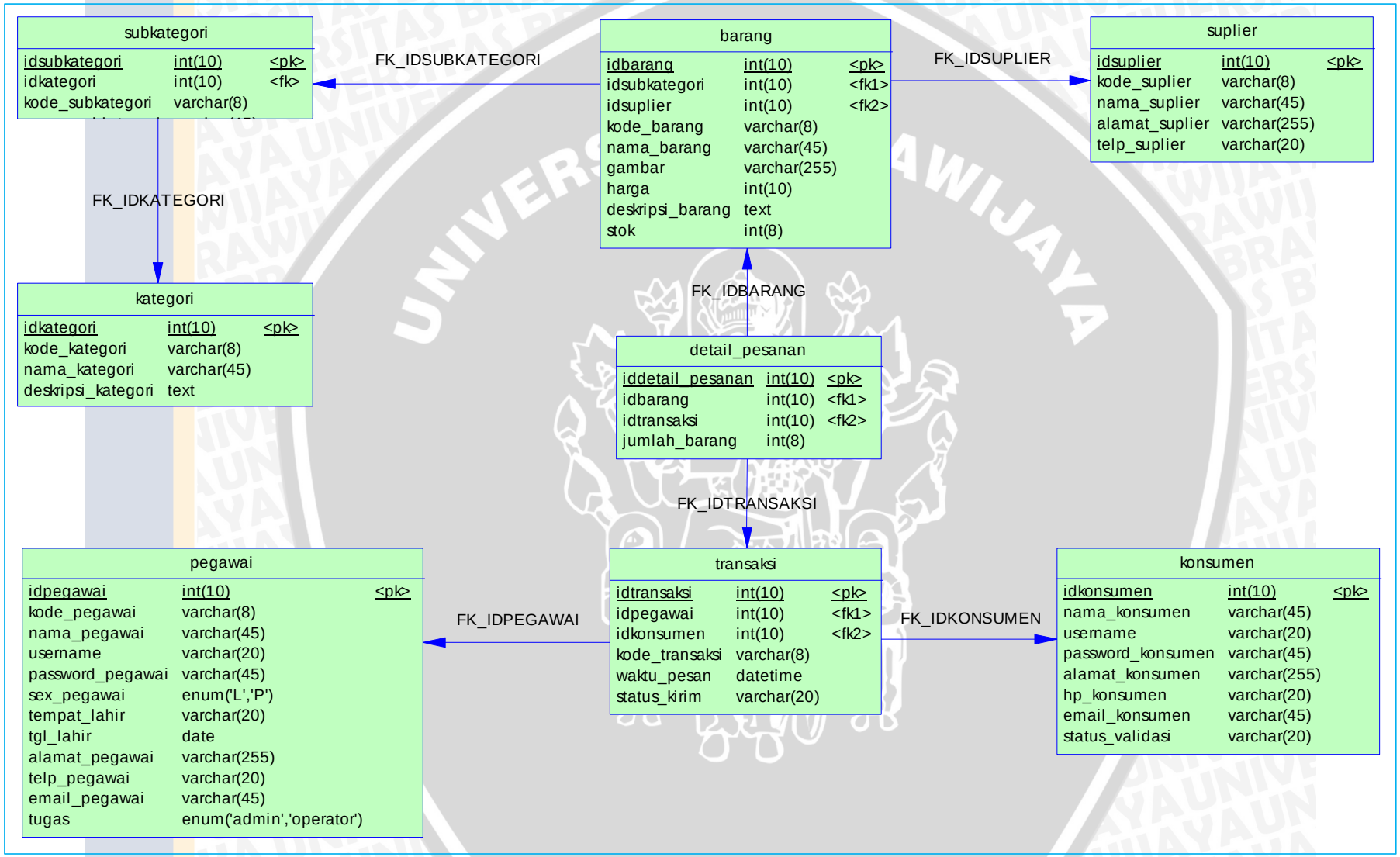
Conceptual Data Model menggambarkan struktur basis data secara detail dalam bentuk logika dan terdiri dari objek yang belum diimplementasikan secara langsung.



Gambar 4. 19 *Conceptual Data Model* sistem *supermarket online*
Sumber: [Perancangan]

4.2.4.3. *Physical Data Model*

Physical Data Model menggambarkan basis data secara detail dalam bentuk fisik dan memperlihatkan struktur penyimpanan data yang digunakan sesungguhnya.



Gambar 4. 20 *Physical Data Model* sistem *supermarket online*
Sumber: [Perancangan]

4.2.4.4. *Data Object Description*

Data Object Description merupakan penjelasan dari atribut pada masing-masing entitas yang terdapat pada sistem basis data. Atribut masing-masing entitas merupakan kolom-kolom yang terdapat pada masing-masing tabel.

4.2.4.4.1. *Data Object Description* untuk Tabel kategori

Tabel 4. 50 *Object Description* untuk tabel kategori

KEY	COLOUMN NAME	DATA TYPE	SIZE
PK	idkategori	int	10
	kode_kategori	varchar	8
	nama_kategori	varchar	45
	deskripsi_kategori	text	

Sumber: [Perancangan]

Tabel kategori mempunyai empat *field*. *Field* idkategori merupakan *primary key* dari tabel kategori. *Field* idkategori, kode_kategori, dan nama_kategori tidak boleh kosong. Tabel kategori berisi data-data kategori barang.

4.2.4.4.2. *Data Object Description* untuk Tabel subkategori

Tabel 4. 51 *Object Description* untuk tabel subkategori

KEY	COLOUMN NAME	DATA TYPE	SIZE
PK	idsubkategori	int	10
	kode_subkategori	varchar	8
	nama_subkategori	varchar	45
FK	idkategori	int	10

Sumber: [Perancangan]

Tabel kategori mempunyai empat *field*. *Field* idkategori merupakan *primary key* dari tabel kategori. *Field* idkategori, kode_kategori, dan nama_kategori tidak boleh kosong. Tabel kategori berisi data-data kategori barang.

4.2.4.4.3. *Data Object Description* untuk Tabel barang

Tabel 4. 52 *Data Object Description* untuk tabel barang

KEY	COLOUMN NAME	DATA TYPE	SIZE
PK	idbarang	int	10
	kode_barang	varchar	8
	nama_barang	varchar	45
	gambar	varchar	255
	harga	int	10
	deskripsi_barang	text	
	stok	int	8
FK	idsubkategori	int	10
FK	idsuplier	int	10

Sumber: [Perancangan]

Tabel barang mempunyai delapan *field*. *Field* idbarang merupakan *primary key* dari tabel barang. *Field* idkategori merupakan *foreign key* dari tabel kategori. *Field* idsuplier merupakan *foreign key* dari tabel suplier. *Field* idbarang, kode_barang, nama_barang,

gambar, harga, idkategori, dan idsuplier tidak boleh kosong. Tabel barang berisi data-data barang.

4.2.4.4.4. *Data Object Description* untuk Tabel suplier

Tabel 4. 53 *Data Object Description* untuk tabel suplier

KEY	COLOUMN NAME	DATA TYPE	SIZE
PK	idsuplier	int	10
	kode_suplier	varchar	8
	nama_suplier	varchar	45
	alamat_suplier	varchar	255
	telp_suplier	varchar	20

Sumber: [Perancangan]

Tabel suplier mempunyai lima *field*. *Field* idsuplier merupakan *primary key* dari tabel suplier. *Field* idsuplier, kode_suplier, nama_suplier, alamat_suplier, dan telp_suplier tidak boleh kosong. Tabel suplier berisi data-data suplier.

4.2.4.4.5. *Data Object Description* untuk Tabel konsumen

Tabel 4. 54 *Data Object Description* untuk tabel konsumen

KEY	COLOUMN NAME	DATA TYPE	SIZE
PK	idkonsumen	int	10
	nama_konsumen	varchar	45
	username	varchar	20
	password_konsumen	varchar	45
	alamat_konsumen	varchar	255
	hp_konsumen	varchar	20
	email_konsumen	varchar	45
	status_validasi	varchar	20

Sumber: [Perancangan]

Tabel konsumen mempunyai delapan *field*. *Field* idkonsumen merupakan *primary key* dari tabel konsumen. *Field* idkonsumen, nama_konsumen, username, password_konsumen, alamat_konsumen, hp_konsumen, email_konsumen, dan status_validasi tidak boleh kosong. Tabel konsumen berisi data-data konsumen.

4.2.4.4.6. *Data Object Description* untuk Tabel pegawai

Tabel 4. 55 *Data Object Description* untuk tabel pegawai

KEY	COLOUMN NAME	DATA TYPE	SIZE
PK	idpegawai	int	10
	kode_pegawai	varchar	8
	nama_pegawai	varchar	45
	username	varchar	20
	password_pegawai	varchar	45
	sex_pegawai	enum('L','P')	
	tempat_lahir	varchar	20
	tgl_lahir	date	
	alamat_pegawai	varchar	255
	telp_pegawai	varchar	20
	email_pegawai	varchar	45
	tugas	enum('admin','operator')	

Sumber: [Perancangan]

Tabel pegawai mempunyai dua belas *field*. *Field* idpegawai merupakan *primary key* dari tabel pegawai. *Field* idpegawai, kode_pegawai, nama_pegawai, username, password_pegawai, sex_pegawai, tempat_lahir, alamat_pegawai, telp_pegawai, email_pegawai, dan tugas tidak boleh kosong. Tabel pegawai berisi data-data pegawai.

4.2.4.4.7. *Data Object Description* untuk Tabel detail_pesanan

Tabel 4. 56 *Data Object Description* untuk tabel detail_pesanan

KEY	COLOUMN NAME	DATA TYPE	SIZE
PK	iddetail_pesanan	int	10
	jumlah_barang	int	8
FK	idbarang	int	10
FK	idtransaksi	int	10

Sumber: [Perancangan]

Tabel detail_pesanan mempunyai empat *field*. *Field* iddetil_pesanan merupakan *primary key* dari tabel detail_pesanan. *Field* idbarang merupakan *foreign key* dari tabel barang. *Field* idtransaksi merupakan *foreign key* dari tabel transaksi. *Field* iddetail_pesanan, jumlah_barang, idtransaksi, dan idbarang tidak boleh kosong. Tabel detail pesanan berisi data-data detail pesanan barang konsumen.

4.2.4.4.8. *Data Object Description* untuk Tabel transaksi

Tabel 4. 57 *Data Object Description* untuk tabel transaksi

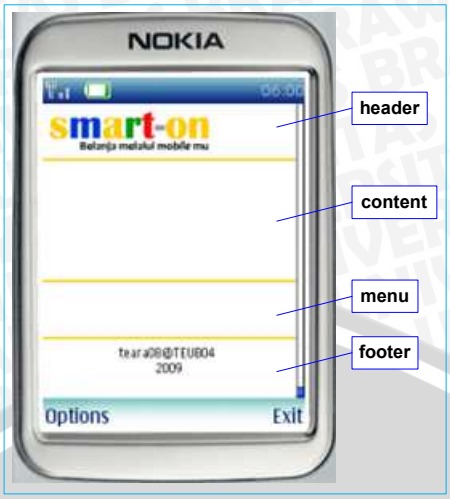
KEY	COLOUMN NAME	DATA TYPE	SIZE
PK	idtransaksi	int	10
	kode_transaksi	varchar	8
	waktu_pesan	datetime	
	status_kirim	varchar	20
FK	idkonsumen	int	10
FK	idpegawai	int	10

Sumber: [Perancangan]

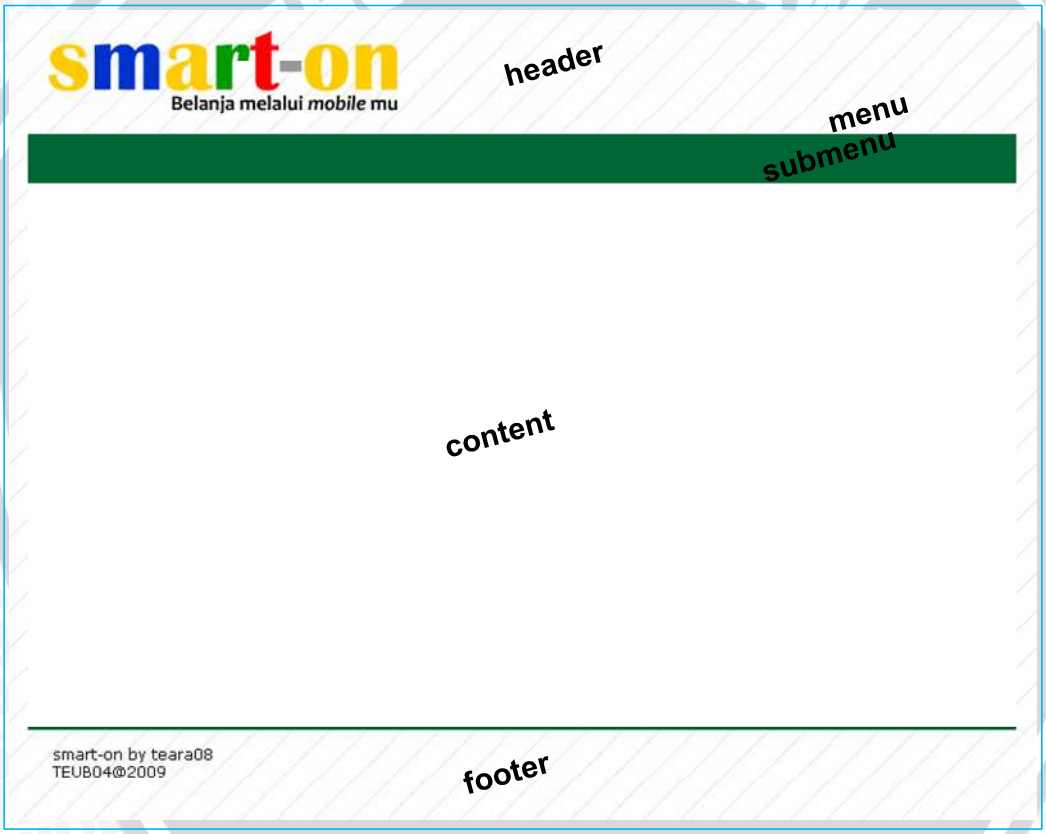
Tabel transaksi mempunyai enam *field*. *Field* idtransaksi merupakan *primary key* dari tabel transaksi. *Field* idkonsumen merupakan *foreign key* dari tabel konsumen. *Field* idpegawai merupakan *foreign key* dari tabel pegawai. *Field* idtransaksi, kode_transaksi, status_kirim, idkonsumen, dan idpegawai tidak boleh kosong. Tabel transaksi berisi data-data transaksi yang dilakukan oleh konsumen.

4.2.5. Perancangan Antarmuka

Pada sistem *mobile online supermarket* ini, halaman antarmuka terdiri dari beberapa komponen, yaitu: *header*, *menu*, *sub menu*, *content*, dan *footer*. Rancangan dari halaman antarmuka untuk konsumen melalui mobile browser dapat dilihat pada Gambar 4.21, dan rancangan dari halaman antarmuka untuk administrator dan operator ditunjukkan pada Gambar 4.22.



Gambar 4. 21 Rancangan antarmuka konsumen
Sumber: [Perancangan]



Gambar 4. 22 Rancangan antarmuka administrator dan operator
Sumber: [Perancangan]

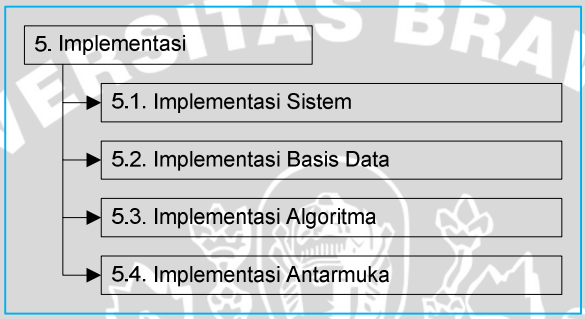
4.2.6. Perancangan Keamanan Sistem

Metode keamanan sistem yang digunakan adalah fungsi MD5 pada *script* PHP. Fungsi MD5 digunakan untuk menghitung sekumpulan karakter string menggunakan RSA *data security* MD5. MD5 adalah *message-diggest algorithm* yang digunakan untuk melakukan pembuatan *password* dalam menjamin keamanan data.

Selain menggunakan MD5, dalam skripsi ini juga menggunakan CAPTCHA (*Completely Automated Public Turing test to tell Computer and Human Apart*). CAPTCHA adalah suatu jenis tes yang diberikan kepada *user* (biasanya pengunjung *website*) untuk memastikan bahwa *user* tersebut adalah seorang manusia dan bukan sebuah komputer (biasa disebut *bot*). Hampir semua mekanisme CAPTCHA yang dipakai di situs-situs internet adalah menggunakan gambar kumpulan huruf-huruf dan angka-angka yang dirusak bentuknya sehingga masih dapat dibaca oleh manusia tetapi tidak dapat dibaca oleh komputer.

BAB V
IMPLEMENTASI

Bab ini membahas tentang implementasi perangkat lunak *mobile online supermarket* berdasarkan hasil yang telah didapatkan dari analisis kebutuhan dan proses perancangan perangkat lunak pada bab 4. Bab ini terdiri dari pembahasan implementasi algoritma dan implementasi antarmuka, dan penjelasan tentang spesifikasi perangkat yang digunakan. Diagram pohon analisis implementasi sistem pada bab ini ditunjukkan pada Gambar 5.1.



Gambar 5. 1 Diagram pohon implementasi sistem
Sumber: [Implementasi]

5.1. Implementasi Sistem

Agar bisa berjalan sesuai dengan kebutuhan sistem *mobile online supermarket*, sistem diimplementasikan pada perangkat keras dan perangkat lunak dengan spesifikasi tertentu.

5.1.1. Spesifikasi Perangkat Keras

Sistem *supermarket online* ini merupakan aplikasi *client-server*. Pada skripsi ini baik *client* maupun *server* terdapat pada satu komputer yang sama. Pengembangan sistem *mobile online supermarket* menggunakan komputer dengan spesifikasi sebagai berikut:

Tabel 5. 1 Spesifikasi perangkat keras

Prosesor	Intel Core Duo T2450
Memori (RAM)	1.5 GB
Harddisk	120 GB
VGA card	Intel VGA 950

Sumber: [Implementasi]

5.1.2. Spesifikasi Perangkat Lunak

Pengembangan sistem *mobile online supermarket* menggunakan perangkat lunak dengan spesifikasi sebagai berikut:

Tabel 5. 2 Spesifikasi perangkat lunak

Sistem operasi	Windows XP Professional Service Pack 2
Bahasa pemrograman	XHTML MP WCSS PHP 5.2.6
Web server	Apache 2.2.9
Database	MySQL 5.0.67
IDE (*Integrated Development Environment*)	Macromedia Dreamweaver 8, Nokia series 40 5th edition SDK emulator

Sumber: [Implementasi]

5.2. Implementasi Basis Data

Implementasi basis data *mobile online supermarket* menggunakan *Data Definition Languange* (DDL). Skema basis data ditentukan sekumpulan definisi yang dinyatakan dengan bahasa tertentu yang disebut *Data Definition Languange* (DDL). Sebagai contoh perintah pada SQL: (*create*) untuk membuat objek *database*, (*alter*) untuk memodifikasi objek *database*, dan (*drop*) untuk menghapus objek *database*. [SIM-06]

5.2.1. Implementasi *create database* smarton

DDL yang digunakan dalam membentuk basis data *smarton* adalah sebagai berikut:

```
CREATE DATABASE smarton;
```

5.2.2. Implementasi *create* tabel

DDL yang digunakan dalam membentuk tabel-tabel mobile online supermarket adalah sebagai berikut :

a. Tabel kategori

```
CREATE TABLE kategori (
    idkategori INT(10) NOT NULL AUTO_INCREMENT,
    kode_kategori VARCHAR(8) NOT NULL,
    nama_kategori VARCHAR(45) NOT NULL,
    deskripsi_kategori TEXT NULL,
    PRIMARY KEY(idkategori)
);
```

b. Tabel subkategori

```
CREATE TABLE kategori (
    idsubkategori INT (10) NOT NULL AUTO_INCREMENT,
    idkategori INT (10) NOT NULL,
    kode_subkategori VARCHAR (8) NOT NULL,
    nama_subkategori VARCHAR (45) NOT NULL,
    PRIMARY KEY(idsubkategori)
);
```

c. Tabel barang

```
CREATE TABLE barang (
    idbarang INT(10) NOT NULL AUTO_INCREMENT,
    idsuplier INT(10) NOT NULL,
    idkategori INT(10) NOT NULL,
    idsubkategori INT (10) NOT NULL,
    kode_barang VARCHAR(8) NOT NULL,
    nama_barang VARCHAR(45) NOT NULL,
```

```
    gambar VARCHAR(255) NOT NULL,
    harga INT(10) NOT NULL,
    deskripsi_barang TEXT NULL,
    stok int(8)NULL,
    PRIMARY KEY(idbarang)
);
```

d. Tabel suplier

```
CREATE TABLE suplier (
    idsuplier INT(10) NOT NULL AUTO_INCREMENT,
    kode_suplier VARCHAR(8) NOT NULL,
    nama_suplier VARCHAR(45) NOT NULL,
    alamat_suplier VARCHAR(255) NOT NULL,
    telp_suplier VARCHAR(20) NOT NULL,
    PRIMARY KEY(idsuplier)
);
```

e. Tabel konsumen

```
CREATE TABLE konsumen (
    idkonsumen INT(10) NOT NULL AUTO_INCREMENT,
    nama_konsumen VARCHAR(45) NOT NULL,
    username VARCHAR(20) NOT NULL,
    password_konsumen VARCHAR(45) NOT NULL,
    alamat_konsumen VARCHAR(255) NOT NULL,
    hp_konsumen VARCHAR(20) NOT NULL,
    email_konsumen VARCHAR(45) NOT NULL,
    status_validasi VARCHAR(20) NOT NULL,
    PRIMARY KEY(idkonsumen)
);
```

f. Tabel pegawai

```
CREATE TABLE pegawai (
    idpegawai INT(10) NOT NULL AUTO_INCREMENT,
    kode_pegawai VARCHAR(8) NOT NULL,
    nama_pegawai VARCHAR(45) NOT NULL,
    username VARCHAR(20) NOT NULL,
    password_pegawai VARCHAR(45) NOT NULL,
    sex_pegawai ENUM('L','P') NOT NULL,
    tempat_lahir VARCHAR(20) NOT NULL,
    tgl_lahir DATE NULL,
    alamat_pegawai VARCHAR(255) NOT NULL,
    telp_pegawai VARCHAR(20) NOT NULL,
    email_pegawai VARCHAR(45) NOT NULL,
    tugas ENUM('admin','operator') NOT NULL,
    PRIMARY KEY(idpegawai)
);
```

g. Tabel detail_pesanan

```
CREATE TABLE detail_pesanan (
    iddetail_pesanan INT(10) NOT NULL AUTO_INCREMENT,
    idbarang INT(10) NOT NULL,
    idtransaksi INT(10) NOT NULL,
    jumlah_barang INT(8) NOT NULL,
    PRIMARY KEY(iddetail_pesanan)
);
```

h. Tabel transaksi

```
CREATE TABLE transaksi (
    idtransaksi INT(10) NOT NULL AUTO_INCREMENT,
    idpegawai INT(10) NOT NULL,
    idkonsumen INT(10) NOT NULL,
    kode_transaksi VARCHAR(8) NOT NULL,
    waktu_pesan DATETIME NOT NULL,
    status_kirim VARCHAR(20) NOT NULL,
    PRIMARY KEY(idtransaksi)
);
```

Setelah membuat semua tabel, maka tabel-tabel yang memiliki relasi dengan tabel yang lain akan direlasikan dengan memodifikasi (alter) pada tabel-tabel yang memiliki hubungan.

```
alter table barang
add foreign key fk_barang_relation__kategori (idkategori) references
kategori (idkategori) on update restrict on delete restrict;

alter table barang
add foreign key fk_barang_relation__subkategori (idsubkategori) references
subkategori (idsubkategori) on update restrict on delete restrict;

alter table barang
add foreign key fk_barang_relation__suplier (idsuplier) references suplier
(idsuplier) on update restrict on delete restrict;

alter table detail_pesanan
add foreign key fk_detail_pesanan_relation__barang (idbarang) references
barang (idbarang) on update restrict on delete restrict;

alter table detail_pesanan
add foreign key fk_detail_pesanan_relation__transaksi (idtransaksi)
references transaksi (idtransaksi) on update restrict on delete restrict;

alter table subkategori
add foreign key fk_subkategori_relation__kategori (idkategori) references
kategori (idkategori) on update restrict on delete restrict;

alter table transaksi
add foreign key fk_transaksi_relation__pegawai (idpegawai) references
pegawai (idpegawai) on update restrict on delete restrict;

alter table transaksi
add foreign key fk_transaksi_relation__konsumen (idkonsumen)
references konsumen (idkonsumen) on update restrict on delete restrict;
```

5.3. Implementasi Algoritma

Setiap *class* yang telah dirancang pada bab perancangan direalisasikan pada sebuah *file* program .php. Tabel 5.3 menjelaskan hubungan antara *package*, *class* dengan *file* yang terdapat dalam *package*,dan *method-method* yang terdapat dalam setiap *class*. Pada setiap *class* terdapat *method* sebagai program untuk membangun halaman-halaman pada aplikasi *mobile online supermarket* ini.

Tabel 5. 3 Implementasi hubungan *package*, *class*, dan *method*

No.	*Package*	*Class* dan atau file (.php)	*Method*
1.	controllers	Barang (barang.php)	Barang()
2.			cekSessionPegawai()
3.			cekSessionKonsumen()
4.			daftarBarang()
5.			tambahBarang()
6.			cekKode()
7.			tambahStok()
8.			editStok($idBarang)
9.			lihatBarang()
10.			ubahBarang($id)
11.			hapusBarang()
12.			cariBarang($keyword = NULL, $offset = NULL)
13.			daftarBarangUser()
14.			detailBarangUser()
15.			cariBarangUser($keyword = NULL, $offset = NULL)
16.			daftarBarangKonsumen()
17.			detailBarangKonsumen()
18.			cariBarangKonsumen($keyword = NULL, $offset = NULL)
19.		Dashboard (dashboard.php)	dashboard()
20.			cekSession()
21.			index()
22.		DashboardKonsumen (dashboardKonsumen.php)	dashboardKonsumen()
23.			cekSession()
24.			index()
25.		DashboardOperator (dashboardOperator.php)	dashboardOperator()
26.			cekSession()
27.			index()
28.		Kategori (kategori.php)	Kategori()
29.			cekSessionPegawai()
30.			cekSessionKonsumen()
31.			daftarKategori()
32.			tambahKategori()
33.			cekKode()
34.			lihatKategori()

35.			ubahKategori($id)
36.			hapusKategori($id)
37.			lihatBarang($idKategori)
38.			cariKategori($keyword = NULL, $offset = NULL)
39.			daftarKategoriUser()
40.			lihatBarangUser($idKategori)
41.			daftarKategoriKonsumen()
42.			lihatBarangKonsumen($idKategori)
43.		Konsumen (konsumen.php)	Konsumen()
44.			cekSessionPegawai()
45.			cekSessionKonsumen()
46.			Registrasi()
47.			cekUsername()
48.			cekEmail()
49.			daftarKonsumen()
50.			lihatKonsumen()
51.			hapusKonsumen($id)
52.			cariKonsumen($keyword = NULL, $offset = NULL)
53.			ubahUserPass()
54.			ubahDataku()
55.		LoginKonsumen (loginKonsumen.php)	loginKonsumen()
56.			cekSession()
57.			index()
58.			cek()
59.			logout()
60.		LoginPegawai (loginPegawai.php)	loginPegawai()
61.			cekSession()
62.			index()
63.			cek()
64.			logout()
65.		LupaPassword (lupaPassword.php)	lupaPassword()
66.			index()
67.			lupaPass($email = NULL)
68.			function index2()
69.			lupaPass2($email = NULL)
70.		Pegawai (pegawai.php)	Pegawai()

71.			cekSessionPegawai()
72.			daftarPegawai()
73.			tambahPegawai()
74.			cekKode()
75.			cekUsername()
76.			cekEmail()
77.			lihatPegawai()
78.			ubahPegawai($id)
79.			ubahAdmin()
80.			hapusPegawai($id)
81.			cariPegawai($keyword = NULL, $offset = NULL)
82.			ubahOperator()
83.		Pesanan (pesanan.php)	pesanan()
84.			cekSessionPegawai()
85.			cekSessionKonsumen()
86.			tambahKeranjangBelanja()
87.			ubahKeranjangBelanja()
88.			hapusItem($rowid)
89.			detailPesanan()
90.			checkOut()
91.			hapusKeranjangBelanja()
92.			daftarTransaksi()
93.			ubahTransaksi($idtransaksi)
94.			nota($idtransaksi)
95.			cetak()
96.			cariTransaksi($keyword = NULL, $offset = NULL)
97.			cariTransaksiKode($keyword = NULL, $offset = NULL)
98.		Subkategori (subkategori.php)	Subkategori()
99.			cekSessionPegawai()
100.			cekSessionKonsumen()
101.			tambahSubkategori()
102.			cekKode()
103.			ubahSubkategori($id)
104.			hapusSubkategori($id)
105.			lihatBarang($idSubkategori)
106.			subkategoriDropDown($id)

107.			lihatSubkategoriUser($idSubkategori)
108.			lihatBarangUser($idSubkategori)
109.			lihatSubkategoriKonsumen($idSubkategori)
110.			lihatBarangKonsumen($idSubkategori)
111.		Suplier (suplier.php)	Suplier()
112.			cekSessionPegawai()
113.			daftarSuplier()
114.			tambahSuplier()
115.			cekKode()
116.			lihatSuplier()
117.			ubahSuplier($id)
118.			hapusSuplier()
119.			cariSuplier($keyword = NULL, $offset = NULL)
120.		ValidasiKonsumen (validasiKonsumen.php)	validasiKonsumen()
121.			cekSessionPegawai()
122.			daftarValidasiKonsumen()
123.			ubahValidasiKonsumen($id)
124.			cariKonsumen($keyword = NULL, $offset = NULL)
125.	Models	Barang_model (barang_model.php)	Barang_model()
126.			tambahBarang($fileName)
127.			cekKode()
128.			getBarang($num,$offset)
129.			getBarangDetail($id)
130.			getStok($id)
131.			ubahStok($idBarang,$stokTambah,$stok)
132.			ubahStok2($idBarang,$stokAkhir)
133.			getJumlahStok($idBarang)
134.			getJumlahStok2($idBarang)
135.			getIdBarang($id)
136.			ubahBarang($id)
137.			ubahBarangBergambar($id,$gambarLama,$gambarBaru)
138.			hapusBarang($id,$imagenya)
139.			count_cari_barang($keyword)
140.			cariBarang($keyword, $limit, $offset)
141.		Kategori_model (kategori_model.php)	Kategori_model()
142.			tambahKategori()

143.			cekKode()
144.			getKategoriDropDown()
145.			getKategori($num,$offset)
146.			getKategoriDetail($id)
147.			getIdKategori($id)
148.			ubahKategori($id)
149.			count_sebagian($idKategori)
150.			lihatBarang($idKategori,$limit,$offset)
151.			hapusKategori($id)
152.			count_cari_kategori($keyword)
153.			cariKategori($keyword, $limit, $offset)
154.		Konsumen_model (konsumen_model.php)	Konsumen_model()
155.			Registrasi()
156.			cekUsername()
157.			cekEmail()
158.			getKonsumen($num,$offset)
159.			getKonsumenDetail($id)
160.			getIdKonsumen($id)
161.			ubahUserPass($id)
162.			ubahKonsumen($id)
163.			ubahKonsumenStatus($id)
164.			hapusKonsumen($id)
165.			count_cari_konsumen($keyword)
166.			cariKonsumen($keyword, $limit, $offset)
167.		Login_model (login_model.php)	login_model()
168.			cekUserPegawai()
169.			getDataPegawai()
160.			cekUserKonsumen()
161.			getDataKonsumen()
162.		LupaPassword_model (lupapassword_model.php)	lupaPassword_model()
163.			cekEmail($email)
164.			ubahPassword($email,$passBaru)
165.			cekEmail2($email)
166.			ubahPassword2($email,$passBaru)
167.		Mcaptcha	_construct($configVal = array())
168.			initialize ($configVal = array())

169.			generateCaptcha ()
170.		Pegawai_model (pegawai_model.php)	Pegawai_model()
171.			tambahPegawai()
172.			cekKode()
173.			cekUsername()
174.			cekEmail()
175.			getPegawai($num,$offset)
176.			getPegawaiDetail($id)
177.			getIdPegawai($id)
178.			ubahPegawai($id)
179			ubahAdmin($id)
180.			ubahOperator($id)
181.			hapusPegawai($id)
182.			count_cari_pegawai($keyword)
183.			cariPegawai($keyword, $limit, $offset)
184.		Pesanan_model (pesanan_model.php)	pesanan_model()
185.			getCheckOut($idKonsumen,$idtransaksi)
186.			getDetailTransaksi($idtransaksi)
187.			prosesPesanan()
188.			masukkanDetailPesanan($dataBarang)
189.			getTransaksi($num,$offset)
190.			getIdTransaksi($idtransaksi)
191.			statusKirim($idtransaksi)
192.			ubahStatusKirim($idtransaksi)
193.			getOperator()
194.			masukkanOperator()
195.			count_cari_statusKirim($keyword)
196.			cariStatusKirim($keyword, $limit, $offset)
197.			count_cari_kodeTransaksi($keyword)
198.			cariKodeTransaksi($keyword, $limit, $offset)
199.		Subkategori_model (subkategori_model.php)	Subkategori_model()
200.			tambahSubkategori()
201.			cekKode()
202.			getSubkategoriDropDown()
203.			getSubkategoriDropDown2()
204.			getSubkategori()

205.			getSubkategoriHp()
206.			getIdSubkategori($id)
207.			ubahSubkategori($id)
208.			count_sebagian($idSubkategori)
209.			lihatBarang($idSubkategori,$limit,$offset)
210.			hapusSubkategori($id)
211.		Suplier_model (suplier_model.php)	Suplier_model()
212.			tambahSuplier()
213.			cekKode()
214.			getSuplierDropDown()
215.			getSuplier($num,$offset)
216.			getSuplierDetail($id)
217.			getIdSuplier($id)
218.			ubahSuplier($id)
219.			hapusSuplier($id)
220.			count_cari_suplier($keyword)
221.			cariSuplier($keyword, $limit, $offset)
222.	views::admin	adminHome.php	-
223.		adminUbah.php	
224.		barangCari.php	
225.		barangDaftar.php	
226.		barangLihat.php	
227.		barangTambah.php	
228.		barangUbah.php	
229.		cariGagal.php	
230.		emailSama.php	
231.		kategoriCari.php	
232.		kategoriDaftar.php	
233.		kategoriLihat.php	
234.		kategoriTambah.php	
235.		kategoriUbah.php	
236.		kodeSama.php	
237.		konsumenCari.php	
238.		konsumenDaftar.php	
239.		konsumenLihat.php	
240.		lihatBarang.php	

241.		pegawaiCari.php	
242.		pegawaiDaftar.php	
243.		pegawaiLihat.php	
244.		pegawaiTambah.php	
245.		pegawaiUbah.php	
246.		stokTambah.php	
247.		subkategoriTambah.php	
248.		subkategoriUbah.php	
249.		suksesHapus.php	
250.		suksesTambah.php	
251.		suksesUbah.php	
252.		suksesUbahDataku.php	
253.		suplierCari.php	
254.		suplierDaftar.php	
255.		suplierLihat.php	
256.		suplierTambah.php	
257.		suplierUbah.php	
258.		userSama.php	
259.	views::konsumen	barangKonsumen.php	-
260.		barangKonsumenCari.php	
261.		barangUser.php	
262.		barangUserCari.php	
263.		cariGagalKonsumen.php	
264.		cariGagalUser.php	
265.		checkOut.php	
266.		detailBarangKonsumen.php	
267.		detailBarangUser.php	
268.		detailPesanan.php	
269.		emailSama.php	
270.		emailSamaReg.php	
271.		kategoriKonsumen.php	
272.		kategoriUser.php	
273.		keranjangBelanja.php	
274.		konsumenHome.php	
275.		loginKonsumen.php	
276.		lupaPassword.php	

277.		registrasi.php	
278.		salahChaptcha.php	
279.		suksesEmail.php	
280.		suksesHapus.php	
281.		suksesPesan.php	
282.		suksesRegistrasi.php	
283.		suksesUbah.php	
284.		suksesUbahUserPass.php	
285.		ubahDataku.php	
286.		ubahUserPass.php	
287.		userSama.php	
288.		userSamaReg.php	
289.	views::operator	cariGagal.php	-
290.		cetak.php	
291.		loginPegawai.php	
292.		lupaPassword.php	
293.		nota.php	
294.		operatorHome.php	
295.		operatorUbah.php	
296.		pesananCari.php	
297.		pesananDaftar.php	
298.		pesananKodeCari.php	
299.		pesananLihat.php	
300.		suksesEmail.php	
301.		suksesProses.php	
302.		suksesUbah.php	
303.		suksesUbahDataku.php	
304.		userSama.php	
305.		validasiKonsumenCari.php	
306.		validasiKonsumenDaftar.php	
307.		validasiKonsumenUbah.php	

Sumber: [Implementasi]

5.3.1. Pseudocode Method `detailBarangUser()` dari Class Barang

Method `detailBarangUser()` digunakan untuk menampilkan data barang yang dapat dilihat *user* yang tidak *login*. *Pseudocode* dari *method* `detailBarangUser()` ditunjukkan dalam Gambar 5.2.

```
METHOD : detailBarangUser()

DECLARATION :
data IS Array

DESCRIPTION :
1     data['daftar']  <-  CALL barang_model->getBarangDetail(CALL uri->segment(3))
2     CALL load->view('konsumen/detailBarangUser.php',$data)
```

Gambar 5. 2 *Pseudocode method* detailBarangUser()
Sumber: [Implementasi]

Penjelasan *pseudocode method* `detailBarangUser()` pada Gambar 5.2 adalah sebagai berikut:

1. Baris 1-2 menjelaskan mengambil data dari `getBarangDetail(CALL uri->segment(3))` dan menampilkannya pada halaman.

5.3.2. Pseudocode Method `tambahKeranjangBelanja()` dari Class Pesanan

Method `tambahKeranjangBelanja()` digunakan untuk menampilkan dan menambah data barang yang dipesan. *Pseudocode* dari *method* `tambahKeranjangBelanja()` ditunjukkan dalam Gambar 5.3.

Penjelasan *pseudocode method* `tambahKeranjangBelanja()` pada Gambar 5.3 adalah sebagai berikut:

1. Baris 1 menjelaskan identifikasi konsumen yang *login*.
2. Baris 2 menjelaskan masukan data dari konsumen berupa idBarang.
3. Baris 3 menjelaskan pembuatan *query* ke basis data untuk mendapatkan data barang berdasarkan idbarang.
4. Baris 4 menjelaskan pemeriksaan jumlah *record* lebih besar dari 0.
5. Baris 5-14 menjelaskan masukan data berupa data idbarang, kode_barang, nama_barang, harga, jumlah_barang, dan memeriksa stok barang.
6. Baris 15 menjelaskan pemeriksaan apakah jumlah barang yang dipesan harus lebih kecil dari stok barang.
7. Baris 16-25 menjelaskan pemeriksaan isi keranjang belanja tidak kosong, jumlah barang nilai awal 0. Untuk masing-masing barang yang sudah ada, penambahan jumlah barang pada idbarang yang sama. Jika berhasil dijalankan (bernilai true), jumlah barang bertambah.

8. Baris 26-31 menjelaskan aliran alternatif dari baris 17-25, apabila proses penambahan gagal (bernilai false).
9. Baris 32-35 menjelaskan pemeriksaan jumlah barang bernilai 0. Maka memasukan barang. Jika berhasil dilakukan (bernilai true) memanggil halaman pesan.
10. Baris 36-40 menjelaskan aliran alternatif dari baris 32-35, apabila proses gagal dilakukan (bernilai false) maka memanggil halaman pesan.
11. Baris 41-44 menjelaskan aliran alternatif dari baris 16-25, keranjang belanja kosong. Jika proses menambahkan barang berhasil (bernilai true), maka memanggil halaman pesan.
12. Baris 45-49 menjelaskan aliran alternatif baris 41-44, apabila proses menambahkan barang gagal (bernilai false), maka memanggil halaman pesan.
13. Baris 50-54 menjelaskan aliran alternatif baris 15, apabila jumlah barang yang dipesan lebih besar dari stok barang, maka memanggil halaman pesan.

```
METHOD : tambahKeranjangBelanja

DECLARATION :
idBarang, idbarang, sisa, sisaStok, i, IS int
query, item, isiCart, data, key, result IS Array
kodeBarang IS Varchar

DESCRIPTION :
1     CALL cekSessionKonsumen()
2     idBarang  <-  CALL input->post('idBarang')
3     query  <-  CALL db->get_where('barang',array('idbarang'=>$idBarang),1)
4     IF query->num_rows()>0
5          item  <-  query->row()
6          kodeBarang  <-  item->kode_barang
7          idbarang  <-  item->idbarang
8          isiCart  <-  CALL cart->contents()
9          data  <-  Array('id' <- $item->idbarang, 'name' <- $item->nama_barang,
10              'price' <- item->harga, 'qty' <- CALL input->post('jumlah_barang'))
11         sisaStok  <-  CALL barang_model->getJumlahStok($idBarang)
12         FOREACH key  <-  sisaStok
13              sisa  <-  key->stok
14         END FOREACH
15         IF data['qty'] < sisa
16              IF isiCart > 0
17                   i  <-  0
18                   FOREACH key  <-  isiCart
19                        IF key['id']= idbarang
20                             data  <-  Array('rowid' <- $key['rowid'],
21                             'qty' <- $key['qty'] + CALL input->post('jumlah_barang'))
22                             result  <-  CALL cart->update($data)
23                             IF result = TRUE
24                                 INCREMENT i
25                                 data['pesan']  <-  "BARANG BERHASIL DIPESAN"
26                             ELSE
27                                 data['pesan']  <-  "BARANG GAGAL DIPESAN"
28                             END IF
29                             CALL load->view('konsumen/suksesPesan.php',$data)
30                        END IF
31                   END FOREACH
32                   IF i = 0
33                        result  <-  CALL cart->insert($data)
34                        IF result = TRUE
35                             data['pesan']  <-  "BARANG BERHASIL DIPESAN"
36                        ELSE
37                             data['pesan']  <-  "BARANG GAGAL DIPESAN"
38                        END IF
39                        CALL load->view('konsumen/suksesPesan.php',$data)
40                   END IF
41              ELSE
42                   result  <-  CALL cart->insert($data)
43                   IF result = TRUE
44                        data['pesan']  <-  "BARANG BERHASIL DIPESAN"
45                   ELSE
46                        data['pesan']  <-  "BARANG GAGAL DIPESAN"
47                   END IF
48                   CALL load->view('konsumen/suksesPesan.php',$data)
49              END IF
50         ELSE
51              data['stok']  <-  "STOK BARANG TIDAK MENCUKUPI"
52              CALL load->view('konsumen/stokHabis.php',$data)
53         END IF
54    END IF
```

Gambar 5. 3 *Pseudocode method* tambahKeranjangBelanja()
Sumber: [Implementasi]

5.3.3. Pseudocode Method `detailPesanan()` dari Class Pesanan

Method `detailPesanan()` digunakan untuk menampilkan data barang yang dipesan. *Pseudocode* dari *method* `detailPesanan()` ditunjukkan dalam Gambar 5.4.

```
METHOD : detailPesanan

DECLARATION :
data, _SESSION, captcha IS Array
captcha IS Varchar

DESCRIPTION :
1     CALL cekSessionKonsumen()
2     data['cartDetail'] <- CALL cart->contents()
3     captcha <- CALL mcaptcha->generateCaptcha()
4     data['captcha'] <- captcha
5     _SESSION['captchaWord'] <- captcha['word']
6     CALL load->view('konsumen/detailPesanan.php',$data)
```

Gambar 5. 4 *Pseudocode method* detailPesanan()
Sumber: [Implementasi]

Penjelasan *pseudocode method* `detailPesanan()` pada Gambar 5.4 adalah sebagai berikut:

1. Baris 1 menjelaskan identifikasi konsumen yang *login*.
2. Baris 2 menjelaskan mengambil data dari *cart*.
3. Baris 3-5 menjelaskan pembuatan kode keamanan *captcha*.
4. Baris 6 menjelaskan pemanggilan halaman detailPesanan untuk menampilkan data dari *cart* dan *captcha*.

5.3.4. Pseudocode Method `checkOut()` dari Class Pesanan

Method `checkOut()` digunakan untuk menampilkan data transaksi. *Pseudocode* dari *method* `checkOut()` ditunjukkan dalam Gambar 5.5.

Penjelasan *pseudocode method* `checkOut()` pada Gambar 5.5 adalah sebagai berikut:

1. Baris 1 menjelaskan identifikasi konsumen yang *login*.
2. Baris 2 menjelaskan mengambil data dari *cart*.
3. Baris 3-5 menjelaskan untuk setiap jumlah barang yang dipesan pada *cart*.
4. Baris 6 menjelaskan pemeriksaan data konfirmasi *captcha* yang dimasukkan sama dengan data *captcha* pada *session* dan apakah pemeriksaan jumlah barang yang dipesan >= 0.
5. Baris 7-8 menjelaskan masukan data berupa data transaksi, idtransaksi, dan data keranjang belanja.

6. Baris 9-13 menjelaskan pemanggilan setiap stok barang pada *cart* untuk mendapatkan stokAkhir yang akan dimasukkan ke basis data.
7. Baris 14-19 menjelaskan apabila proses mengubah data stok barang dan menyimpan ke basis data berhasil (bernilai true), maka untuk setiap isi keranjang belanja disimpan ke basis data.
8. Baris 20-32 menjelaskan apabila transaksi benar maka memasukkan data konsumen dan data checkout ke basis data. Untuk masing-masing data checkOut yang ditampilkan kode_transaksi, waktu_pesan, nama_konsumen, alamat_konsumen, hp_konsumen, dan total.
9. Baris 33-35 menjelaskan pemeriksaan checkOut berhasil dilakukan (bernilai true), maka menghapus data keranjang belanja dan memanggil halaman checkout.
10. Baris 36-40 menjelaskan aliran alternatif baris 33-35, apabila checkout gagal dilakukan maka memanggil halaman pesan.
11. Baris 41-44 menjelaskan aliran alternatif baris 6, apabila konfirmasi kode *captcha* yang dimasukkan salah atau jumlah barang yang dicek tidak valid, maka memanggil halaman pesan.

```
METHOD : checkOut

DECLARATION :
transaksi, isiCart, key, dataBarang, checkOut, _SESSION, result, data IS Array
idtransaksi, idTransaksi, kodeTransaksi, namaKonsumen, alamatKonsumen, hpKonsumen, IS Varchar
waktuPesan IS Datetime
idKonsumen, sisaStok, qtyPesan, stokAkhir, total IS int

DESCRIPTION :
1     CALL cekSessionKonsumen()
2     isiCart  <-  CALL cart->contents()
3     FOREACH key <- isiCart
4           qtyPesan = key['qty']
5     END FOREACH
6     IF  (CALL input->post('confirmCaptcha')== _SESSION['captchaWord']) && (qtyPesan >= 0)
7           transaksi  <-  CALL pesanan_model->prosesPesanan()
8           idtransaksi  <-  CALL db->insert_id()
9           FOREACH key <- isiCart
10                sisaStok = CALL barang_model->getJumlahStok2($key['id'])
11                qtyPesan = key['qty']
12                stokAkhir = sisaStok-qtyPesan
13                result = CALL barang_model->ubahStok2($key['id'],$stokAkhir)
14                IF result = TRUE
15                      dataBarang  <-  Array('transaksi_idtransaksi' <- idtransaksi,
16                            'jumlah_barang' <- key['qty'], 'barang_idbarang' <- key['id']
17                      CALL pesanan_model->masukkanDetailPesanan($dataBarang)
18                END IF
19          END FOREACH
20          IF transaksi = TRUE
21                idKonsumen  <-  _SESSION['useridKonsumen']
22                checkOut  <-  CALL pesanan_model->getCheckOut($idKonsumen,$idtransaksi)
23                data['CheckOut']  <-  checkOut
24                FOREACH key <- checkOut
25                      idTransaksi  <-  key->idtransaksi
26                      kodeTransaksi  <-  key->kode_transaksi
27                      waktuPesan  <-  key->waktu_pesan
28                      namaKonsumen  <-  key->nama_konsumen
29                      alamatKonsumen  <-  key->alamat_konsumen
30                      hpKonsumen  <-  key->hp_konsumen
31                      total  <-  CALL cart->format_number(CALL cart->total())
32                END FOREACH
33                IF checkOut = TRUE
34                      CALL cart->destroy()
35                      CALL load->view('konsumen/checkOut.php',$data)
36                ELSE
37                      data['pesan']  <-  "MAAF, PROSES CHECK OUT YANG ANDA LAKUKAN GAGAL!"
38                      CALL load->view('konsumen/suksesPesan.php',$data)
39                END IF
40          END IF
41    ELSE
42          data['chaptcha']=" KODE KEAMANAN ATAU JUMLAH BARANG YANG DIPESAN TIDAK VALID!"
43          CALL load->view('konsumen/salahChaptcha.php',$data)
44    END IF
```

Gambar 5. 5 *Pseudocode method* checkOut()
Sumber: [Implementasi]

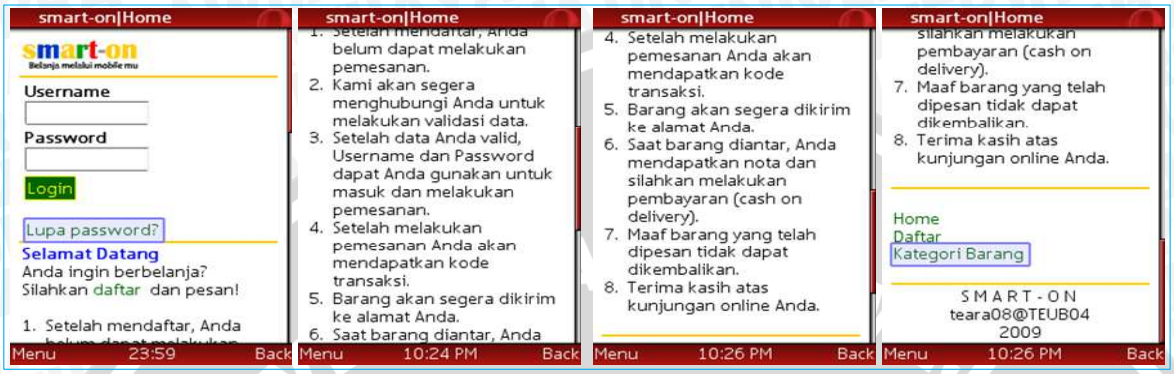
5.4. Implementasi Antarmuka

Pada subbab ini membahas semua implementasi yang didapat dari daftar kebutuhan fungsional. Implementasi antarmuka ini dibagi menjadi implementasi antarmuka user, administrator, operator, dan konsumen.

5.4.1. Implementasi Antarmuka *User* Yang Ditampilkan Melalui *Mobile Phone*

Pada implementasi antarmuka *user* ini ditampilkan melalui *mobile phone*. Tampilan disesuaikan dengan layar *mobile phone* sehingga *user* nyaman saat melihatnya.

5.4.1.1. Implementasi Antarmuka *Login*



Gambar 5. 6 Halaman *login* melalui *mobile phone*
Sumber: [Implementasi]

Gambar 5.6 menunjukkan tampilan halaman *login*. Halaman *login* yang juga merupakan halaman *index* ini menyediakan fasilitas *login* untuk autentifikasi konsumen agar dapat masuk dan menggunakan fasilitas sistem sebagai konsumen. Bagi *user* terdapat layanan informasi dan *link* untuk mendaftar sebagai konsumen.

5.4.1.2. Implementasi Antarmuka Registrasi

Halaman registrasi ini menyediakan fasilitas untuk mengisi *form* registrasi. *Form* registrasi yang harus diisi terdiri dari *field username, password,* ulangi *password,* nama lengkap, alamat lengkap, no. *handphone,* dan *email*. Gambar 5.7 menunjukkan tampilan halaman registrasi.

Gambar 5. 7 Halaman registrasi melalui *mobile phone*
Sumber: [Implementasi]

Setelah registrasi disimpan maka sistem akan memberikan konfirmasi registrasi yang dilakukan sukses, seperti yang ditunjukkan pada Gambar 5.8.

Gambar 5. 8 Halaman sukses registrasi melalui *mobile phone*
Sumber: [Implementasi]

5.4.1.3. Implementasi Antarmuka Daftar Kategori Barang

Halaman daftar kategori barang ini menampilkan daftar kategori barang, yang setiap kategori barang menjadi *link* untuk melihat daftar barang per kategori. Setiap halaman menampilkan 10 kategori barang. Gambar 5.9 menunjukkan tampilan halaman daftar kategori barang.

Gambar 5. 9 Halaman daftar kategori barang melalui *mobile phone*
Sumber: [Implementasi]

5.4.1.4. Implementasi Antarmuka Daftar Subkategori Barang

Halaman daftar subkategori barang ini menampilkan daftar subkategori per kategori dan daftar barang per kategori, yang setiap subkategori barang menjadi *link* untuk melihat daftar barang per subkategori. Setiap halaman menampilkan semua subkategori barang. Gambar 5.10 menunjukkan tampilan halaman daftar subkategori barang.

Gambar 5. 10 Halaman daftar subkategori barang melalui *mobile phone*
Sumber: [Implementasi]

5.4.1.5. Implementasi Antarmuka Melihat Daftar Barang

Gambar 5. 11 Halaman melihat daftar barang melalui *mobile phone*
Sumber: [Implementasi]

Gambar 5.11 ini menunjukkan tampilan halaman daftar barang. Halaman melihat daftar barang ini menampilkan daftar barang per kategori atau per subkategori. Setiap barang memiliki nama, kode, gambar, dan harga. Nama barang menjadi *link* untuk ke halaman data barang. Dan pada setiap halaman hanya menampilkan 5 barang

5.4.1.6. Implementasi Antarmuka Melihat Data Barang

Halaman melihat data barang ini menampilkan data barang yang terdiri dari nama, kode, gambar, harga, dan keterangan. Gambar barang yang ditampilkan ukurannya lebih besar dari ukuran gambar pada daftar barang. Gambar 5.12 ini menunjukkan gambar halaman daftar barang.

Gambar 5. 12 Halaman melihat data barang melalui *mobile phone*
Sumber: [Implementasi]

5.4.1.7. Implementasi Antarmuka Mencari Barang

Halaman mencari barang ini menyediakan fasilitas *form input* pencarian barang. *Keyword* yang digunakan untuk mencari adalah nama barang. Gambar 5.13 menunjukkan tampilan halaman hasil pencarian barang dan terdapat *form input* pencarian barang.

Gambar 5. 13 Halaman hasil pencarian barang melalui *mobile phone*
Sumber: [Implementasi]

Gambar 5. 14 Halaman konfirmasi *keyword* kosong melalui *mobile phone*
Sumber: [Implementasi]

Pada gambar 5.14 menunjukkan halaman konfirmasi *keyword* kosong. Halaman konfirmasi ini terjadi bila *user* tidak memasukkan suatu karakter sehingga *form input* pencarian barang kosong saat dijalankan (tombol "Cari" di-*klik*).

Pada gambar 5.15 menunjukkan halaman konfirmasi pencarian tidak ditemukan. Halaman konfirmasi ini terjadi bila nama barang yang dicari tidak ada pada basis data.

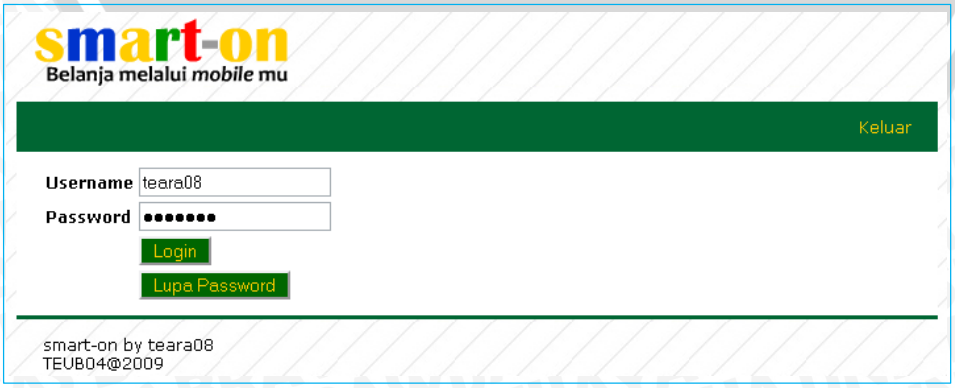
Gambar 5. 15 Halaman konfirmasi pencarian tidak ditemukan melalui *mobile phone*
Sumber: [Implementasi]

5.4.2. Implementasi Antarmuka Administrator Yang Ditampilkan Melalui PC

Pada implementasi antarmuka administrator ini ditampilkan melalui PC, karena menyesuaikan kebutuhan tabel dan *form* yang lebih besar sehingga membutuhkan layar yang lebih luas untuk melihatnya. Dan terdapat fasilitas untuk memudahkan meng-*upload* gambar barang.

5.4.2.1. Implementasi Antarmuka *Login*

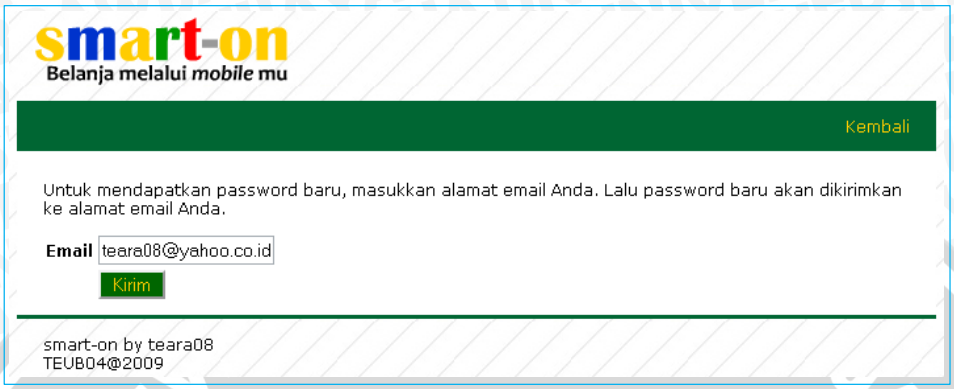
Halaman *login* menyediakan fasilitas untuk proses autentifikasi bagi *user* agar dapat masuk dan menggunakan fasilitas sistem sebagai administrator. Gambar 5.16 menunjukkan tampilan halaman *login*.



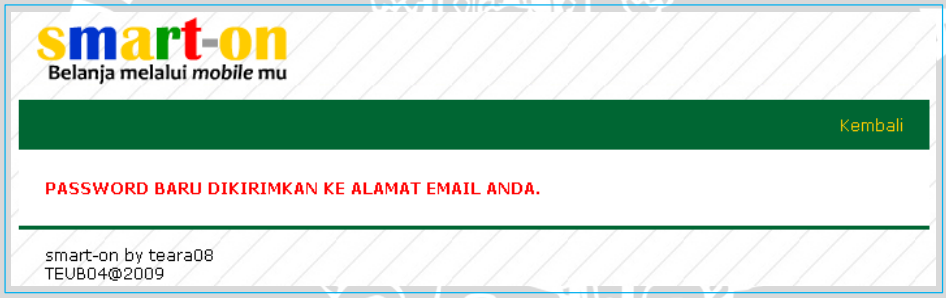
Gambar 5. 16 Halaman *login* melalui PC
Sumber: [Implementasi]

5.4.2.2. Implementasi Antarmuka Lupa *Password*

Halaman lupa *password* ini menyediakan fasilitas me-*request password* baru. Setelah administrator mengisi *form input email* dan meng-*klik* tombol "Kirim", sistem akan memproses dan mengirimkan *password* baru ke alamat *email* yang diisikan pada *form input email*. Gambar 5.17 menunjukkan tampilan halaman lupa *password*.



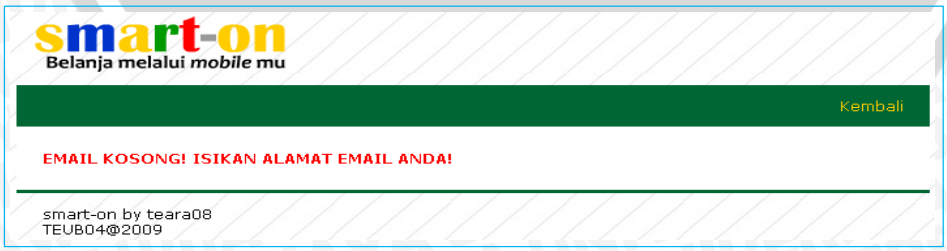
Gambar 5. 17 Halaman lupa *password* melalui PC
Sumber: [Implementasi]



Gambar 5. 18 Halaman sukses mengirim *password* melalui PC
Sumber: [Implementasi]

Pada gambar 5.18 menunjukkan tampilan halaman konfirmasi *password* baru telah berhasil dikirimkan ke alamat *email.*

Pada gambar 5.19 menunjukkan halaman konfirmasi *form input email* kosong. Halaman konfirmasi ini terjadi bila administrator tidak memasukkan suatu karakter sehingga *form input email* kosong saat dijalankan (tombol "Kirim" di-*klik*).



Gambar 5. 19 Halaman konfirmasi *form input email* kosong melalui PC
Sumber: [Implementasi]

Pada gambar 5.20 menunjukkan tampilan halaman konfirmasi *email* salah. Halaman konfirmasi ini terjadi bila alamat *email* yang diisikan tidak ada pada basis data.

Gambar 5. 20 Halaman konfirmasi *email* salah melalui PC
Sumber: [Implementasi]

5.4.2.3. Implementasi Antarmuka *Home* Administrator

Gambar 5. 21 Halaman *home* administrator melalui PC
Sumber: [Implementasi]

Gambar 5.21 menunjukkan tampilan halaman *home* administrator. Halaman *home* administrator ini berisi *link-link* untuk mengatur seluruh kebutuhan administrator.

5.4.2.4. Implementasi Antarmuka Mengubah *Id Login* Administrator

Halaman mengubah *id login* administrator menyediakan fasilitas untuk mengubah *username* dan *password* administrator. Halaman ini berisi *form input username*, *password* baru, dan ulangi *password*. Halaman mengubah *id login* administrator ditunjukkan pada Gambar 5.22.

Gambar 5. 22 Halaman mengubah *id login* administrator melalui PC
Sumber: [Implementasi]

Jika *username* dan *password* berhasil diubah dan disimpan di dalam basis data maka sistem akan menampilkan konfirmasi *username* dan *password* barhasil diubah. Gambar 5.23 menunjukkan tampilan halaman sukses mengubah *id login* administrator.

Gambar 5. 23 Halaman sukses mengubah *id login* administrator melalui PC
Sumber: [Implementasi]

5.4.2.5. Implementasi Antarmuka Melihat Daftar Barang

Halaman melihat daftar barang ini menampilkan daftar barang yang terdiri dari no, kode, nama, harga (Rp.), dan aksi. Pada setiap halaman diberi batas, daftar barang yang ditampilkan 10 barang. Kolom aksi berisi *link* ubah, stok, dan hapus. Gambar 5.24 ini menunjukkan tampilan halaman melihat daftar barang.

Gambar 5. 24 Halaman melihat daftar barang melalui PC
Sumber: [Implementasi]

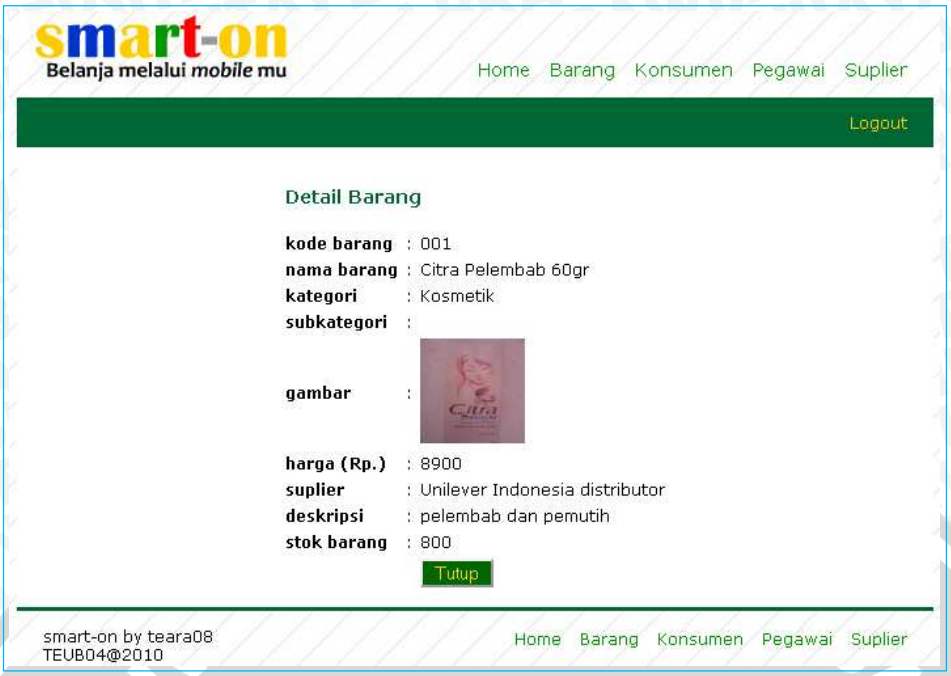
5.4.2.6. Implementasi Antarmuka Menghapus Barang

Pada Gambar 5.25 menunjukkan tampilan halaman konfirmasi barang berhasil dihapus.

Gambar 5. 25 Halaman menghapus barang melalui PC
Sumber: [Implementasi]

5.4.2.7. Implementasi Antarmuka Melihat Data Barang

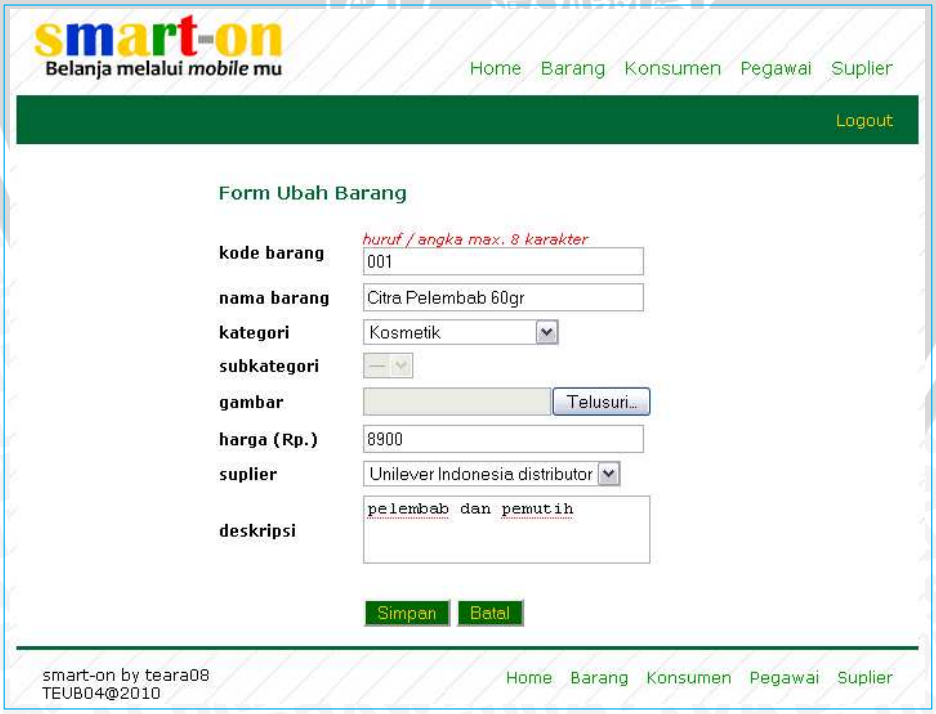
Halaman melihat data barang ini menampilkan data barang yang terdiri dari kode barang, nama barang, kategori, subkategori, gambar, harga, suplier, dan deskripsi. Gambar 5.26 ini menunjukkan gambar halaman data barang.



Gambar 5. 26 Halaman melihat data barang melalui PC
Sumber: [Implementasi]

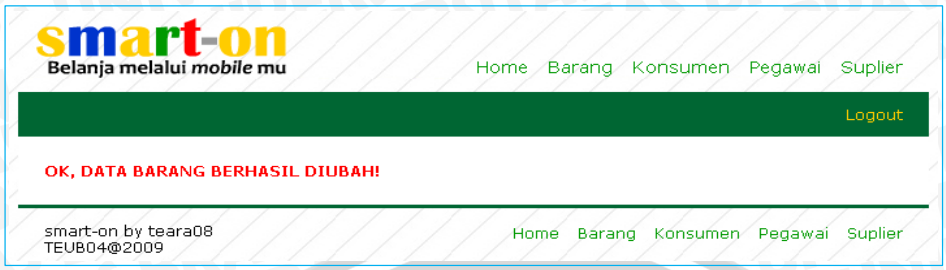
5.4.2.8. Implementasi Antarmuka Mengubah Data Barang

Halaman mengubah data barang menyediakan fasilitas untuk administrator mengubah data barang. Data yang dapat diubah adalah kode barang, nama barang, kategori, subkategori, gambar, harga, suplier, dan deskripsi. Gambar 5.27 menunjukkan tampilan halaman mengubah data barang.



Gambar 5. 27 Halaman mengubah data barang melalui PC
Sumber: [Implementasi]

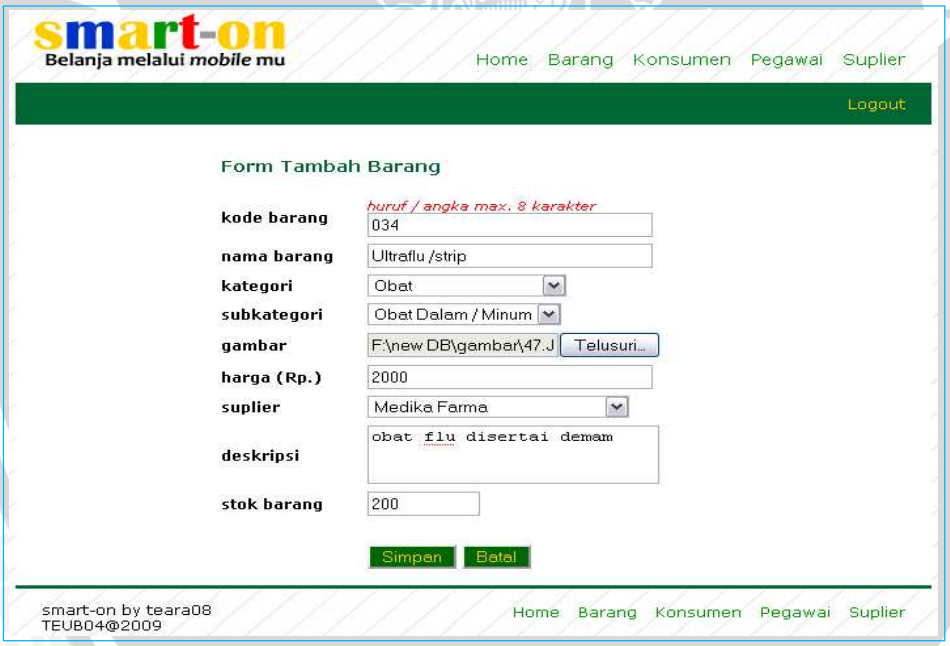
Pada Gambar 5.28 menunjukkan tampilan halaman konfirmasi data barang telah berhasil diubah dan disimpan ke basis data.



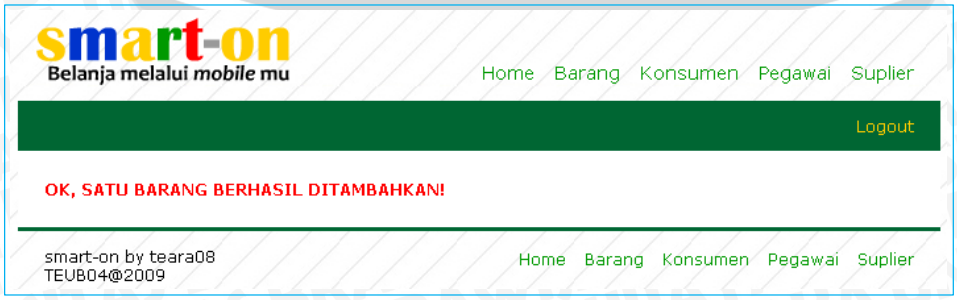
Gambar 5. 28 Halaman sukses mengubah data barang melalui PC
Sumber: [Implementasi]

5.4.2.9. Implementasi Antarmuka Menambah Barang

Halaman menambah barang menyediakan fasilitas untuk administrator menambah barang. Data yang diisikan adalah kode barang, nama barang, kategori, subkategori, gambar, harga, suplier, deskripsi, dan stok barang. Gambar 5.29 menunjukkan tampilan halaman menambah barang.



Gambar 5. 29 Halaman menambah barang melalui PC
Sumber: [Implementasi]



Gambar 5. 30 Halaman sukses menambah barang melalui PC
Sumber: [Implementasi]

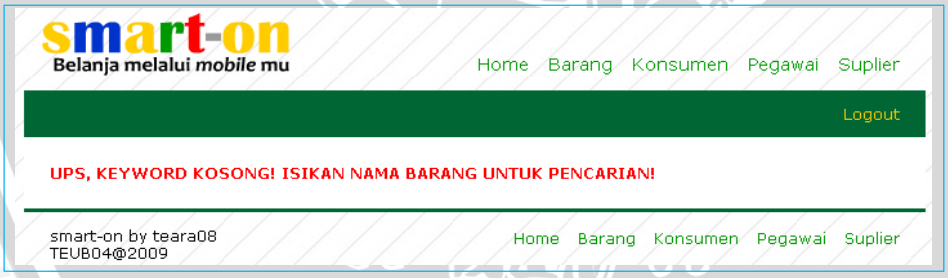
Pada Gambar 5.30 menunjukkan tampilan halaman konfirmasi barang telah berhasil ditambah dan disimpan ke basis data.

5.4.2.10. Implementasi Antarmuka Mencari Barang

Halaman mencari barang menyediakan fasilitas bagi administrator untuk mencari barang dengan mengisi *form input* pencarian menggunakan *keyword* nama barang. Gambar 5.31 menunjukkan tampilan hasil pencarian barang dan *form input* pencarian.

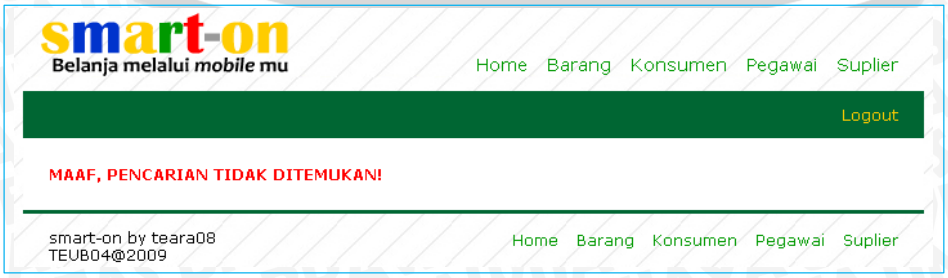
Gambar 5. 31 Halaman hasil pencarian barang melalui PC
Sumber: [Implementasi]

Pada gambar 5.32 menunjukkan halaman konfirmasi *keyword* kosong. Halaman konfirmasi ini terjadi bila administrator tidak memasukkan suatu karakter sehingga *form input* pencarian barang kosong saat dijalankan (tombol "Cari" di-*klik*).



Gambar 5. 32 Halaman konfirmasi *keyword* kosong melalui PC
Sumber: [Implementasi]

Pada gambar 5.33 menunjukkan halaman konfirmasi pencarian tidak ditemukan. Halaman konfirmasi ini terjadi bila nama barang yang dicari tidak ada pada basis data.

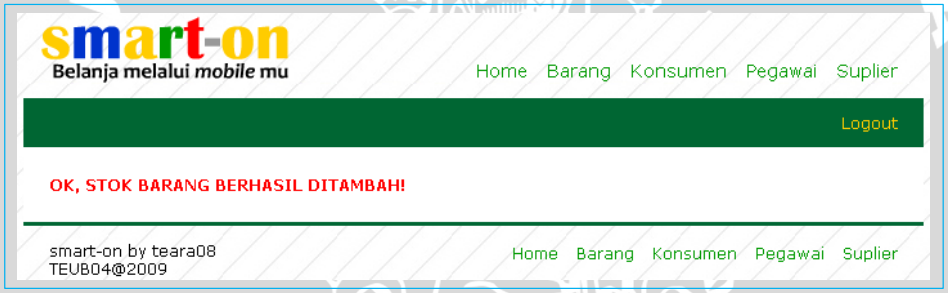


Gambar 5. 33 Halaman konfirmasi pencarian tidak ditemukan melalui PC
Sumber: [Implementasi]

5.4.2.11. Implementasi Antarmuka Menambah Stok Barang

Halaman menambah stok barang menyediakan fasilitas untuk administrator menambah stok barang. Data yang diisikan adalah stok tambah. Gambar 5.34 menunjukkan tampilan halaman menambah stok barang.

Gambar 5. 34 Halaman menambah stok barang melalui PC
Sumber: [Implementasi]



Gambar 5. 35 Halaman sukses menambah stok barang melalui PC
Sumber: [Implementasi]

Pada Gambar 5.35 menunjukkan tampilan halaman konfirmasi stok barang telah berhasil ditambah dan disimpan ke basis data.

5.4.2.12. Implementasi Antarmuka Melihat Daftar Kategori Barang

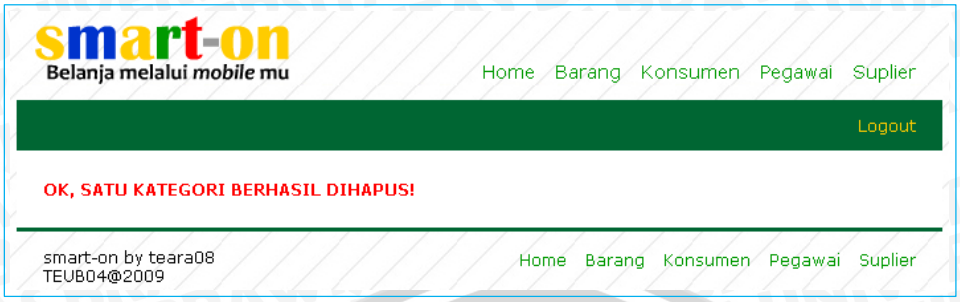
Halaman melihat daftar kategori barang ini menampilkan daftar kategori barang yang terdiri dari no, kode, nama, harga, dan aksi. Pada setiap halaman diberi batas, daftar kategori barang yang ditampilkan 10 kategori barang. Kolom aksi berisi *link* ubah dan hapus. Gambar 5.36 ini menunjukkan gambar halaman melihat daftar kategori barang.

Gambar 5. 36 Halaman melihat daftar kategori barang melalui PC
Sumber: [Implementasi]

Gambar 5. 37 Halaman melihat daftar barang per kategori dan per subkategori melalui PC
Sumber: [Implementasi]

Gambar 5.37 menunjukkan tampilan halaman daftar barang per kategori dan per subkategori. Halaman ini menampilkan tabel yang berisi no, kode, nama, harga (Rp.), dan aksi. Aksi berisi *link* ubah, stok, dan hapus.

5.4.2.13. Implementasi Antarmuka Menghapus Kategori Barang



Gambar 5. 38 Halaman menghapus kategori barang melalui PC
Sumber: [Implementasi]

Pada Gambar 5.38 menunjukkan halaman konfirmasi kategori barang berhasil dihapus.

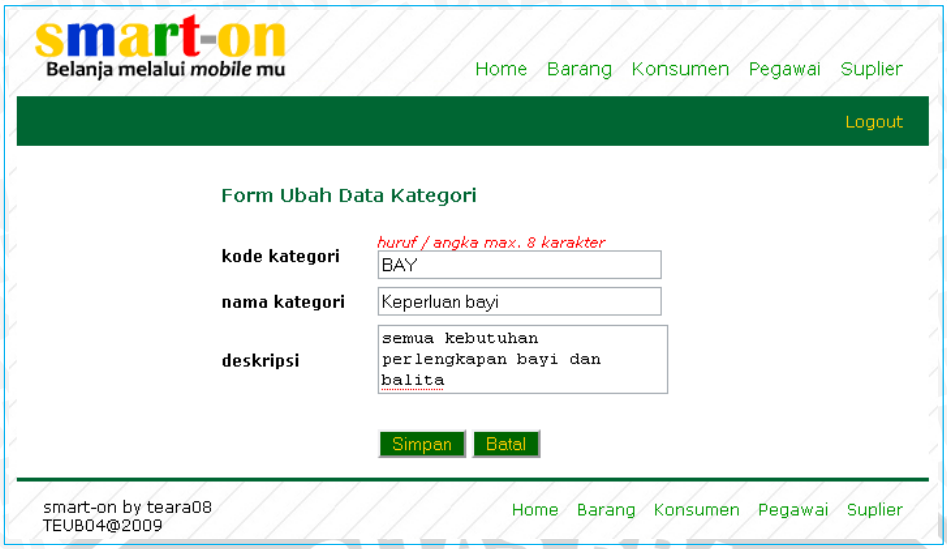
5.4.2.14. Implementasi Antarmuka Melihat Data Kategori Barang

Halaman melihat data kategori barang ini menampilkan data kategori barang yang terdiri dari kode kategori, nama kategori, dan deskripsi. Gambar 5.39 ini menunjukkan tampilan halaman data kategori barang.

Gambar 5. 39 Halaman melihat data kategori melalui PC
Sumber: [Implementasi]

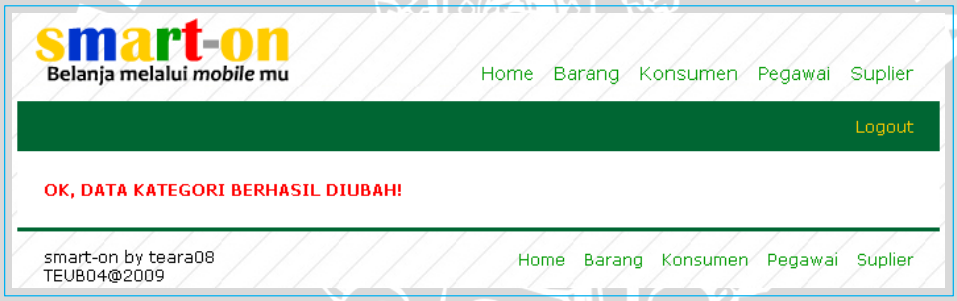
5.4.2.15. Implementasi Antarmuka Mengubah Data Kategori Barang

Halaman mengubah data kategori barang menyediakan fasilitas untuk administrator mengubah data kategori barang. Data yang dapat diubah adalah kode kategori, nama kategori, dan deskripsi. Gambar 5.40 menunjukkan tampilan halaman mengubah data kategori barang.



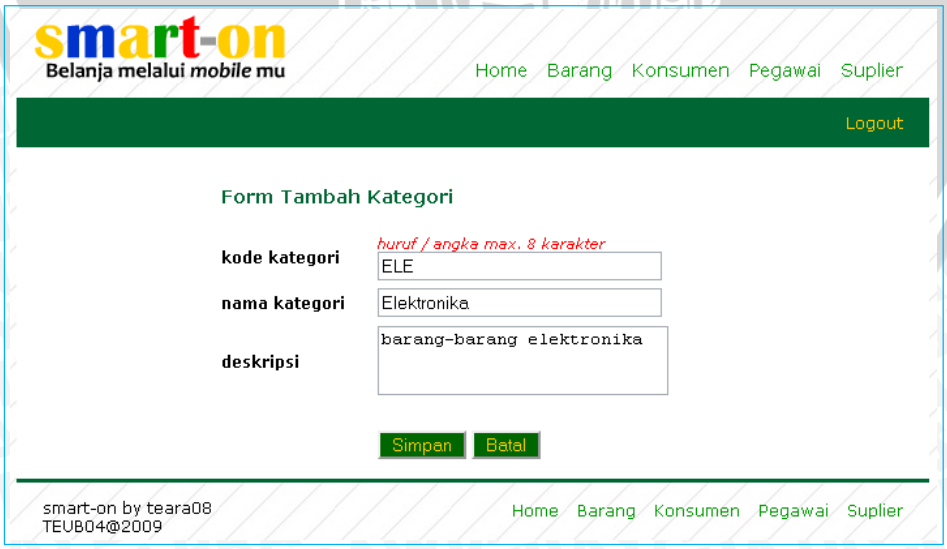
Gambar 5. 40 Halaman mengubah data kategori barang melalui PC
Sumber: [Implementasi]

Pada Gambar 5.41 menunjukkan tampilan halaman konfirmasi data kategori barang telah berhasil diubah dan disimpan ke basis data.



Gambar 5. 41 Halaman sukses mengubah data kategori barang melalui PC
Sumber: [Implementasi]

5.4.2.16. Implementasi Antarmuka Menambah Kategori Barang



Gambar 5. 42 Halaman menambah kategori barang melalui PC
Sumber: [Implementasi]

Gambar 5.42 menunjukkan tampilan halaman menambah kategori barang. Halaman menambah kategori barang menyediakan fasilitas untuk administrator menambah kategori barang. Data yang diisikan adalah kode kategori, nama kategori, dan deskripsi.

Pada Gambar 5.43 menunjukkan tampilan halaman konfirmasi kategori barang telah berhasil ditambah dan disimpan ke basis data.

Gambar 5. 43 Halaman sukses menambah kategori barang melalui PC
Sumber: [Implementasi]

5.4.2.17. Implementasi Antarmuka Mencari Kategori Barang

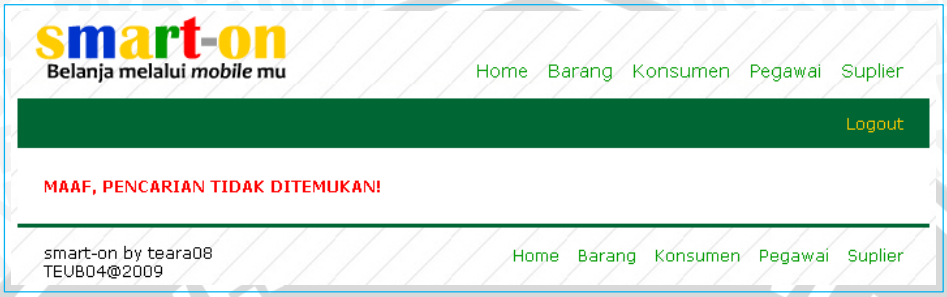
Halaman mencari kategori barang menyediakan fasilitas bagi administrator untuk mencari kategori barang dengan mengisi *form input* pencarian menggunakan *keyword* nama kategori barang. Gambar 5.44 menunjukkan tampilan hasil pencarian kategori barang dan *form input* pencarian.

Gambar 5. 44 Halaman hasil pencarian kategori barang melalui PC
Sumber: [Implementasi]

Gambar 5. 45 Halaman konfirmasi *keyword* kosong melalui PC
Sumber: [Implementasi]

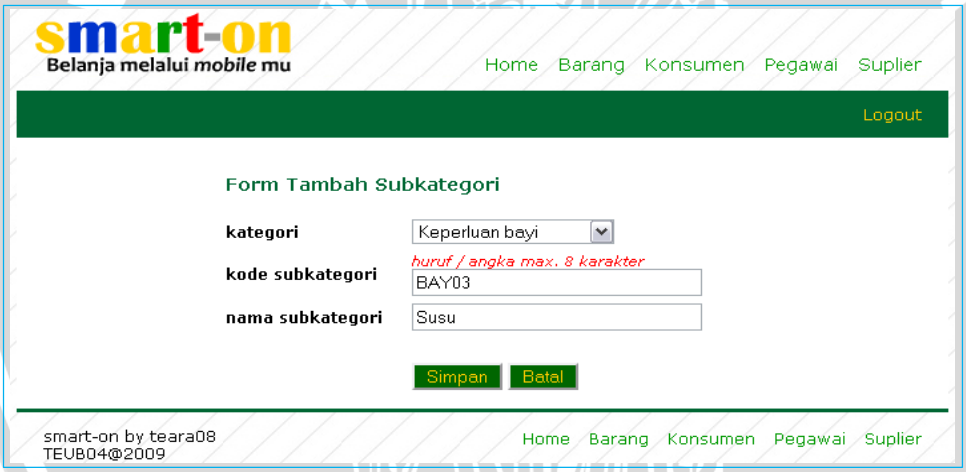
Pada gambar 5.45 menunjukkan halaman konfirmasi *keyword* kosong. Halaman konfirmasi ini terjadi bila administrator tidak memasukkan suatu karakter sehingga *form input* pencarian kategori barang kosong saat dijalankan (tombol "Cari" di-*klik*).

Pada gambar 5.46 menunjukkan tampilan halaman konfirmasi pencarian tidak ditemukan. Halaman konfirmasi ini terjadi bila nama kategori barang yang dicari tidak ada pada basis data.



Gambar 5. 46 Halaman konfirmasi pencarian tidak ditemukan melalui PC
Sumber: [Implementasi]

5.4.2.18. Implementasi Antarmuka Menambah Subkategori Barang



Gambar 5. 47 Halaman menambah subkategori barang melalui PC
Sumber: [Implementasi]

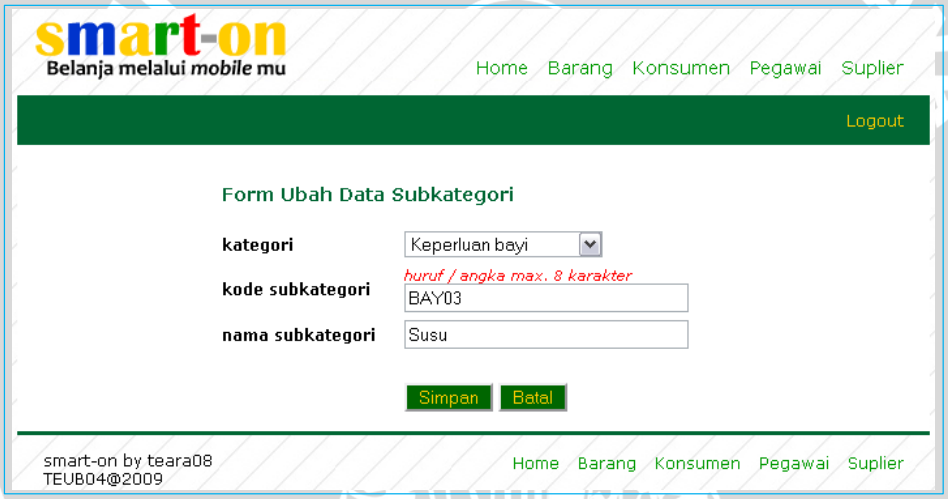
Gambar 5.47 menunjukkan tampilan halaman menambah subkategori barang. Halaman menambah subkategori barang menyediakan fasilitas untuk administrator menambah subkategori barang. Data yang diisikan adalah nama kategori, kode subkategori, dan nama subkategori.

Pada Gambar 5.48 menunjukkan tampilan halaman konfirmasi subkategori barang telah berhasil ditambah dan disimpan ke basis data.

Gambar 5. 48 Halaman sukses menambah subkategori barang melalui PC
Sumber: [Implementasi]

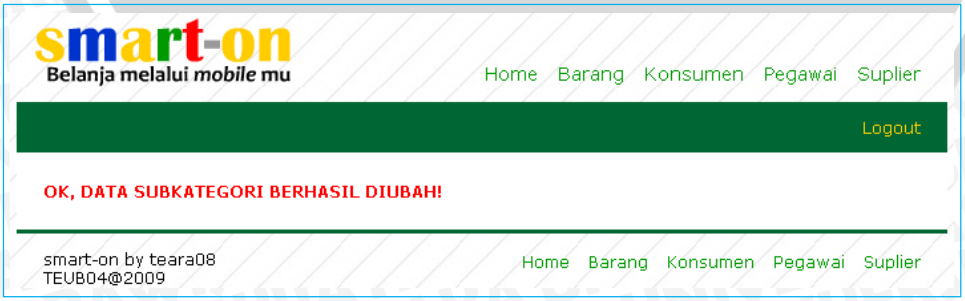
5.4.2.19. Implementasi Antarmuka Mengubah Data Subkategori Barang

Halaman mengubah data subkategori barang menyediakan fasilitas untuk administrator mengubah data subkategori barang. Data yang dapat diubah adalah nama kategori, kode subkategori, dan nama subkategori. Gambar 5.49 menunjukkan tampilan halaman mengubah data subkategori barang.



Gambar 5. 49 Halaman mengubah data subkategori barang melalui PC
Sumber: [Implementasi]

Pada Gambar 5.50 menunjukkan tampilan halaman konfirmasi data subkategori barang telah berhasil diubah dan disimpan ke basis data.



Gambar 5. 50 Halaman sukses mengubah data subkategori barang melalui PC
Sumber: [Implementasi]

5.4.2.20. Implementasi Antarmuka Menghapus Subkategori Barang

Gambar 5. 51 Halaman menghapus subkategori barang melalui PC
Sumber: [Implementasi]

Pada Gambar 5.51 menunjukkan halaman konfirmasi subkategori barang berhasil dihapus.

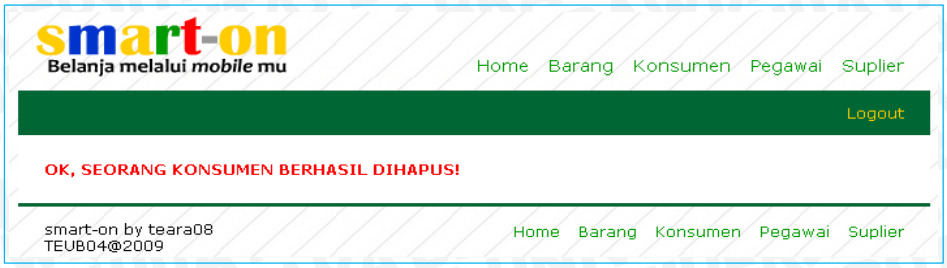
5.4.2.21. Implementasi Antarmuka Melihat Daftar Konsumen

Halaman melihat daftar konsumen ini menampilkan daftar konsumen yang terdiri dari no, nama, status validasi, dan aksi. Pada setiap halaman diberi batas, daftar konsumen yang ditampilkan 10 konsumen. Kolom status validasi berisi status validasi yang merupakan *link* yang menuju ke halaman melihat data konsumen. Kolom aksi berisi *link* ubah dan hapus. Gambar 5.52 ini menunjukkan tampilan halaman melihat daftar konsumen.

Gambar 5. 52 Halaman melihat daftar konsumen melalui PC
Sumber: [Implementasi]

5.4.2.22. Implementasi Antarmuka Menghapus Konsumen

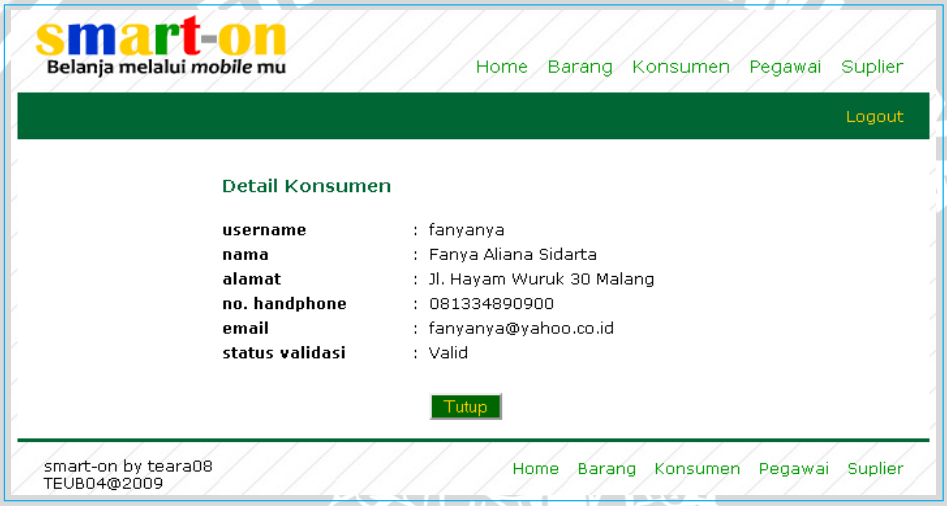
Pada Gambar 5.53 menunjukkan tampilan halaman konfirmasi konsumen berhasil dihapus.



Gambar 5. 53 Halaman menghapus konsumen melalui PC
Sumber: [Implementasi]

5.4.2.23. Implementasi Antarmuka Melihat Data Konsumen

Halaman melihat data konsumen ini menampilkan data konsumen yang terdiri dari *username*, nama, alamat, no. *handphone*, *email*, dan status validasi. Gambar 5.54 ini menunjukkan tampilan halaman data konsumen.



Gambar 5. 54 Halaman melihat data konsumen melalui PC
Sumber: [Implementasi]

5.4.2.24. Implementasi Antarmuka Mencari Konsumen

Gambar 5. 55 Halaman hasil pencarian konsumen melalui PC
Sumber: [Implementasi]

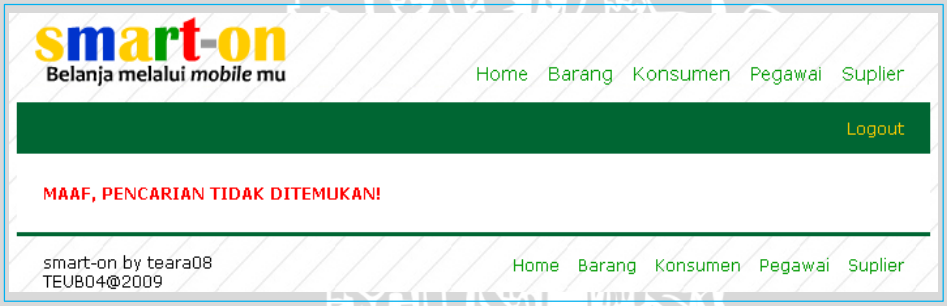
Gambar 5.55 menunjukkan tampilan hasil pencarian konsumen dan *form input* pencarian. Halaman mencari konsumen menyediakan fasilitas bagi administrator untuk

mencari konsumen dengan mengisi *form input* pencarian menggunakan *keyword* status validasi.

Pada gambar 5.56 menunjukkan tampilan halaman konfirmasi *keyword* kosong. Halaman konfirmasi ini terjadi bila administrator tidak memasukkan suatu karakter sehingga *form input* pencarian konsumen kosong saat dijalankan (tombol "Cari" di-*klik*).

Gambar 5. 56 Halaman konfirmasi *keyword* kosong melalui PC
Sumber: [Implementasi]

Pada gambar 5.57 menunjukkan tampilan halaman konfirmasi pencarian tidak ditemukan. Halaman konfirmasi ini terjadi bila nama konsumen yang dicari tidak ada pada basis data.



Gambar 5. 57 Halaman konfirmasi pencarian tidak ditemukan melalui PC
Sumber: [Implementasi]

5.4.2.25. Implementasi Antarmuka Melihat Daftar Pegawai

Gambar 5. 58 Halaman melihat daftar pegawai melalui PC
Sumber: [Implementasi]

Pada Gambar 5.58 menunjukkan tampilan halaman melihat daftar pegawai. Halaman melihat daftar pegawai ini menampilkan daftar pegawai yang terdiri dari no, kode, nama, tugas, dan aksi. Pada setiap halaman diberi batas, daftar pegawai yang ditampilkan 10 pegawai. Kolom kode berisi kode pegawai yang merupakan *link* yang menuju ke halaman melihat data pegawai. Kolom aksi berisi *link* ubah dan hapus.

5.4.2.26. Implementasi Antarmuka Menghapus Pegawai

Pada Gambar 5.59 menunjukkan tampilan halaman konfirmasi pegawai berhasil dihapus.

Gambar 5. 59 Halaman menghapus pegawai melalui PC
Sumber: [Implementasi]

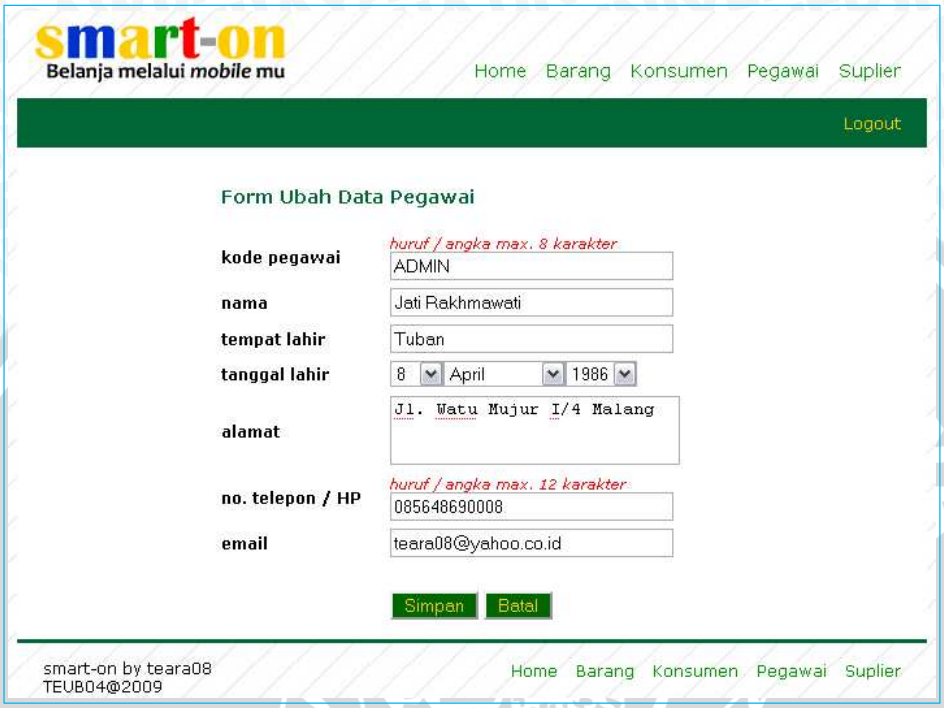
5.4.2.27. Implementasi Antarmuka Melihat Data Pegawai

Gambar 5. 60 Halaman melihat detail pegawai melalui PC
Sumber: [Implementasi]

Gambar 5.60 ini menunjukkan tampilan halaman data pegawai. Halaman melihat data pegawai ini menampilkan data pegawai yang terdiri dari kode pegawai, *username*, nama lengkap, *sex*, tempat lahir, tanggal lahir, alamat, no. telepon/HP, *email*, dan tugas.

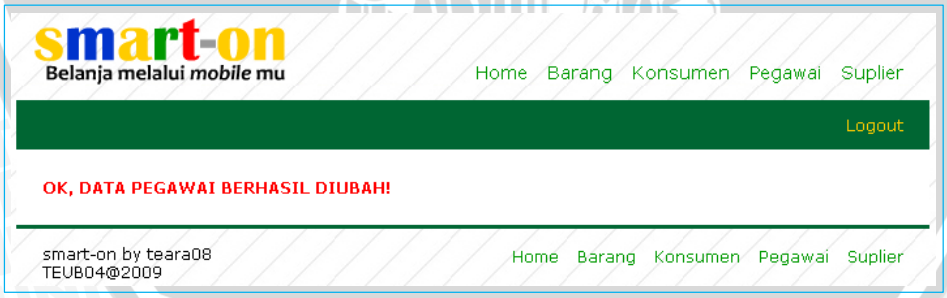
5.4.2.28. Implementasi Antarmuka Mengubah Data Pegawai

Halaman mengubah data pegawai menyediakan fasilitas untuk administrator mengubah data pegawai. Data yang dapat diubah adalah kode pegawai, nama, tempat lahir, tanggal lahir, alamat, no. telepon/HP, dan *email*. Gambar 5.61 menunjukkan tampilan halaman mengubah data pegawai.



Gambar 5. 61 Halaman mengubah data pegawai melalui PC
Sumber: [Implementasi]

Pada Gambar 5.62 menunjukkan tampilan halaman konfirmasi data pegawai telah berhasil diubah dan disimpan ke basis data.



Gambar 5. 62 Halaman sukses mengubah data pegawai melalui PC
Sumber: [Implementasi]

5.4.2.29. Implementasi Antarmuka Menambah Pegawai

Halaman menambah pegawai menyediakan fasilitas untuk administrator menambah pegawai. Data yang diisikan adalah kode pegawai, *username*, *password*, ulangi *password*, nama lengkap, *sex*, tempat lahir, tanggal lahir, alamat, no. telepon/HP, *email*, dan tugas. Gambar 5.63 menunjukkan tampilan halaman menambah pegawai.

Gambar 5. 63 Halaman menambah pegawai melalui PC
Sumber: [Implementasi]

Pada Gambar 5.64 menunjukkan tampilan halaman konfirmasi pegawai telah berhasil ditambah dan disimpan ke basis data.

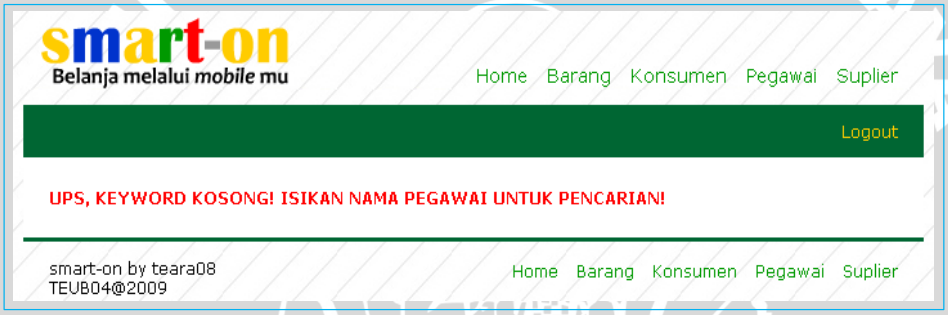
Gambar 5. 64 Halaman sukses menambah pegawai melalui PC
Sumber: [Implementasi]

5.4.2.30. Implementasi Antarmuka Mencari Pegawai

Halaman mencari pegawai menyediakan fasilitas bagi administrator untuk mencari pegawai dengan mengisi *form input* pencarian menggunakan *keyword* nama pegawai. Gambar 5.65 menunjukkan tampilan hasil pencarian pegawai dan *form input* pencarian.

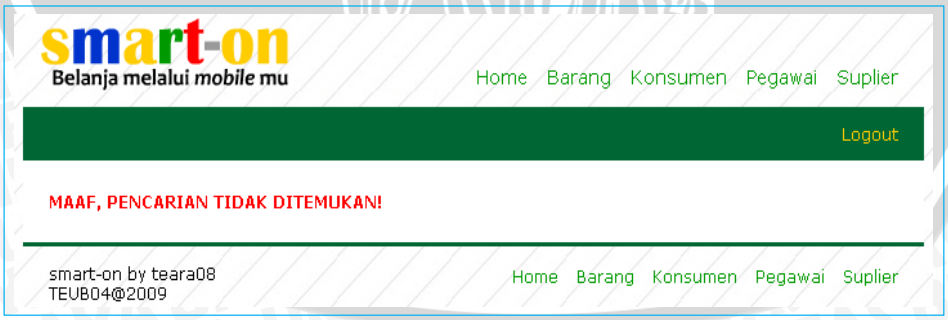
Gambar 5. 65 Halaman hasil pencarian pegawai melalui PC
Sumber: [Implementasi]

Pada gambar 5.66 menunjukkan tampilan halaman konfirmasi *keyword* kosong. Halaman konfirmasi ini terjadi bila administrator tidak memasukkan suatu karakter sehingga *form input* pencarian pegawai kosong saat dijalankan (tombol "Cari" di-*klik*).



Gambar 5. 66 Halaman konfirmasi *keyword* kosong melalui PC
Sumber: [Implementasi]

Pada gambar 5.67 menunjukkan tampilan halaman konfirmasi pencarian tidak ditemukan. Halaman konfirmasi ini terjadi bila nama pegawai yang dicari tidak ada pada basis data.



Gambar 5. 67 Halaman konfirmasi pencarian tidak ditemukan melalui PC
Sumber: [Implementasi]

5.4.2.31. Implementasi Antarmuka Melihat Daftar Suplier

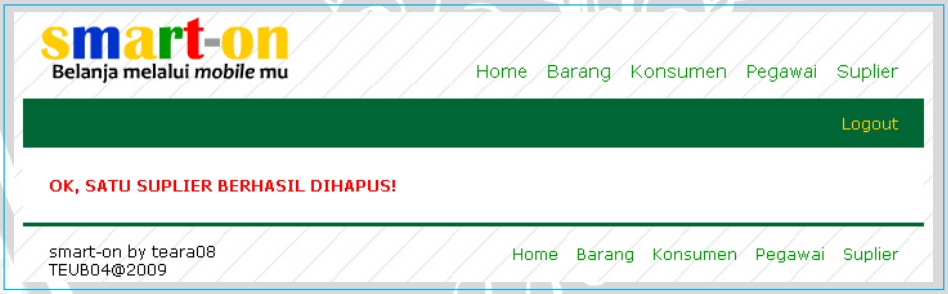
Halaman melihat daftar suplier ini menampilkan daftar suplier yang terdiri dari no, kode, nama, no. telp, dan aksi. Pada setiap halaman diberi batas, daftar suplier yang

ditampilkan 10 suplier. Kolom kode berisi kode suplier yang merupakan *link* yang menuju ke halaman melihat data suplier. Kolom aksi berisi *link* ubah dan hapus. Gambar 5.68 ini menunjukkan gambar halaman melihat daftar suplier.

Gambar 5. 68 Halaman melihat daftar suplier melalui PC
Sumber: [Implementasi]

5.4.2.32. Implementasi Antarmuka Sukses Menghapus Suplier

Pada Gambar 5.69 menunjukkan tampilan halaman konfirmasi suplier berhasil dihapus dari daftar dan basis data.



Gambar 5. 69 Halaman sukses menghapus suplier melalui PC
Sumber: [Implementasi]

5.4.2.33. Implementasi Antarmuka Melihat Data Suplier

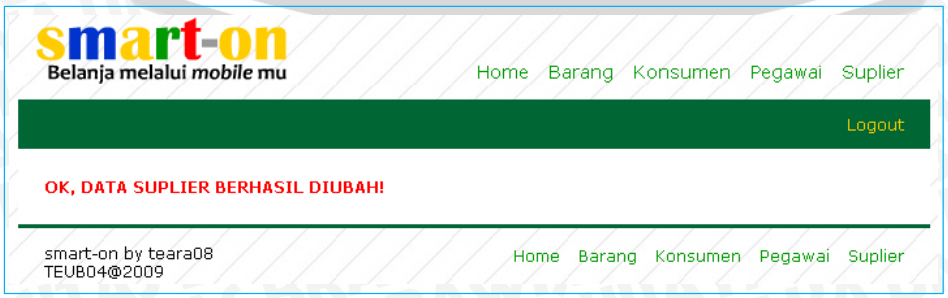
Halaman melihat data suplier ini menampilkan data suplier yang terdiri dari kode suplier, nama suplier, alamat, dan no. telp. Gambar 5.70 ini menunjukkan tampilan halaman data suplier.

Gambar 5. 70 Halaman melihat detail suplier melalui PC
Sumber: [Implementasi]

5.4.2.34. Implementasi Antarmuka Mengubah Data Suplier

Halaman mengubah data suplier menyediakan fasilitas untuk administrator mengubah data suplier. Data yang dapat diubah adalah kode suplier, nama suplier, alamat, dan no. telp. Gambar 5.71 menunjukkan tampilan halaman mengubah data suplier.

Gambar 5. 71 Halaman mengubah data suplier melalui PC
Sumber: [Implementasi]



Gambar 5. 72 Halaman sukses mengubah data suplier melalui PC
Sumber: [Implementasi]

Pada Gambar 5.72 menunjukkan tampilan halaman konfirmasi data suplier telah berhasil diubah dan disimpan ke basis data.

5.4.2.35. Implementasi Antarmuka Menambah Suplier

Halaman menambah suplier menyediakan fasilitas untuk administrator menambah suplier. Data yang diisikan adalah kode suplier, nama suplier, alamat, dan no. telp. Gambar 5.73 menunjukkan tampilan halaman menambah suplier.

Gambar 5. 73 Halaman menambah suplier melalui PC
Sumber: [Implementasi]

Pada Gambar 5.74 menunjukkan tampilan halaman konfirmasi suplier telah berhasil ditambah dan disimpan ke basis data.

Gambar 5. 74 Halaman sukses menambah suplier melalui PC
Sumber: [Implementasi]

5.4.2.36. Implementasi Antarmuka Mencari Suplier

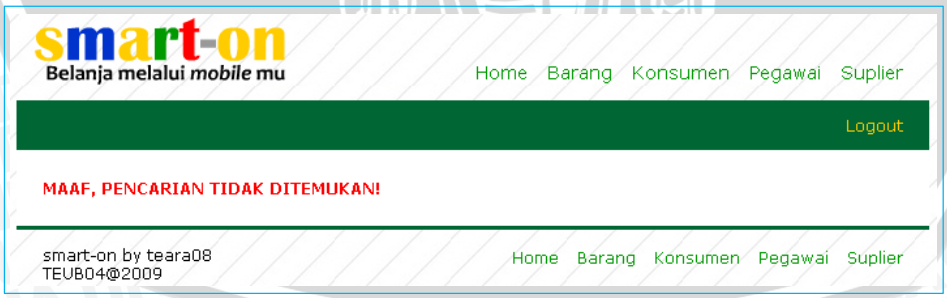
Halaman mencari suplier menyediakan fasilitas bagi administrator untuk mencari suplier dengan mengisi *form input* pencarian menggunakan *keyword* nama suplier. Gambar 5.75 menunjukkan tampilan halaman hasil pencarian suplier dan *form input* pencarian.

Gambar 5. 75 Halaman hasil pencarian suplier melalui PC
Sumber: [Implementasi]

Pada gambar 5.76 menunjukkan tampilan halaman konfirmasi *keyword* kosong. Halaman konfirmasi ini terjadi bila administrator tidak memasukkan suatu karakter sehingga *form input* pencarian suplier kosong saat dijalankan (tombol "Cari" di-*klik*).

Gambar 5. 76 Halaman konfirmasi *keyword* kosong melalui PC
Sumber: [Implementasi]

Pada gambar 5.77 menunjukkan tampilan halaman konfirmasi pencarian tidak ditemukan. Halaman konfirmasi ini terjadi bila nama suplier yang dicari tidak ada pada basis data.



Gambar 5. 77 Halaman konfirmasi pencarian tidak ditemukan melalui PC
Sumber: [Implementasi]

5.4.3. Implementasi Antarmuka Operator Yang Ditampilkan Melalui PC

Pada implementasi antarmuka operator ini sama dengan implementasi antarmuka pada administrator. Dan terdapat fasilitas yang mempermudah proses mencetak nota dengan PC yang terkoneksi dengan *printer*.

5.4.3.1. Implementasi Antarmuka *Home* Operator

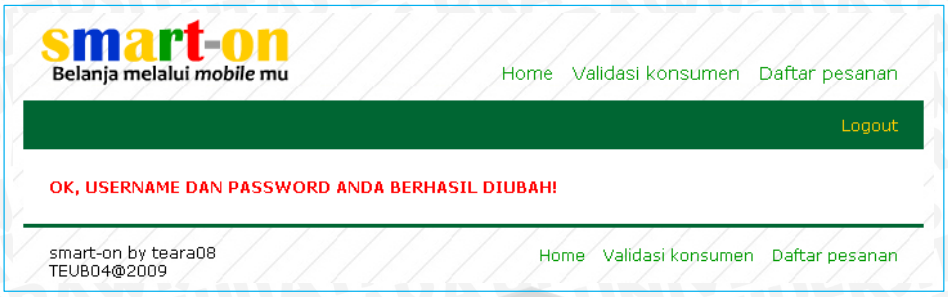
Setelah melakukan *login* pada halaman *login* yang sama pada Gambar 5.16 dan masuk serta menggunakan fasilitas sistem sebagai operator. Halaman *home* operator ini berisi *link-link* untuk mengatur seluruh kebutuhan operator. Gambar 5.78 menunjukkan tampilan halaman *home* operator.

Gambar 5. 78 Halaman *home* operator melalui PC
Sumber: [Implementasi]

5.4.3.2. Implementasi Antarmuka Mengubah *Id Login* Operator

Gambar 5. 79 Halaman mengubah *id login* operator melalui PC
Sumber: [Implementasi]

Halaman mengubah *id login* operator ditunjukkan pada Gambar 5.79. Halaman mengubah *id login* operator menyediakan fasilitas untuk mengubah *username* dan *password* operator. Halaman ini berisi *form input username*, *password* baru, dan ulangi *password*.



Gambar 5. 80 Halaman sukses mengubah *id login* operator melalui PC
Sumber: [Implementasi]

Gambar 5.80 menunjukkan tampilan halaman sukses mengubah *id login* operator. Jika *username* dan *password* berhasil diubah dan disimpan di dalam basis data maka sistem akan menampilkan konfirmasi *username* dan *password* barhasil diubah.

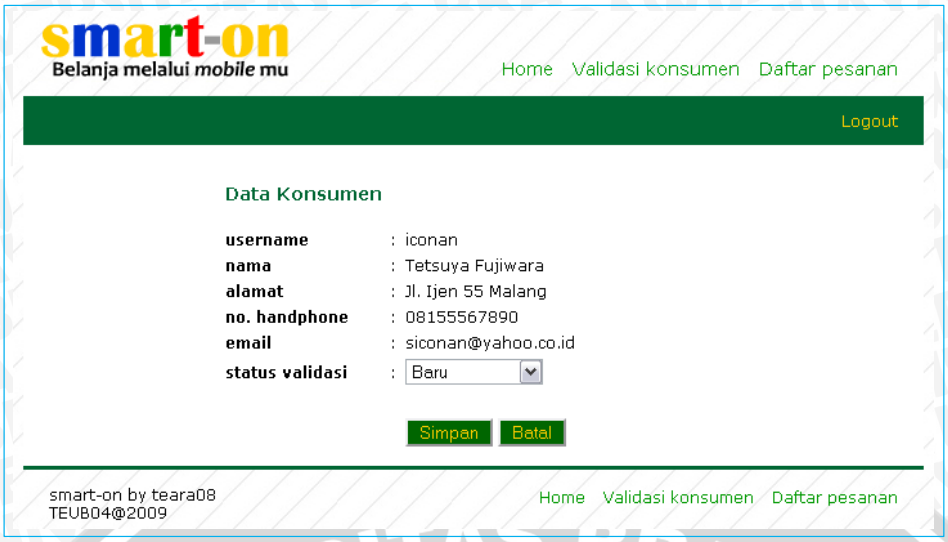
5.4.3.3. Implementasi Antarmuka Melihat Daftar Validasi Konsumen

Halaman melihat daftar validasi konsumen ini menampilkan daftar validasi konsumen yang terdiri dari no, nama, no. *handphone*, dan status validasi. Pada setiap halaman diberi batas, daftar validasi konsumen yang ditampilkan 10 konsumen. Kolom status validasi berisi status validasi yang merupakan *link* yang menuju ke halaman mengubah data konsumen. Gambar 5.81 ini menunjukkan tampilan halaman melihat daftar validasi konsumen.

Gambar 5. 81 Halaman melihat daftar validasi konsumen melalui PC
Sumber: [Implementasi]

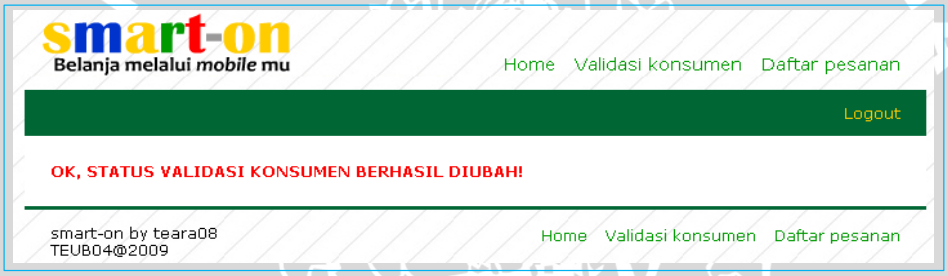
5.4.3.4. Implementasi Antarmuka Mengubah Data Konsumen

Halaman mengubah data konsumen menyediakan fasilitas untuk operator melihat data konsumen dan mengubah data konsumen. Data yang dapat dilihat adalah *username*, nama, alamat, no. *handphone*, *email*, dan status validasi. Data yang dapat diubah adalah status validasi. Gambar 5.82 menunjukkan tampilan halaman mengubah data pegawai.



Gambar 5. 82 Halaman mengubah data konsumen melalui PC
Sumber: [Implementasi]

Pada Gambar 5.83 menunjukkan tampilan konfirmasi data konsumen telah berhasil diubah dan disimpan ke basis data.



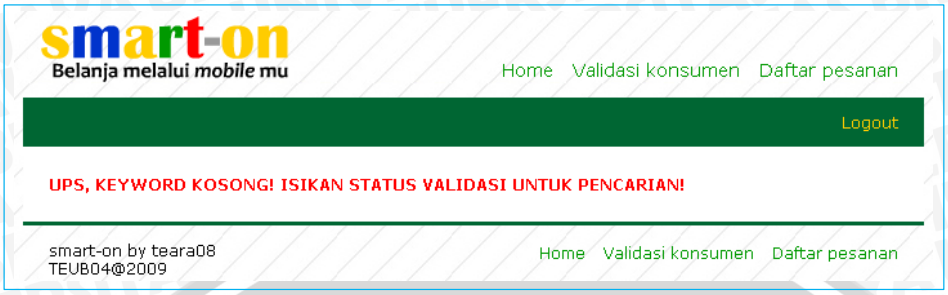
Gambar 5. 83 Halaman sukses mengubah data konsumen melalui PC
Sumber: [Implementasi]

5.4.3.5. Implementasi Antarmuka Mencari Validasi Konsumen

Halaman mencari validasi konsumen menyediakan fasilitas bagi operator untuk mencari konsumen dengan mengisi *form input* pencarian menggunakan *keyword* status validasi. Gambar 5.84 menunjukkan tampilan halaman hasil pencarian validasi konsumen dan *form input* pencarian.

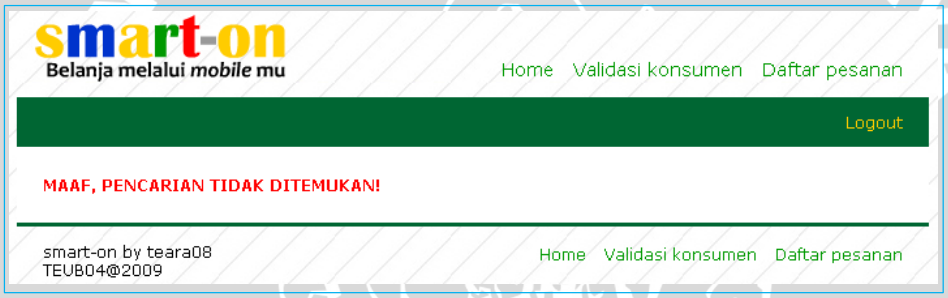
Gambar 5. 84 Halaman hasil pencarian validasi konsumen melalui PC
Sumber: [Implementasi]

Pada gambar 5.85 menunjukkan tampilan halaman konfirmasi *keyword* kosong. Halaman konfirmasi ini terjadi bila operator tidak memasukkan suatu karakter sehingga *form input* pencarian validasi konsumen kosong saat dijalankan (tombol "Cari" di-*klik*).



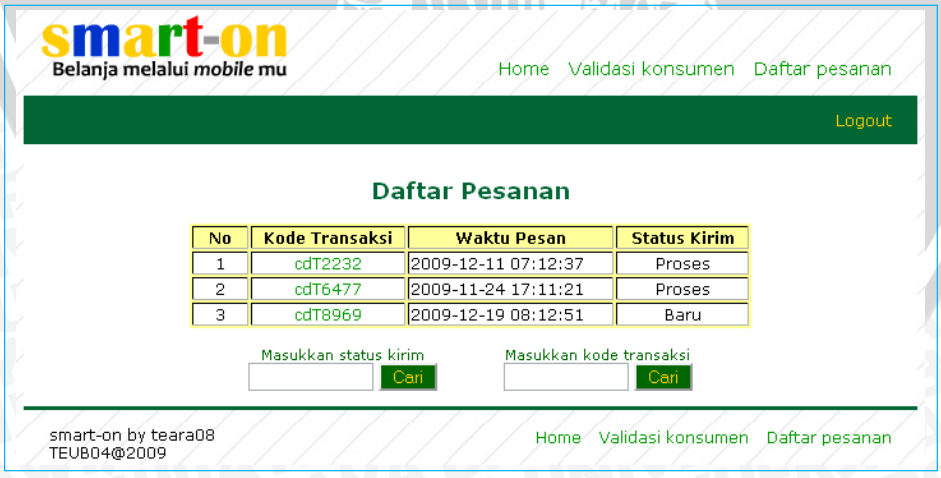
Gambar 5. 85 Halaman konfirmasi *keyword* kosong melalui PC
Sumber: [Implementasi]

Pada gambar 5.86 menunjukkan tampilan halaman konfirmasi pencarian tidak ditemukan. Halaman konfirmasi ini terjadi bila status validasi yang dicari tidak ada pada basis data.



Gambar 5. 86 Halaman konfirmasi pencarian tidak ditemukan melalui PC
Sumber: [Implementasi]

5.4.3.6. Implementasi Antarmuka Melihat Daftar Pesanan



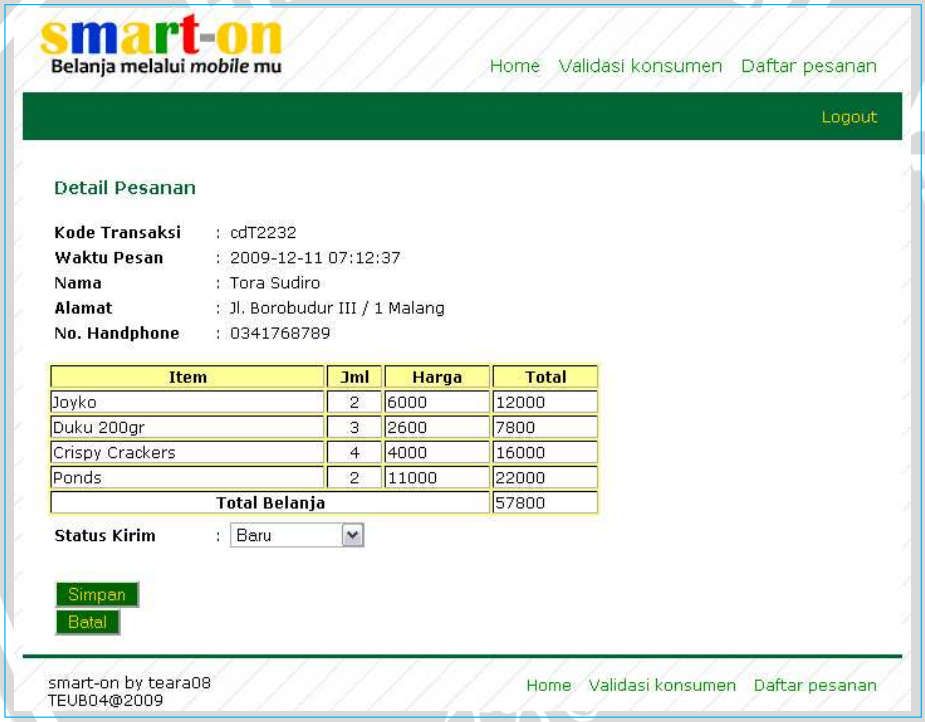
Gambar 5. 87 Halaman melihat daftar pesanan melalui PC
Sumber: [Implementasi]

Gambar 5.87 ini menunjukkan tampilan halaman melihat daftar pesanan. Halaman melihat daftar pesanan ini menampilkan daftar pesanan yang terdiri dari no, kode transaksi,

waktu pesan, dan status kirim. Pada setiap halaman diberi batas, daftar pesanan yang ditampilkan 10 pesanan. Kolom kode transaksi berisi kode transaksi yang merupakan *link* yang menuju ke halaman mengubah data pesanan.

5.4.3.7. Implementasi Antarmuka Mengubah Data Pesanan

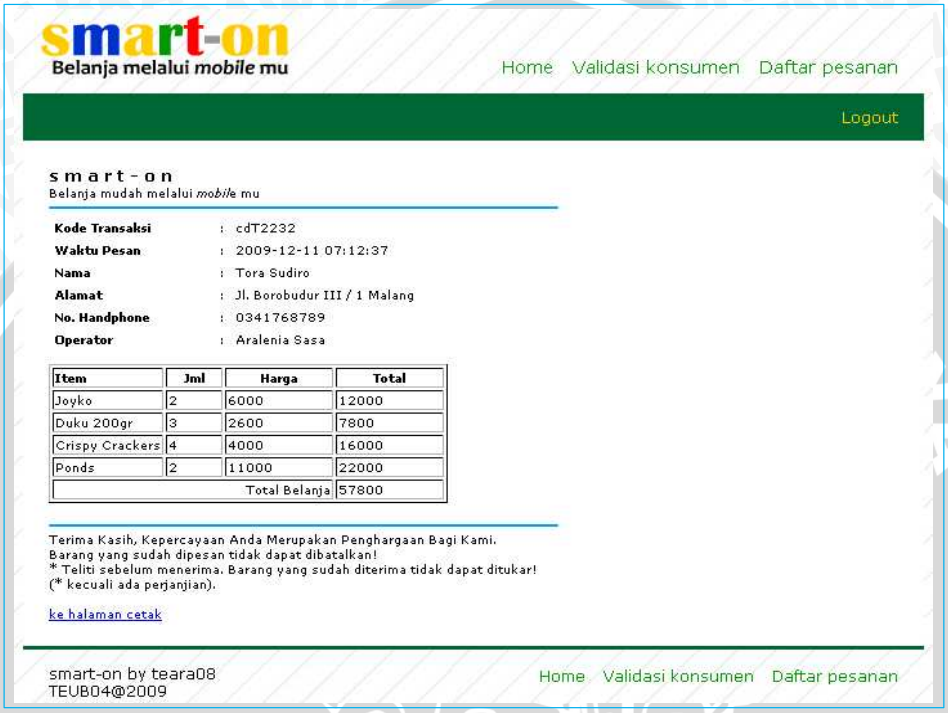
Halaman mengubah data pesanan menyediakan fasilitas untuk operator melihat data konsumen, melihat data pesanan, dan mengubah data pesanan. Data konsumen yang dapat dilihat adalah nama, alamat, dan no. *handphone*. Data pesanan yang dapat dilihat adalah kode transaksi, waktu pesan, daftar barang yang dipesan, dan status kirim. Data yang dapat diubah adalah status kirim. Secara *default*, status kirim bernilai "Baru". Gambar 5.88 menunjukkan tampilan halaman mengubah data pesanan.



Gambar 5. 88 Halaman mengubah data pesanan melalui PC
Sumber: [Implementasi]

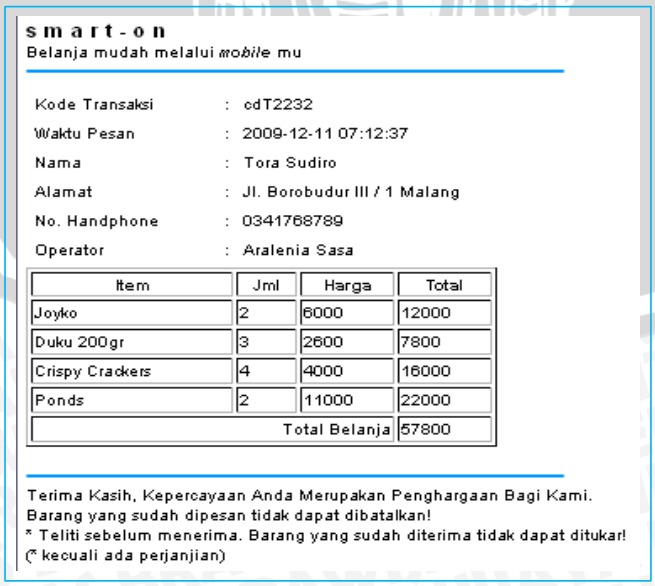
Pada Gambar 5.89 menunjukkan tampilan halaman konfirmasi data pesanan telah berhasil diubah dan disimpan ke basis data. Jika status kirim diubah menjadi "Terkirim" maka akan menuju ke halaman konfirmasi berhasil mengubah status kirim ditunjukkan pada Gambar 5.89 dan disimpan ke basis data. Jika status kirim diubah menjadi "Proses" maka akan menuju ke halaman melihat nota ditunjukkan pada Gambar 5.90 dan status kirim disimpan ke basis data.

Gambar 5. 89 Halaman sukses mengubah status kirim menjadi 'Terkirim' melalui PC
Sumber: [Implementasi]



Gambar 5. 90 Halaman sukses mengubah status kirim menjadi 'Proses' melalui PC
Sumber: [Implementasi]

5.4.3.8. Implementasi Antarmuka Mencetak Nota



Gambar 5. 91 Halaman mencetak nota melalui PC
Sumber: [Implementasi]

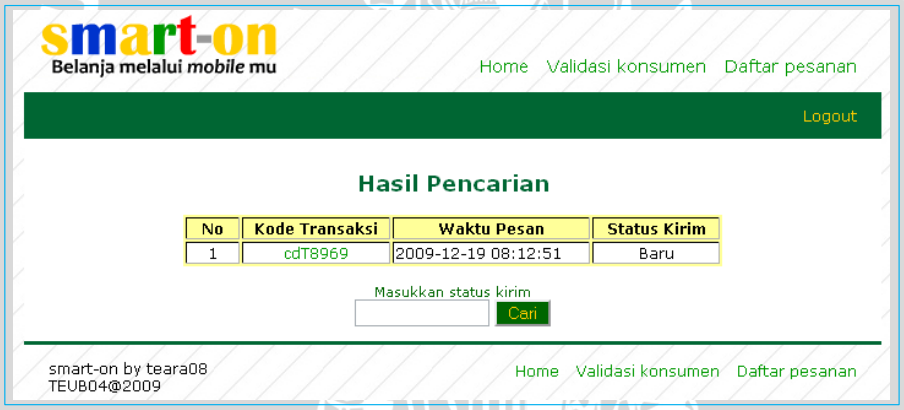
Gambar 5.91 menunjukkan tampilan halaman mencetak nota. Halaman mencetak nota merupakan tampilan nota yang nantinya dicetak dan menjadi bukti transaksi.

Pada Gambar 5.92 menunjukkan tampilan nota telah berhasil dicetak.

Gambar 5. 92 Halaman sukses mencetak nota melalui PC
Sumber: [Implementasi]

5.4.3.9. Implementasi Antarmuka Mencari Pesanan

Halaman mencari pesanan menyediakan fasilitas bagi operator untuk mencari pesanan dengan mengisi *form input* pencarian, dapat menggunakan *keyword* status kirim atau kode transaksi. Gambar 5.93 menunjukkan tampilan hasil pencarian pesanan dengan menggunakan *keyword* status kirim pada *form input* pencarian. Gambar 5.94 menunjukkan tampilan hasil pencarian pesanan dengan menggunakan *keyword* kode transaksi pada *form input* pencarian.

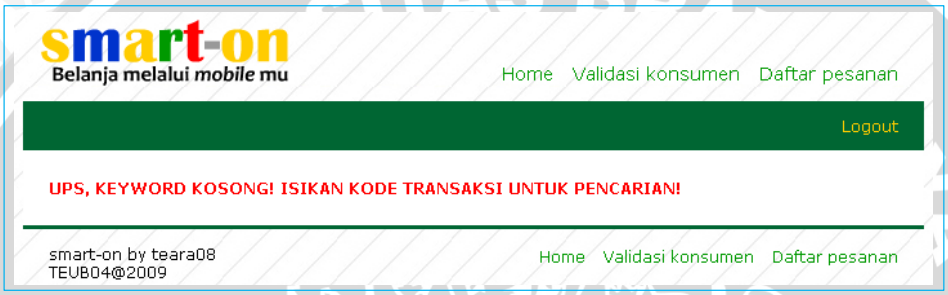


Gambar 5. 93 Halaman hasil pencarian dengan *keyword* status kirim melalui PC
Sumber: [Implementasi]

Gambar 5. 94 Halaman hasil pencarian dengan *keyword* kode transaksi melalui PC
Sumber: [Implementasi]

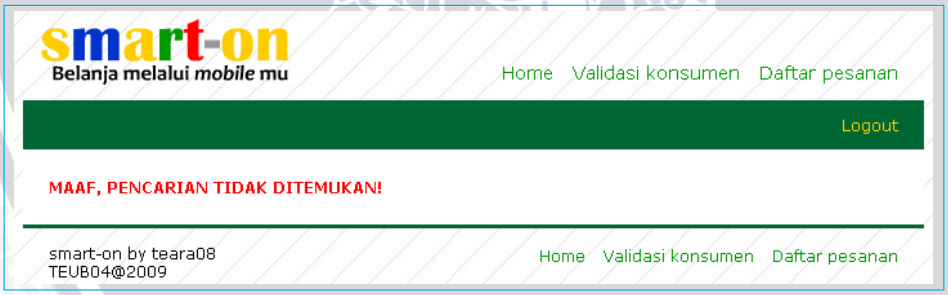
Pada Gambar 5.95 dan Gambar 5.96 menunjukkan halaman konfirmasi *keyword* kosong. Halaman konfirmasi ini terjadi bila operator tidak memasukkan suatu karakter sehingga *form input* pencarian pesanan kosong saat dijalankan (tombol "Cari" di-*klik*).

Gambar 5. 95 Halaman konfirmasi *keyword* status kirim melalui PC
Sumber: [Implementasi]



Gambar 5. 96 Halaman konfirmasi *keyword* kode transaksi kosong melalui PC
Sumber: [Implementasi]

Pada gambar 5.97 menunjukkan halaman konfirmasi pencarian tidak ditemukan. Halaman konfirmasi ini terjadi bila kode transaksi atau status kirim yang dicari tidak ada pada basis data.



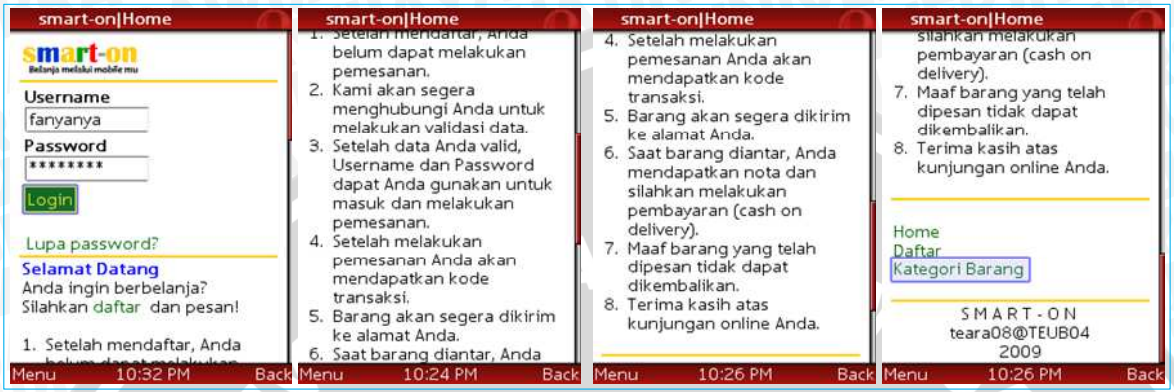
Gambar 5. 97 Halaman konfirmasi pencarian tidak ditemukan melalui PC
Sumber: [Implementasi]

5.4.4. Implementasi Antarmuka Konsumen Yang Ditampilkan Melalui Mobile *Phone*

Pada implementasi antarmuka konsumen ini ditampilkan melalui *mobile phone*. Tampilan disesuaikan dengan layar *mobile phone* sehingga konsumen nyaman saat melihatnya.

5.4.4.1. Implementasi Antarmuka *Login*

Halaman *login* yang juga merupakan halaman *index* ini menyediakan fasilitas *login* untuk autentifikasi konsumen agar dapat masuk dan menggunakan fasilitas sistem sebagai konsumen. Bagi *user* terdapat layanan informasi dan *link* untuk mendaftar sebagai konsumen. Gambar 5.98 menunjukkan tampilan halaman *login*.



Gambar 5. 98 Halaman *login* konsumen melalui *mobile phone*
Sumber: [Implementasi]

5.4.4.2. Implementasi Antarmuka Lupa *Password*

Halaman lupa *password* ini menyediakan fasilitas me-*request password* baru. Setelah konsumen mengisi *form input email* dan meng-*klik* tombol "Kirim", sistem akan memproses dan mengirimkan *password* baru ke alamat *email* yang diisikan pada *form input email*. Gambar 5.99 menunjukkan tampilan halaman lupa *password*.

Gambar 5. 99 Halaman lupa *password* melalui *mobile phone*
Sumber: [Implementasi]

Pada gambar 5.100 menunjukkan tampilan halaman konfirmasi *password* baru telah berhasil dikirimkan ke alamat *email.*

Gambar 5. 100 Halaman sukses mengirim *password* baru melalui *mobile phone*
Sumber: [Implementasi]

Pada gambar 5.101 menunjukkan halaman konfirmasi *form input email* kosong. Halaman konfirmasi ini terjadi bila konsumen tidak memasukkan suatu karakter sehingga *form input email* kosong saat dijalankan (tombol "Kirim" di-*klik*).

Gambar 5. 101 Halaman konfirmasi *email* tidak diisi melalui *mobile phone*
Sumber: [Implementasi]

Pada gambar 5.102 menunjukkan halaman konfirmasi *email* salah. Halaman konfirmasi ini terjadi bila alamat *email* yang diisikan tidak ada pada basis data.

Gambar 5. 102 Halaman konfirmasi *email* tidak terdaftar melalui *mobile phone*
Sumber: [Implementasi]

5.4.4.3. Implementasi Antarmuka *Home* Konsumen

Halaman *home* konsumen ini berisi *link-link* untuk mengatur seluruh kebutuhan konsumen. Gambar 5.103 menunjukkan tampilan halaman *home* konsumen.

Gambar 5. 103 Halaman *home* konsumen melalui *mobile phone*
Sumber: [Implementasi]

5.4.4.4. Implementasi Antarmuka Mengubah *Id Login* Konsumen

Gambar 5. 104 Halaman mengubah *id login* konsumen melalui *mobile phone*
Sumber: [Implementasi]

Halaman mengubah *id login* konsumen ditunjukkan pada Gambar 5.104. Halaman mengubah *id login* konsumen menyediakan fasilitas untuk mengubah *username* dan *password* konsumen. Halaman ini berisi *form input username*, *password*, dan ulangi *password*.

Jika *username* dan *password* berhasil diubah dan disimpan di dalam basis data maka sistem akan menampilkan konfirmasi *username* dan *password* barhasil diubah. Gambar 5.105 menunjukkan tampilan halaman sukses mengubah *id login* konsumen.

Gambar 5. 105 Halaman sukses mengubah *id login* konsumen melalui *mobile phone*
Sumber: [Implementasi]

5.4.4.5. Implementasi Antarmuka Mengubah Data Konsumen

Halaman mengubah data konsumen menyediakan fasilitas untuk konsumen mengubah datanya. Data yang dapat diubah adalah nama lengkap, alamat lengkap, no. *handphone*, dan *email*. Gambar 5.106 menunjukkan tampilan halaman mengubah data konsumen.

Gambar 5. 106 Halaman mengubah data konsumen melalui *mobile phone*
Sumber: [Implementasi]

Gambar 5. 107 Halaman konfirmasi *email* sama melalui *mobile phone*
Sumber: [Implementasi]

Gambar 5.107 menunjukkan tampilan halaman konfirmasi alamat *email* sama. Jika alamat *email* yang diisikan pada *form input email* sudah ada di dalam basis data maka sistem akan menampilkan konfirmasi alamat *email* sudah digunakan.

Pada Gambar 5.108 menunjukkan tampilan konfirmasi data konsumen telah berhasil diubah dan disimpan ke basis data. Sebelum operator menghubungi dan memvalidasi data konsumen maka konsumen yang telah mengubah datanya belum dapat melakukan *login*.

Gambar 5. 108 Halaman sukses mengubah data konsumen melalui *mobile phone*
Sumber: [Implementasi]

5.4.4.6. Implementasi Antarmuka Melihat Daftar Kategori Barang

Gambar 5. 109 Halaman melihat daftar kategori barang melalui *mobile phone*
Sumber: [Implementasi]

Gambar 5.109 ini menunjukkan gambar halaman melihat daftar kategori barang. Halaman melihat daftar kategori barang ini menampilkan daftar kategori barang. Pada setiap halaman diberi batas, daftar kategori barang yang ditampilkan 10 kategori barang. Setiap kategori barang merupakan *link* ke halaman melihat daftar subkategori per kategori dan barang per kategori seperti ditunjukkan pada Gambar 5.110.

5.4.4.7. Implementasi Antarmuka Melihat Daftar Subkategori Barang

Gambar 5. 110 Halaman melihat daftar subkategori barang melalui *mobile phone*
Sumber: [Implementasi]

Gambar 5.110 ini menunjukkan gambar halaman melihat daftar subkategori barang. Halaman melihat daftar subkategori barang ini menampilkan daftar subkategori per kategori dan daftar barang per kategori. Setiap subkategori barang merupakan *link* ke halaman melihat daftar barang per subkategori seperti ditunjukkan pada Gambar 5.114.

5.4.4.8. Implementasi Antarmuka Mencari Barang

Halaman mencari barang menyediakan fasilitas bagi konsumen untuk mencari barang dengan mengisi *form input* pencarian menggunakan *keyword* nama barang. Gambar 5.111 menunjukkan tampilan hasil pencarian barang dan *form input* pencarian.

Gambar 5. 111 Halaman hasil pencarian barang melalui *mobile phone*
Sumber: [Implementasi]

Pada Gambar 5.112 menunjukkan halaman konfirmasi *keyword* kosong. Halaman konfirmasi ini terjadi bila konsumen tidak memasukkan suatu karakter sehingga *form input* pencarian barang kosong saat dijalankan (tombol “Cari” di-*klik*).

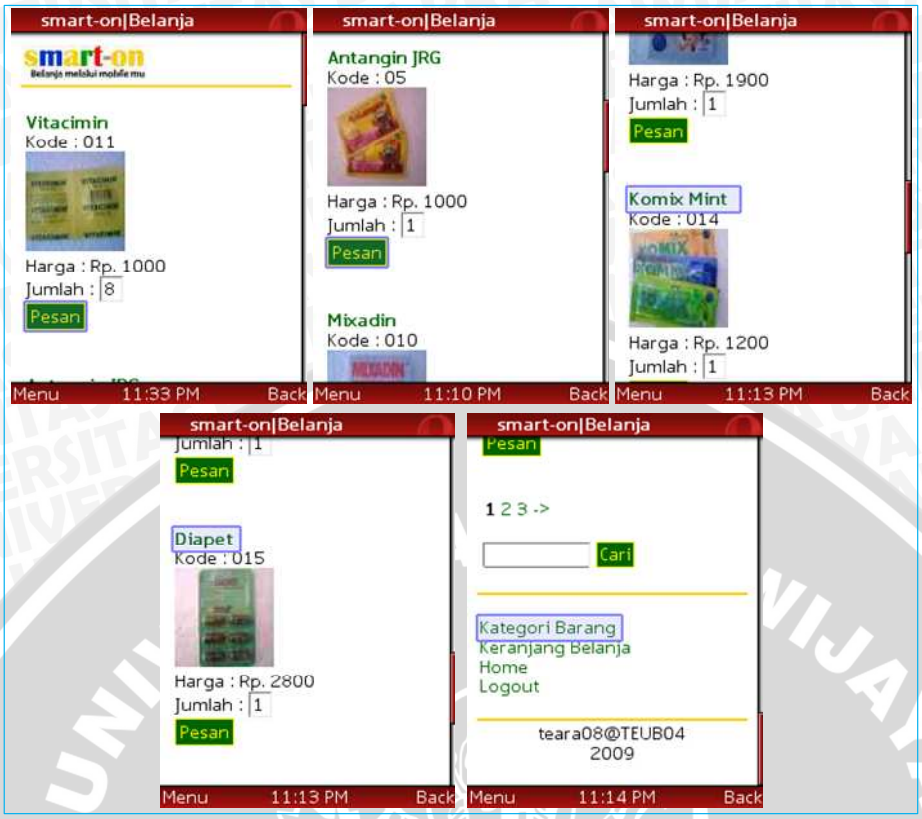
Gambar 5. 112 Halaman konfirmasi *keyword* kosong melalui *mobile phone*
Sumber: [Implementasi]

Pada Gambar 5.113 menunjukkan halaman konfirmasi pencarian tidak ditemukan. Halaman konfirmasi ini terjadi bila nama barang yang dicari tidak ada pada basis data.

Gambar 5. 113 Halaman konfirmasi pencarian tidak ditemukan melalui *mobile phone*
Sumber: [Implementasi]

5.4.4.9. Implementasi Antarmuka Memesan Barang

Pada halaman memesan barang ini menyediakan fasilitas bagi konsumen untuk memesan barang dengan jumlah yang diinginkan. Gambar 5.114 menunjukkan tampilan halaman daftar barang per subkategori.



Gambar 5. 114 Halaman memesan pada daftar barang per subkategori melalui *mobile phone*
Sumber: [Implementasi]

Gambar 5. 115 Halaman memesan pada data barang melalui *mobile phone*
Sumber: [Implementasi]

Gambar 5.115 menunjukkan tampilan halaman data barang. Memesan barang juga dapat dilakukan saat melihat halaman data barang, yang mana ukuran gambar lebih besar dari pada ukuran gambar pada halaman daftar barang per subkategori serta mempunyai keterangan tentang barang.

Pada Gambar 5.116 menunjukkan halaman konfirmasi memesan barang berhasil dilakukan dan disimpan di dalam keranjang belanja.

Gambar 5. 116 Halaman sukses memesan barang melalui *mobile phone*
Sumber: [Implementasi]

Pada Gambar 5.117 menunjukkan halaman konfirmasi memesan barang tidak berhasil dilakukan karena stok barang tidak mencukupi untuk dipesan.

Gambar 5. 117 Halaman stok barang tidak mencukupi melalui *mobile phone*
Sumber: [Implementasi]

5.4.4.10. Implementasi Antarmuka Melihat Keranjang Belanja

Gambar 5. 118 Halaman melihat keranjang barang melalui *mobile phone*
Sumber: [Implementasi]

Pada Gambar 5.118 ini menunjukkan tampilan halaman melihat keranjang belanja. Halaman melihat keranjang belanja ini menampilkan daftar barang yang akan dipesan. Di halaman keranjang belanja setiap barang memiliki *form input* jumlah barang sehingga

konsumen dapat mengubah jumlah barang dan terdapat menu “hapus” untuk menghapus barang.

5.4.4.11. Implementasi Antarmuka Menghapus Keranjang Belanja

Pada Gambar 5.119 menunjukkan halaman konfirmasi keranjang belanja berhasil dikosongkan atau dengan kata lain semua barang yang terdapat di keranjang belanja telah dihapus.

Gambar 5. 119 Halaman sukses menghapus keranjang belanja melalui *mobile phone*
Sumber: [Implementasi]

5.4.4.12. Implementasi Antarmuka Melihat Detail Pesanan

Halaman melihat detail pesanan ini menampilkan daftar barang yang akan dipesan. Sama dengan halaman keranjang belanja, namun pada detail pesanan ini data barang yang akan dipesan tidak dapat diubah serta menampilkan kode keamanan dan *form* isian konfirmasi kode keamanan. Gambar 5.120 ini menunjukkan tampilan halaman melihat detail pesanan.

Gambar 5. 120 Halaman melihat detail pesanan melalui *mobile phone*
Sumber: [Implementasi]

Pada Gambar 5.121 menunjukkan halaman konfirmasi kode keamanan yang diisikan salah sehingga *check out* yang dilakukan tidak berhasil.

Gambar 5. 121 Halaman konfirmasi kode keamanan salah dan gagal *check out* melalui *mobile phone*
Sumber: [Implementasi]

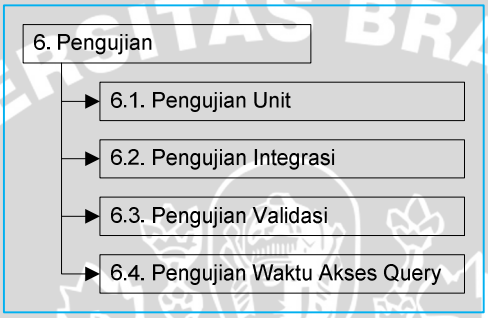
5.4.4.13. Implementasi Antarmuka *Check Out*

Halaman *check out* ini menampilkan data transaksi dan data konsumen. Halaman *check out* ini juga merupakan konfirmasi bahwa konfirmasi kode keamanan yang diisikan benar, sehingga pemesanan berhasil diproses dan data telah disimpan di basis data. Data yang ditampilkan adalah kode transaksi, waktu pesan, nama, alamat, no. *handphone*, dan total belanja. Gambar 5.122 ini menunjukkan tampilan halaman *check out*.

Gambar 5. 122 Halaman *check out* melalui *mobile phone*
Sumber: [Implementasi]

BAB VI
PENGUJIAN

Pada bab ini dilakukan proses pengujian terhadap implementasi sistem *mobile online supermarket*. Proses pengujian dilakukan melalui tiga tahapan yaitu pengujian unit, pengujian integrasi, dan pengujian validasi. Pada pengujian unit dan pengujian integrasi, akan digunakan teknik pengujian *White Box (White Box Testing)*. Pada pengujian validasi akan digunakan teknik pengujian *Black Box (Black Box Testing)*. Pengujian waktu akses query yang dilakukan secara terpisah.



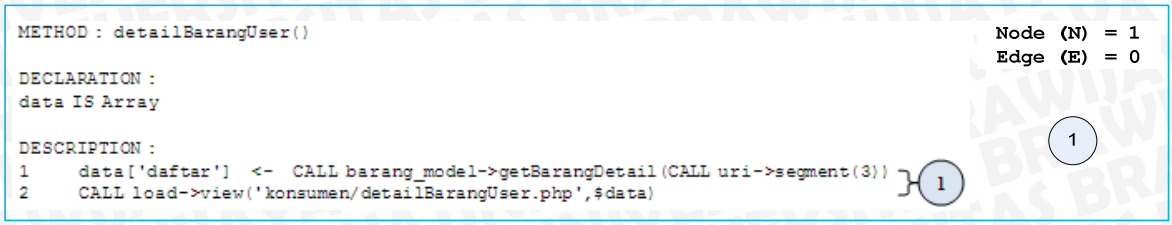
Gambar 6. 1 Diagram pohon pengujian sistem
Sumber: [Pengujian]

6.1. Pengujian Unit

Sistem yang dibangun dengan paradigma pemrograman berorientasi objek menerapkan pengujian unit untuk suatu *method* (operasi) dari suatu klas. Pada pengujian unit sistem *mobile online supermarket* 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 *method*).

a. Pengujian Unit untuk Method `detailBarangUser()` dari Class Barang

Method `detailBarangUser()` digunakan untuk menampilkan data barang yang dapat dilihat *user* yang tidak *login*. *Method* `detailBarangUser()` terdapat di dalam klas barang. Gambar 6.2 memperlihatkan proses pemodelan dalam *flow graph* pada *method* `detailBarangUser()`.



Gambar 6. 2 Pemodelan *method* `detailBarangUser()` ke dalam *flow graph*
Sumber: [Pengujian]

Pemodelan ke dalam *flow graph* yang telah dilakukan terhadap *method* `detailBarangUser()` 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
= 0 – 1 + 2
= 1

Dari nilai *cyclomatic complexity* yang telah dihasilkan dari perhitungan yaitu 1 ditentukan sebuah basis set dari jalur independen yaitu:

Jalur 1 : 1

Penentuan kasus uji untuk masing-masing jalur dan hasil eksekusi untuk masing-masing kasus uji dijelaskan pada Tabel 6.1.

Tabel 6. 1 *Test case* untuk pengujian unit *method* `detailBarangUser()`

Jalur	Kasus uji	Hasil yang diharapkan	Hasil yang didapatkan
1	`CALL barang_model->getBarangDetail(CALL uri->segment(3))`	Sistem menampilkan data barang untuk *user* yang tidak *login*.	Sistem menampilkan data barang untuk *user* yang tidak *login*.

Sumber: [Pengujian]

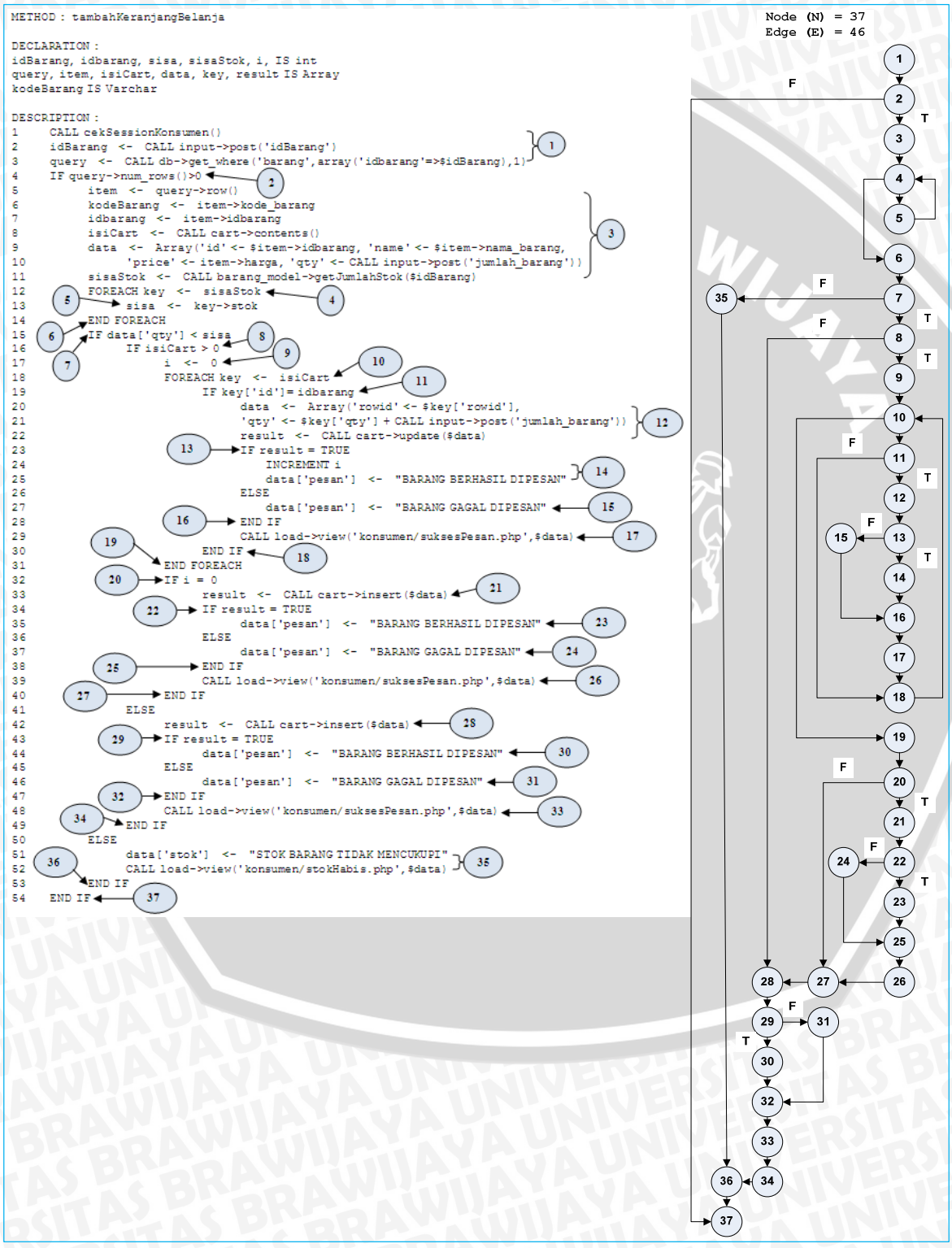
Dari hasil pengujian unit *method* `detailBarangUser()` yang dilakukan dapat diketahui bahwa unit ini dapat bekerja dengan baik karena keluaran hasil yang didapatkan sesuai dengan hasil yang diharapkan.

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*, dimana modul-modul yang diintegrasikan masing-masing diuji terlebih dahulu dalam pengujian unit dan kemudian bergerak menuju ke pengujian modul-modul kontrol yang mengintegrasikannya.

a. Pengujian Integrasi untuk Method `tambahKeranjangBelanja()` Class Pesanan



Gambar 6. 3 Pemodelan *method* tambahKeranjangBelanja() ke dalam *flow graph*
Sumber: [Pengujian]

Gambar 6.3 memperlihatkan proses pemodelan dalam *flow graph* pada *method* `tambahKeranjangBelanja()`. *Method* `tambahKeranjangBelanja()` digunakan untuk menampilkan dan menambah data barang yang dipesan. *Method* `tambahKeranjangBelanja()` terdapat di dalam klas pesanan.

Pemodelan ke dalam *flow graph* yang telah dilakukan terhadap *method* `tambahKeranjangBelanja()` 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
= 46 – 37 + 2
= 11

Dari nilai *cyclomatic complexity* yang telah dihasilkan dari perhitungan yaitu 11 ditentukan sebelas basis set dari jalur independen yaitu:

Jalur 1 : 1-2-3-4-5-4-6-7-8-9-10-11-12-13-14-16-17-18-10-19-20-21-22-23-25-26-27-28-29-30-32-33-34-36-37

Jalur 2 : 1-2-3-...-29-31-32-33-34-36-37

Jalur 3 : 1-2-3-...-22-24-25-26-27-28-29-30-32-33-34-36-37

Jalur 4 : 1-2-3-...-20-21-22-23-25-26-27-28-29-30-32-33-34-36-37

Jalur 5 : 1-2-3-...-20-27-28-29-30-32-33-34-36-37

Jalur 6 : 1-2-3-...-11-18-10-19-20-21-22-23-25-26-27-28-29-30-32-33-34-36-37

Jalur 7 : 1-2-3-...-11-18-10-19-20-27-28-29-31-32-33-34-36-37

Jalur 8 : 1-2-3-...-8-28-29-30-32-33-34-36-37

Jalur 9 : 1-2-3-...-8-28-29-31-32-33-34-36-37

Jalur 10 : 1-2-3-4-5-4-6-7-35-36-37

Jalur 11 : 1-2-37

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* `tambahKeranjangBelanja()`

Jalur	Kasus uji	Hasil yang diharapkan	Hasil yang didapatkan
1	Cart tidak kosong, stok barang ada, terdapat idbarang yang sama, `result` bernilai TRUE	Sistem menyimpan penambahan jumlah per barang dalam array data dan menampilkan pesan.	Sistem menyimpan penambahan jumlah per barang dalam array data dan menampilkan pesan.
2	Cart tidak kosong, stok barang ada, terdapat idbarang yang sama,	Sistem menampilkan pesan error	Sistem menampilkan pesan error

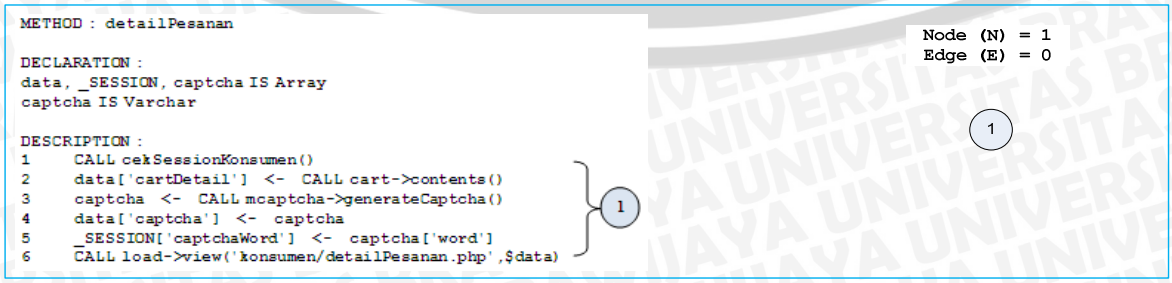
	`result` bernilai FALSE		
3	Cart tidak kosong, barang belum ada dalam cart ,`CALL cart insert ($data )`bernilai FALSE	Sistem tidak menyimpan dan menampilkan pesan error	Sistem tidak menyimpan dan menampilkan pesan error
4	Cart tidak kosong, barang belum ada dalam cart ,`CALL cart insert ($data )`bernilai TRUE	Sistem menyimpan barang baru dalam array data dan menampilkan pesan.	Sistem menyimpan barang baru dalam array data dan menampilkan pesan.
5	Cart kosong, `CALL cart insert ($data )`bernilai TRUE	Sistem menyimpan barang baru dalam array data dan menampilkan pesan.	Sistem menyimpan barang baru dalam array data dan menampilkan pesan.
6	Pengecekan isiCart dengan idBarang yang sama, jika belum ada barang dengan idBarang yang sama, `CALL cart insert ($data)`bernilai TRUE	Sistem menyimpan barang baru dalam array data dan menampilkan pesan.	Sistem menyimpan barang baru dalam array data dan menampilkan pesan.
7	Pengecekan isiCart dengan keadaan cart kosong, `CALL cart insert ($data)`bernilai FALSE	Sistem tidak menyimpan dan menampilkan pesan error	Sistem tidak menyimpan dan menampilkan pesan error
8	`CALL cart insert ($data)pada` bernilai TRUE	Sistem menyimpan barang baru dalam array data dan menampilkan pesan.	Sistem menyimpan barang baru dalam array data dan menampilkan pesan.
9	`CALL cart insert ($data)` bernilai FALSE	Sistem tidak menyimpan dan menampilkan pesan error	Sistem tidak menyimpan dan menampilkan pesan error
10	`data [qty] > sisa`, jumlah barang yang dipesan lebih besar dari stok yang ada	Sistem menampilkan pesan barang tidak mencukupi	Sistem menampilkan pesan barang tidak mencukupi
11	`query->num_rows() > 0`, tidak ada data	Sistem menampilkan cart kosong	Sistem menampilkan cart kosong

Sumber: [Pengujian]

Dari hasil pengujian integrasi *method* `tambahKeranjangBelanja()` yang dilakukan dapat diketahui bahwa sistem dapat melakukan penambahan jumlah barang dan menyimpan barang dengan baik.

b. Pengujian Integrasi untuk Method `detailPesanan()`Class Pesanan

Method `detailPesanan()` digunakan untuk menampilkan data barang yang dipesan. *Method* `detailPesanan()` terdapat di dalam klas pesanan. Gambar 6.4 memperlihatkan proses pemodelan dalam *flow graph* pada *method* `detailPesanan()`.



Gambar 6. 4 Pemodelan *method* `detailPesanan()` ke dalam *flow graph*
Sumber: [Pengujian]

Pemodelan ke dalam *flow graph* yang telah dilakukan terhadap *method* `detailPesanan()` 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
= 0 – 1 + 2
= 1

Dari nilai *cyclomatic complexity* yang telah dihasilkan dari perhitungan yaitu 1 ditentukan sebuah basis set dari jalur independen yaitu:

Jalur 1 : 1

Penentuan kasus uji untuk masing-masing jalur dan hasil eksekusi untuk kasus uji dijelaskan pada Tabel 6.3.

Tabel 6. 3 *Test case* untuk pengujian integrasi *method* `detailPesanan()`

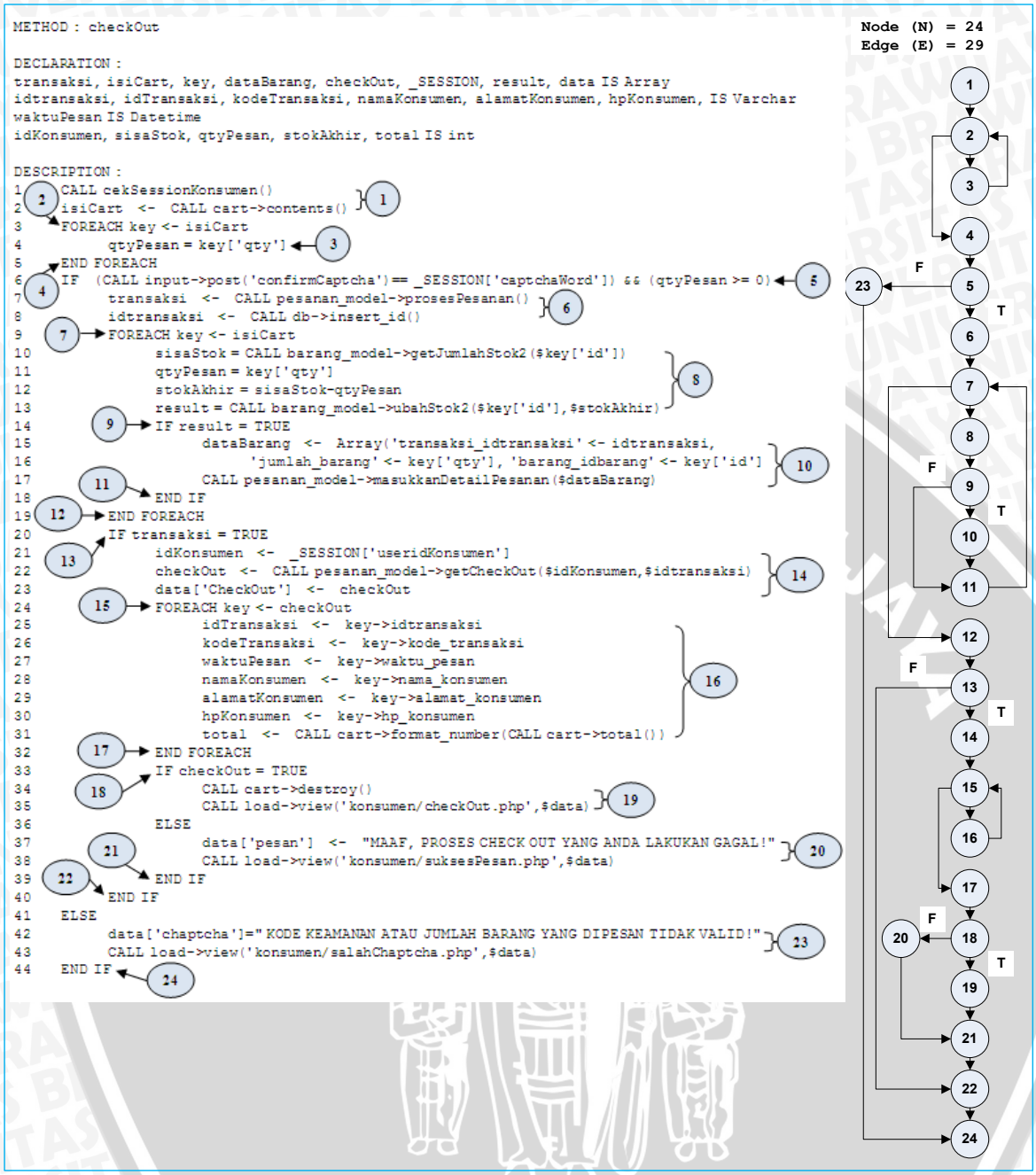
Jalur	Kasus uji	Hasil yang diharapkan	Hasil yang didapatkan
1	`CALL cart -> contents(), CALL mcaptcha -> generateChaptcha()`	Sistem menampilkan semua isi *cart* dan *form* kode keamanan	Sistem menampilkan semua isi *cart* dan *form* kode keamanan

Sumber: [Pengujian]

Dari hasil pengujian integrasi *method* `detailPesanan()` yang dilakukan dapat diketahui bahwa sistem dapat menampilkan daftar barang yang akan dipesan dengan baik.

c. Pengujian Integrasi untuk Method `checkOut()` dari Class Pesanan

Method `checkOut()` digunakan untuk menampilkan data transaksi. *Method* `checkOut()` terdapat di dalam klas pesanan. Gambar 6.5 memperlihatkan proses pemodelan dalam *flow graph* pada *method* `checkOut()`.



Gambar 6. 5 Pemodelan *method* `checkOut()` ke dalam *flow graph*
Sumber: [Pengujian]

Pemodelan ke dalam *flow graph* yang telah dilakukan terhadap *method* `checkOut()` 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
= 29 – 24 + 2
= 7

Dari nilai *cyclomatic complexity* yang telah dihasilkan dari perhitungan yaitu 7 ditentukan tujuh buah basis set dari jalur independen yaitu:

Jalur 1 : 1-2-3-2-4-5-6-7-8-9-10-11-7-12-13-14-15-16-15-17-18-19-21-22-24

Jalur 2 : 1-2-3-...-18-20-21-22-24

Jalur 3 : 1-2-3-...-13-22-24

Jalur 4 : 1-2-3-...-9-11-7-12-13-14-15-16-15-17-18-19-21-22-24

Jalur 5 : 1-2-3-...-9-11-7-12-13-14-15-16-15-17-18-20-21-22-24

Jalur 6 : 1-2-3-...-9-11-7-12-13-22-24

Jalur 7 : 1-2-3-2-4-5-23-24

Penentuan kasus uji untuk masing-masing jalur dan hasil eksekusi untuk kasus uji dijelaskan pada Tabel 6.4.

Tabel 6. 4 *Test case* untuk pengujian integrasi *method* `checkOut()`

Jalur	Kasus uji	Hasil yang diharapkan	Hasil yang didapatkan
1	Masukkan kode konfirmasi dan jumlah barang yang dipesan valid, `result` bernilai TRUE, `checkOut` bernilai TRUE	Data barang dihapus dari array dataBarang dan menampilkan data checkOut	Data barang dihapus dari array dataBarang dan menampilkan data checkOut
2	Masukkan kode konfirmasi dan jumlah barang yang dipesan valid, `result` bernilai TRUE, `checkOut` bernilai FALSE	Sistem menampilkan pesan error bahwa check out gagal	Sistem menampilkan pesan error bahwa check out gagal
3	Masukkan kode konfirmasi dan jumlah barang yang dipesan valid, `result` bernilai TRUE, `transaksi` bernilai FALSE	Sistem menampilkan pesan error bahwa check out gagal	Sistem menampilkan pesan error bahwa check out gagal
4	Masukkan kode konfirmasi dan jumlah barang yang dipesan valid, `result` bernilai FALSE, `checkOut` bernilai TRUE	Sistem mengulang sampai `result` bernilai TRUE, sehingga data barang dihapus dari array dataBarang dan menampilkan data checkOut	Sistem mengulang sampai `result` bernilai TRUE, sehingga data barang dihapus dari array dataBarang dan menampilkan data checkOut
5	Masukkan kode konfirmasi dan jumlah barang yang dipesan valid, `result` bernilai FALSE, `checkOut` bernilai FALSE	Sistem mengulang sampai `result` bernilai TRUE, sehingga dapat menampilkan pesan error bahwa check out gagal	Sistem mengulang sampai `result` bernilai TRUE, sehingga dapat menampilkan pesan error bahwa check out gagal
6	Masukkan kode konfirmasi dan jumlah barang yang dipesan valid, `result` bernilai FALSE, `transaksi` bernilai FALSE	Sistem menampilkan pesan error bahwa check out gagal	Sistem menampilkan pesan error bahwa check out gagal
7	`input->post ('confirmCaptcha')` tidak sama dengan `_SESSION('captchaWord')`, kode konfirmasi salah, atau `qtyPesan >= 0` salah	Sistem menampilkan pesan bahwa kode konfirmasi dan jumlah barang yang dipesan tidak valid	Sistem menampilkan pesan bahwa kode konfirmasi dan jumlah barang yang dipesan tidak valid

Sumber: [Pengujian]

Dari hasil pengujian integrasi *method* `checkOut()` yang dilakukan dapat diketahui bahwa sistem dapat menyimpan data transaksi di basis data dan menampilkan informasi transaksi dengan baik.

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.

1. Kasus Uji Login

Tabel 6.5 dan tabel 6.6 menunjukkan kasus uji *login*.

Tabel 6. 5 Kasus uji *login*

Nama kasus uji	Kasus uji *login*
Obyek uji	Kebutuhan nomer 1 (aliran utama)
Tujuan pengujian	Menguji bahwa hanya pengguna tertentu yang telah terdaftar dapat masuk dan mengakses fasilitas tertentu (sebagai suatu aktor tertentu).
Prosedur uji	1. Sistem telah berjalan. 2. Memasukkan *username* dan *password* pada kotak isian (*username* = ”teara08”, *password* = “teara08”). 3. Menekan tombol login.
Hasil yang diharapkan	Pengguna berhasil login pada sistem.

Sumber: [Pengujian]

Tabel 6. 6 Kasus uji *login* dimana pasangan *username* dan *password* tidak ada dalam sistem

Nama kasus uji	Kasus uji *login* dimana pasangan *username* dan *password* tidak ada dalam sistem
Obyek uji	Kebutuhan nomer 1 (aliran alternatif 1)
Tujuan pengujian	Menguji bahwa hanya pengguna tertentu yang telah terdaftar dapat masuk dan mengakses fasilitas tertentu (sebagai suatu aktor tertentu).
Prosedur uji	1. Sistem telah berjalan. 2. Memasukkan *username* dan *password* pada kotak isian yang tidak ada dalam sistem (*username* = ”teara”, *password* = “teara”). 3. Menekan tombol login.
Hasil yang diharapkan	Kembali ke halaman *login*.

Sumber: [Pengujian]

2. Kasus Uji Logout

Tabel 6.7 menunjukkan kasus uji *logout*.

Tabel 6. 7 Kasus uji *logout*

Nama kasus uji	Kasus uji *logout*
Obyek uji	Kebutuhan nomer 2
Tujuan pengujian	Menguji bahwa sistem dapat menyediakan fasilitas agar pengguna yang telah *login* dapat keluar dari sistem.
Prosedur uji	1. Sistem telah berjalan dan pengguna telah *login*. 2. Pengguna menekan tombol *logout*.
Hasil yang diharapkan	Pengguna berhasil *logout* dari sistem.

Sumber: [Pengujian]

3. Kasus Uji Melihat Daftar Pegawai

Tabel 6.8 menunjukkan kasus melihat daftar pegawai.

Tabel 6. 8 Kasus uji melihat daftar pegawai

Nama kasus uji	Kasus uji melihat daftar pegawai
Obyek uji	Kebutuhan nomer 3
Tujuan pengujian	Menguji bahwa sistem dapat menampilkan semua pegawai yang tercatat di dalam sistem. Data yang ditampilkan berupa kode, nama, dan tugas.
Prosedur uji	1. Sistem telah berjalan dan pengguna telah login sebagai administrator. 2. Memilih menu pegawai.
Hasil yang diharapkan	Semua daftar pegawai dapat ditampilkan.

Sumber: [Pengujian]

4. Kasus Uji Melihat Data Pegawai

Tabel 6.9 menunjukkan kasus uji melihat data pegawai.

Tabel 6. 9 Kasus uji melihat data pegawai

Nama kasus uji	Kasus uji melihat data pegawai
Obyek uji	Kebutuhan nomer 4
Tujuan pengujian	Menguji bahwa sistem dapat menampilkan data pegawai yang tercatat di dalam sistem. Data yang ditampilkan berupa kode, *username*, nama, *sex*, tempat lahir, tanggal lahir, alamat, no. telepon, *email*, dan tugas.
Prosedur uji	1. *Use case* melihat daftar pegawai telah dijalankan. 2. Memilih kode salah satu pegawai dari daftar pegawai.
Hasil yang diharapkan	Data pegawai dapat ditampilkan.

Sumber: [Pengujian]

5. Kasus Uji Menambah Pegawai

Tabel 6.10, 6.11, dan 6.12 menunjukkan kasus uji menambah pegawai.

Tabel 6. 10 Kasus uji menambah pegawai

Nama kasus uji	Kasus uji menambah pegawai
Obyek uji	Kebutuhan nomer 5 (aliran utama)
Tujuan pengujian	Menguji bahwa sistem dapat menyediakan fasilitas untuk menambah pegawai. Data yang dicatat berupa kode, nama, *username*,*password*, *sex*, tempat lahir, tanggal lahir, alamat, no. telepon, *email*, dan tugas.
Prosedur uji	1. *Use case* melihat daftar pegawai telah dijalankan 2. Memilih menu tambah pegawai. 3. Memasukkan data-data pada *form* (kode= ”ADMIN”, nama= ”Jati Rakhmawati”, *username*= ”teara08”, *password*= ”teara08”, *sex*= ”P”, tempat lahir= ”Tuban”, tanggal lahir= ”8 April 1986”, alamat= ”Jl. Watu Mujur I/4 Malang”, no. telepon= ”081335540408”, *email*=

	”teara08@yahoo.co.id”, tugas= ”Admin”). 4. Menekan tombol simpan.
Hasil yang diharapkan	Pegawai berhasil ditambahkan.

Sumber: [Pengujian]

Tabel 6. 11 Kasus uji menambah pegawai dimana masukkan kode, nama, *username, password, sex,* tempat lahir, alamat, no. telepon, *email*, atau tugas kosong

Nama kasus uji	Kasus uji menambah pegawai dimana masukkan kode, nama, *username, password, sex,* tempat lahir, alamat, no. telepon, *email*, atau tugas kosong
Obyek uji	Kebutuhan nomer 5 (aliran alternatif 1)
Tujuan pengujian	Menguji sistem dapat menampilkan pesan bahwa masukkan kode, nama, *username, password, sex,* tempat lahir, alamat, no. telepon, *email*, atau tugas kosong.
Prosedur uji	1. *Use case* melihat daftar pegawai telah dijalankan 2. Memilih menu tambah pegawai. 3. Memasukkan data-data pada *form* (kode= ””, nama= ””, *username*= ””, *password*= ””, *sex*= ””, tempat lahir= ””, tanggal lahir= ””, alamat= ””, no. telepon= ””, *email*= ””, tugas= ””). 4. Menekan tombol simpan.
Hasil yang diharapkan	Sistem kembali ke *form* dan menampilkan pesan bahwa masukkan kode, nama, *username, password, sex,* tempat lahir, alamat, no. telepon, *email*, atau tugas kosong.

Sumber: [Pengujian]

Tabel 6. 12 Kasus uji menambah pegawai dimana kode, *username,* atau *email* tersebut telah ada di dalam sistem

Nama kasus uji	Kasus uji menambah pegawai dimana kode, *username,* atau *email* tersebut telah ada di dalam sistem
Obyek uji	Kebutuhan nomer 5 (aliran alternatif 2)
Tujuan pengujian	Menguji sistem dapat menampilkan pesan kesalahan bahwa kode, *username,* atau *email* tersebut telah ada di dalam sistem.
Prosedur uji	1. *Use case* melihat daftar pegawai telah dijalankan 2. Memilih menu tambah pegawai. 3. Memasukkan data-data pada *form* (kode= ”ADMIN”, nama= ”Jati Rakhmawati”, *username*= ”teara08”, *password*= ”teara08”, *sex*= ”P”, tempat lahir= ”Tuban”, tanggal lahir= ”8 April 1986”, alamat= ”Jl. Watu Mujur I/4 Malang”, no. telepon= ”081335540408”, *email*= ”teara08@yahoo.co.id”, tugas= ”Admin”). 4. Menekan tombol simpan.
Hasil yang diharapkan	Sistem menampilkan pesan kesalahan bahwa kode, *username,* atau *email* tersebut telah ada di dalam sistem.

Sumber: [Pengujian]

6. Kasus Uji Mengubah Data Pegawai

Tabel 6.13, 6.14 dan 6.15 menunjukkan kasus uji mengubah data pegawai.

Tabel 6. 13 Kasus uji mengubah data pegawai

Nama kasus uji	Kasus uji mengubah data pegawai
Obyek uji	Kebutuhan nomer 6 (aliran utama)
Tujuan pengujian	Menguji bahwa sistem dapat menyediakan fasilitas untuk mengubah data pegawai. Data yang dapat diubah adalah kode, nama, tempat lahir, tanggal lahir, alamat, no. telepon, dan *email.*
Prosedur uji	1. *Use case* melihat daftar pegawai telah dijalankan 2. Memilih tombol ubah pada salah satu baris. 3. Memasukkan data-data pada *form* (kode= ”ADMIN”, nama= ”Jati Rakhmawati”, tempat lahir= ”Tuban”, tanggal lahir= ”8 April 1986”, alamat= ”Jl. Watu Mujur I/4 Malang”, no. telepon= ”081335540408”, *email*= ”teara08@yahoo.co.id”).

	4. Menekan tombol simpan.
Hasil yang diharapkan	Data pegawai dapat diubah dan disimpan.

Sumber: [Pengujian]

Tabel 6. 14 Kasus uji mengubah data pegawai dimana masukkan kode, nama, tempat lahir, alamat, no. telepon, atau *email* kosong

Nama kasus uji	Kasus uji mengubah data pegawai dimana masukkan kode, nama, tempat lahir, alamat, no. telepon, atau *email* kosong
Obyek uji	Kebutuhan nomer 6 (aliran alternatif 1)
Tujuan pengujian	Menguji sistem dapat menampilkan pesan bahwa masukkan kode, nama, tempat lahir, alamat, no. telepon, atau *email* kosong.
Prosedur uji	1. *Use case* melihat daftar pegawai telah dijalankan 2. Memilih tombol ubah pada salah satu baris. 3. Memasukkan data-data pada *form* (kode= ””, nama= ””, *username*= ””, *password*= ””, *sex*= ””, tempat lahir= ””, tanggal lahir= ””, alamat= ””, no. telepon= ””, *email*= ””, tugas= ””). 4. Menekan tombol simpan.
Hasil yang diharapkan	Sistem kembali ke *form* dan menampilkan pesan bahwa masukkan kode, nama, tempat lahir, alamat, no. telepon, atau *email* kosong.

Sumber: [Pengujian]

Tabel 6. 15 Kasus uji mengubah data pegawai dimana kode, *username*, atau *email* tersebut telah ada di dalam sistem

Nama kasus uji	Kasus uji mengubah data pegawai dimana kode, *username*, atau *email* tersebut telah ada di dalam sistem
Obyek uji	Kebutuhan nomer 6 (aliran alternatif 2)
Tujuan pengujian	Menguji sistem dapat menampilkan pesan kesalahan bahwa kode, *username*, atau *email* tersebut telah ada di dalam sistem.
Prosedur uji	1. *Use case* melihat daftar pegawai telah dijalankan 2. Memilih tombol ubah pada salah satu baris. 3. Memasukkan data-data pada *form* (kode= ”ADMIN”, nama= ”Jati Rakhmawati”, tempat lahir= ”Tuban”, tanggal lahir= ”8 April 1986”, alamat= ”Jl. Watu Mujur I/4 Malang”, no. telepon= ”081335540408”, *email*= ”teara08@yahoo.co.id”). 4. Menekan tombol simpan.
Hasil yang diharapkan	Sistem menampilkan pesan kesalahan bahwa kode, *username*, atau *email* tersebut telah ada di dalam sistem.

Sumber: [Pengujian]

7. Kasus Uji Menghapus Pegawai

Tabel 6.16 menunjukkan kasus uji menghapus pegawai.

Tabel 6. 16 Kasus uji menghapus pegawai

Nama kasus uji	Kasus uji menghapus pegawai
Obyek uji	Kebutuhan nomer 7
Tujuan pengujian	Menguji bahwa sistem dapat menghapus data pegawai tertentu.
Prosedur uji	1. *Use case* melihat daftar pegawai telah dijalankan. 2. Memilih tombol hapus pada salah satu baris pegawai.
Hasil yang diharapkan	Data pegawai dapat dihapus.

Sumber: [Pengujian]

8. Kasus Uji Mencari Pegawai

Tabel 6.17 dan 6.18 menunjukkan kasus uji mencari pegawai.

Tabel 6. 17 Kasus uji mencari pegawai

Nama kasus uji	Kasus uji mencari pegawai
Obyek uji	Kebutuhan nomer 8 (aliran utama)

Tujuan pengujian	Menguji bahwa sistem dapat mencari pegawai tertentu yang ada di dalam sistem. Pencarian pegawai berdasarkan nama pegawai.
Prosedur uji	1. *Use case* melihat daftar pegawai telah dijalankan. 2. Memasukkan nama pegawai. 3. Menekan tombol cari.
Hasil yang diharapkan	Seorang pegawai berhasil ditampilkan.

Sumber: [Pengujian]

Tabel 6. 18 Kasus uji mencari pegawai dimana masukkan *keyword* kosong

Nama kasus uji	Kasus uji mencari pegawai dimana masukkan *keyword* kosong
Obyek uji	Kebutuhan nomer 8 (aliran alternatif 1)
Tujuan pengujian	Menguji bahwa sistem dapat menampilkan pesan bahwa *keyword* pencarian tidak boleh kosong.
Prosedur uji	1. *Use case* melihat daftar pegawai telah dijalankan. 2. Menekan tombol cari (*keyword* kosong).
Hasil yang diharapkan	Sistem menampilkan pesan bahwa *keyword* pencarian tidak boleh kosong.

Sumber: [Pengujian]

9. Kasus Uji Melihat Daftar Barang

Tabel 6.19 menunjukkan kasus uji melihat daftar barang.

Tabel 6. 19 Kasus uji melihat daftar barang

Nama kasus uji	Kasus uji melihat daftar barang
Obyek uji	Kebutuhan nomer 9
Tujuan pengujian	Menguji bahwa sistem dapat menampilkan semua barang yang tercatat di dalam sistem. Data yang ditampilkan berupa kode, nama, dan harga.
Prosedur uji	1. Sistem telah berjalan dan pengguna telah login sebagai administrator. 2. Memilih menu barang.
Hasil yang diharapkan	Semua daftar barang dapat ditampilkan.

Sumber: [Pengujian]

10. Kasus Uji Melihat Data Barang

Tabel 6.20 menunjukkan kasus uji melihat data barang.

Tabel 6. 20 Kasus uji melihat data barang

Nama kasus uji	Kasus uji melihat data barang
Obyek uji	Kebutuhan nomer 10
Tujuan pengujian	Menguji bahwa sistem dapat menampilkan data barang yang tercatat di dalam sistem. Data yang ditampilkan berupa kode, nama, kategori, gambar, harga, suplier, dan deskripsi.
Prosedur uji	1. *Use case* melihat daftar barang telah dijalankan. 2. Memilih kode salah satu barang dari daftar barang.
Hasil yang diharapkan	Data barang dapat ditampilkan.

Sumber: [Pengujian]

11. Kasus Uji Menambah Barang

Tabel 6.21, 6.22, dan 6.23 menunjukkan kasus uji menambah barang.

Tabel 6. 21 Kasus uji menambah barang

Nama kasus uji	Kasus uji menambah barang
Obyek uji	Kebutuhan nomer 11 (aliran utama)
Tujuan pengujian	Menguji bahwa sistem dapat menyediakan fasilitas untuk menambah barang. Data yang dicatat berupa kode, nama, kategori, gambar, harga, suplier, dan deskripsi.
Prosedur uji	1. *Use case* melihat daftar barang telah dijalankan

	2. Memilih menu tambah barang. 3. Memasukkan data-data pada *form* (kode= ”001”, nama= ”Ponds Flawless”, kategori= ”Kosmetik”, gambar = ”x.jpg”, harga =”29000”, suplier= ”Unilever Indonesia”, dan deskripsi= ”pelembab dan pemutih”). 4. Menekan tombol simpan.
Hasil yang diharapkan	Barang berhasil ditambahkan.

Sumber: [Pengujian]

Tabel 6. 22 Kasus uji menambah barang dimana masukkan kode, nama, atau harga kosong

Nama kasus uji	Kasus uji menambah barang dimana masukkan kode, nama atau harga kosong
Obyek uji	Kebutuhan nomer 11 (aliran alternatif 1)
Tujuan pengujian	Menguji sistem dapat menampilkan pesan bahwa masukkan kode, nama, atau harga kosong.
Prosedur uji	1. *Use case* melihat daftar barang telah dijalankan 2. Memilih menu tambah barang. 3. Memasukkan data-data pada *form* (kode= ””, nama= ””, kategori= ”Kosmetik”, gambar = ”x.jpg”, harga =””, suplier= ”Unilever Indonesia”, dan deskripsi= ”pelembab dan pemutih”). 4. Menekan tombol simpan.
Hasil yang diharapkan	Sistem kembali ke *form* dan menampilkan pesan bahwa masukkan kode, nama, atau harga kosong.

Sumber: [Pengujian]

Tabel 6. 23 Kasus uji menambah barang dimana kode tersebut telah ada di dalam sistem

Nama kasus uji	Kasus uji menambah barang dimana kode tersebut telah ada di dalam sistem
Obyek uji	Kebutuhan nomer 11 (aliran alternatif 2)
Tujuan pengujian	Menguji sistem dapat menampilkan pesan kesalahan bahwa kode tersebut telah ada di dalam sistem.
Prosedur uji	1. *Use case* melihat daftar barang telah dijalankan 2. Memilih menu tambah barang. 3. Memasukkan data-data pada *form* (kode= ”001”, nama= ”Ponds Flawless”, kategori= ”Kosmetik”, gambar = ”x.jpg”, harga =”29000”, suplier= ”Unilever Indonesia”, dan deskripsi= ”pelembab dan pemutih”). 4. Menekan tombol simpan.
Hasil yang diharapkan	Sistem menampilkan pesan kesalahan bahwa kode tersebut telah ada di dalam sistem.

Sumber: [Pengujian]

12. Kasus Uji Menambah Stok Barang

Tabel 6.24 menunjukkan kasus uji menambah stok barang.

Tabel 6. 24 Kasus uji menambah stok barang

Nama kasus uji	Kasus uji menambah stok barang
Obyek uji	Kebutuhan nomer 12
Tujuan pengujian	Menguji bahwa sistem dapat menyediakan fasilitas untuk menambah stok barang. Data yang dicatatstok tambah.
Prosedur uji	1. *Use case* melihat daftar barang telah dijalankan 2. Memilih tombol stok pada salah satu baris barang. 3. Memasukkan data *form* (stok tambah= ”100”). 4. Menekan tombol tambah.
Hasil yang diharapkan	Stok barang berhasil ditambahkan.

Sumber: [Pengujian]

13. Kasus Uji Mengubah Data Barang

Tabel 6.25, 6.26, dan tabel 6.27 menunjukkan kasus uji mengubah data barang.

Tabel 6. 25 Kasus uji mengubah data barang

Nama kasus uji	Kasus uji mengubah data barang
Obyek uji	Kebutuhan nomer 13 (aliran utama)
Tujuan pengujian	Menguji bahwa sistem dapat menyediakan fasilitas untuk mengubah data barang. Data yang dapat diubah adalah kode, nama, kategori, gambar, harga, suplier, dan deskripsi.
Prosedur uji	1. *Use case* melihat daftar barang telah dijalankan 2. Memilih tombol ubah pada salah satu baris. 3. Memasukkan data-data pada *form* (kode= ”001”, nama= ”Ponds Flawless”, kategori= ”Kosmetik”, gambar = ”x.jpg”, harga =”29000”, suplier= ”Unilever Indonesia”, dan deskripsi= ”pelembab dan pemutih”). 4. Menekan tombol simpan.
Hasil yang diharapkan	Data barang dapat diubah dan disimpan.

Sumber: [Pengujian]

Tabel 6. 26 Kasus uji mengubah data barang dimana masukkan kode, nama, atau harga kosong

Nama kasus uji	Kasus uji mengubah data barang dimana masukkan kode, nama atau harga kosong
Obyek uji	Kebutuhan nomer 13 (aliran alternatif 1)
Tujuan pengujian	Menguji sistem dapat menampilkan pesan bahwa masukkan kode, nama atau harga kosong.
Prosedur uji	1. *Use case* melihat daftar barang telah dijalankan 2. Memilih tombol ubah pada salah satu baris. 3. Memasukkan data-data pada *form* (kode= ””, nama= ””, kategori= ”Kosmetik”, gambar = ”x.jpg”, harga =””, suplier= ”Unilever Indonesia”, dan deskripsi= ”pelembab dan pemutih”). 4. Menekan tombol simpan.
Hasil yang diharapkan	Sistem kembali ke *form* dan menampilkan pesan bahwa masukkan kode, nama, atau harga kosong.

Sumber: [Pengujian]

Tabel 6. 27 Kasus uji mengubah data barang dimana kode tersebut telah ada di dalam sistem

Nama kasus uji	Kasus uji mengubah data barang dimana kode tersebut telah ada di dalam sistem
Obyek uji	Kebutuhan nomer 13 (aliran alternatif 2)
Tujuan pengujian	Menguji sistem dapat menampilkan pesan kesalahan bahwa kode tersebut telah ada di dalam sistem.
Prosedur uji	1. *Use case* melihat daftar barang telah dijalankan 2. Memilih tombol ubah pada salah satu baris. 3. Memasukkan data-data pada *form* (kode= ”001”, nama= ”Ponds Flawless”, kategori= ”Kosmetik”, gambar = ”x.jpg”, harga =”29000”, suplier= ”Unilever Indonesia”, dan deskripsi= ”pelembab dan pemutih”). 4. Menekan tombol simpan.
Hasil yang diharapkan	Sistem menampilkan pesan kesalahan bahwa kode tersebut telah ada di dalam sistem.

Sumber: [Pengujian]

14. Kasus Uji Menghapus Barang

Tabel 6.28 menunjukkan kasus uji menghapus barang.

Tabel 6. 28 Kasus uji menghapus barang

Nama kasus uji	Kasus uji menghapus barang
Obyek uji	Kebutuhan nomer 14
Tujuan pengujian	Menguji bahwa sistem dapat menghapus data barang tertentu.

Prosedur uji	1. *Use case* melihat daftar barang telah dijalankan. 2. Memilih tombol hapus pada salah satu baris barang.
Hasil yang diharapkan	Data barang dapat dihapus.

Sumber: [Pengujian]

15. Kasus Uji Mencari Barang

Tabel 6.29 dan 6.30 menunjukkan kasus uji mencari barang.

Tabel 6. 29 Kasus uji mencari barang

Nama kasus uji	Kasus uji mencari barang
Obyek uji	Kebutuhan nomer 15 (aliran utama)
Tujuan pengujian	Menguji bahwa sistem dapat mencari barang tertentu yang ada di dalam sistem. Pencarian barang berdasarkan nama barang.
Prosedur uji	1. *Use case* melihat daftar barang telah dijalankan. 2. Memasukkan nama barang. 3. Menekan tombol cari.
Hasil yang diharapkan	Suatu barang berhasil ditampilkan.

Sumber: [Pengujian]

Tabel 6. 30 Kasus uji mencari barang dimana masukkan *keyword* kosong

Nama kasus uji	Kasus uji mencari barang dimana masukkan *keyword* kosong
Obyek uji	Kebutuhan nomer 15 (aliran alternatif 1)
Tujuan pengujian	Menguji bahwa sistem dapat menampilkan pesan bahwa *keyword* pencarian tidak boleh kosong.
Prosedur uji	1. *Use case* melihat daftar barang telah dijalankan. 2. Menekan tombol cari (*keyword* kosong).
Hasil yang diharapkan	Sistem menampilkan pesan bahwa *keyword* pencarian tidak boleh kosong.

Sumber: [Pengujian]

16. Kasus Uji Melihat Daftar Kategori Barang

Tabel 6.31 menunjukkan kasus uji kategori barang.

Tabel 6. 31 Kasus uji melihat daftar kategori barang

Nama kasus uji	Kasus uji melihat daftar kategori barang
Obyek uji	Kebutuhan nomer 16
Tujuan pengujian	Menguji bahwa sistem dapat menampilkan semua kategori barang yang tercatat di dalam sistem. Data yang ditampilkan berupa kode dan nama.
Prosedur uji	1. Sistem telah berjalan dan pengguna telah login sebagai administrator. 2. *Use case* melihat daftar barang telah dijalankan. 3. Memilih menu kategori barang.
Hasil yang diharapkan	Semua daftar kategori barang dapat ditampilkan.

Sumber: [Pengujian]

17. Kasus Uji Melihat Data Kategori Barang

Tabel 6.32 menunjukkan kasus uji melihat data kategori barang.

Tabel 6. 32 Kasus uji melihat data kategori barang

Nama kasus uji	Kasus uji melihat data kategori barang
Obyek uji	Kebutuhan nomer 17
Tujuan pengujian	Menguji bahwa sistem dapat menampilkan data kategori barang yang tercatat di dalam sistem. Data yang ditampilkan berupa kode, nama, dan deskripsi.
Prosedur uji	1. Sistem telah berjalan dan pengguna telah login sebagai administrator. 2. *Use case* melihat daftar kategori barang telah dijalankan. 3. Memilih kode salah satu kategori barang dari daftar kategori barang.

Hasil yang diharapkan	Data kategori barang dapat ditampilkan.

Sumber: [Pengujian]

18. Kasus Uji Menambah Kategori Barang

Tabel 6.33, 6.34, dan 6.35 menunjukkan kasus uji menambah kategori barang.

Tabel 6. 33 Kasus uji menambah kategori barang

Nama kasus uji	Kasus uji menambah kategori barang
Obyek uji	Kebutuhan nomer 18 (aliran utama)
Tujuan pengujian	Menguji bahwa sistem dapat menyediakan fasilitas untuk menambah kategori barang. Data yang dicatat berupa kode, nama, dan deskripsi.
Prosedur uji	1. *Use case* melihat daftar kategori barang telah dijalankan 2. Memilih menu tambah kategori barang. 3. Memasukkan data-data pada *form* (kode= ”BUA”, nama= ”Buah”, dan deskripsi= ”segala macam buah-buahan”). 4. Menekan tombol simpan.
Hasil yang diharapkan	Kategori barang berhasil ditambahkan.

Sumber: [Pengujian]

Tabel 6. 34 Kasus uji menambah kategori barang dimana masukkan kode atau nama kosong

Nama kasus uji	Kasus uji menambah kategori barang dimana masukkan kode atau nama kosong
Obyek uji	Kebutuhan nomer 18 (aliran alternatif 1)
Tujuan pengujian	Menguji sistem dapat menampilkan pesan bahwa masukkan kode atau nama kosong.
Prosedur uji	1. *Use case* melihat daftar kategori barang telah dijalankan 2. Memilih menu tambah kategori barang. 3. Memasukkan data-data pada *form* (kode= ””, nama= ””, dan deskripsi= ”segala macam buah-buahan”). 4. Menekan tombol simpan.
Hasil yang diharapkan	Sistem kembali ke *form* dan menampilkan pesan bahwa masukkan kode atau nama kosong.

Sumber: [Pengujian]

Tabel 6. 35 Kasus uji menambah kategori barang dimana kode tersebut telah ada di dalam sistem

Nama kasus uji	Kasus uji menambah kategori barang dimana kode tersebut telah ada di dalam sistem
Obyek uji	Kebutuhan nomer 18 (aliran alternatif 2)
Tujuan pengujian	Menguji bahwa sistem dapat menampilkan pesan kesalahan bahwa kode tersebut telah ada di dalam sistem.
Prosedur uji	1. *Use case* melihat daftar kategori barang telah dijalankan 2. Memilih menu tambah kategori barang. 3. Memasukkan data-data pada *form* (kode= ”BUA”, nama= ”Buah”, dan deskripsi= ”segala macam buah-buahan”). 4. Menekan tombol simpan.
Hasil yang diharapkan	Sistem menampilkan pesan kesalahan bahwa kode tersebut telah ada di dalam sistem.

Sumber: [Pengujian]

19. Kasus Uji Mengubah Data Kategori Barang

Tabel 6.36, 6.37, dan 6.38 menunjukkan kasus uji mengubah data kategori barang.

Tabel 6. 36 Kasus uji mengubah data kategori barang

Nama kasus uji	Kasus uji mengubah data kategori barang
Obyek uji	Kebutuhan nomer 19 (aliran utama)

Tujuan pengujian	Menguji bahwa sistem dapat menyediakan fasilitas untuk mengubah data kategori barang. Data yang dapat diubah adalah kode, nama, dan deskripsi.
Prosedur uji	1. *Use case* melihat daftar kategori barang telah dijalankan 2. Memilih tombol ubah pada salah satu baris. 3. Memasukkan data-data pada *form* (kode= ”BUA”, nama= ”Buah”, dan deskripsi= ”segala macam buah-buahan”). 4. Menekan tombol simpan.
Hasil yang diharapkan	Data kategori barang dapat diubah dan disimpan.

Sumber: [Pengujian]

Tabel 6. 37 Kasus uji mengubah data kategori barang dimana masukkan kode atau nama kosong

Nama kasus uji	Kasus uji mengubah data kategori barang dimana masukkan kode atau nama kosong
Obyek uji	Kebutuhan nomer 19 (aliran alternatif 1)
Tujuan pengujian	Menguji sistem dapat menampilkan pesan bahwa masukkan kode atau nama kosong.
Prosedur uji	1. *Use case* melihat daftar kategori barang telah dijalankan 2. Memilih tombol ubah pada salah satu baris. 3. Memasukkan data-data pada *form* (kode= ””, nama= ””, dan deskripsi= ”segala macam buah-buahan”). 4. Menekan tombol simpan.
Hasil yang diharapkan	Sistem kembali ke *form* dan menampilkan pesan bahwa masukkan kode atau nama kosong.

Sumber: [Pengujian]

Tabel 6. 38 Kasus uji mengubah data kategori barang dimana kode tersebut telah ada di dalam sistem

Nama kasus uji	Kasus uji mengubah data kategori barang dimana kode tersebut telah ada di dalam sistem
Obyek uji	Kebutuhan nomer 19 (aliran alternatif 2)
Tujuan pengujian	Menguji sistem dapat menampilkan pesan kesalahan bahwa kode tersebut telah ada di dalam sistem.
Prosedur uji	1. *Use case* melihat daftar kategori barang telah dijalankan 2. Memilih tombol ubah pada salah satu baris. 3. Memasukkan data-data pada *form* (kode= ”BUA”, nama= ”Buah”, dan deskripsi= ”segala macam buah-buahan”). 4. Menekan tombol simpan.
Hasil yang diharapkan	Sistem menampilkan pesan kesalahan bahwa kode tersebut telah ada di dalam sistem.

Sumber: [Pengujian]

20. Kasus Uji Menghapus Kategori Barang

Tabel 6.39 menunjukkan kasus uji menghapus kategori barang.

Tabel 6. 39 Kasus uji menghapus kategori barang

Nama kasus uji	Kasus uji menghapus kategori barang
Obyek uji	Kebutuhan nomer 20
Tujuan pengujian	Menguji bahwa sistem dapat menghapus data kategori barang tertentu.
Prosedur uji	1. *Use case* melihat daftar kategori barang telah dijalankan. 2. Memilih tombol hapus pada salah satu baris kategori barang.
Hasil yang diharapkan	Data kategori barang dapat dihapus.

Sumber: [Pengujian]

21. Kasus Uji Mencari Kategori Barang

Tabel 6.40 dan 6.41 menunjukkan kasus uji mencari kategori barang.

Tabel 6. 40 Kasus uji mencari kategori barang

Nama kasus uji	Kasus uji mencari kategori barang

Obyek uji	Kebutuhan nomer 21 (aliran utama)
Tujuan pengujian	Menguji bahwa sistem dapat mencari kategori barang tertentu yang ada di dalam sistem. Pencarian kategori barang berdasarkan nama kategori barang.
Prosedur uji	1. *Use case* melihat daftar kategori barang telah dijalankan. 2. Memasukkan nama kategori barang. 3. Menekan tombol cari.
Hasil yang diharapkan	Suatu kategori barang berhasil ditampilkan.

Sumber: [Pengujian]

Tabel 6. 41 Kasus uji mencari kategori barang dimana masukkan *keyword* kosong

Nama kasus uji	Kasus uji mencari kategori barang dimana masukkan *keyword* kosong
Obyek uji	Kebutuhan nomer 21 (aliran alternatif 1)
Tujuan pengujian	Menguji bahwa sistem dapat menampilkan pesan bahwa *keyword* pencarian tidak boleh kosong.
Prosedur uji	1. *Use case* melihat daftar kategori barang telah dijalankan. 2. Menekan tombol cari (*keyword* kosong).
Hasil yang diharapkan	Sistem menampilkan pesan bahwa *keyword* pencarian tidak boleh kosong.

Sumber: [Pengujian]

22. Kasus Uji Melihat Daftar Subkategori Barang

Tabel 6.42 menunjukkan kasus uji subkategori barang.

Tabel 6. 42 Kasus uji melihat daftar subkategori barang

Nama kasus uji	Kasus uji melihat daftar subkategori barang
Obyek uji	Kebutuhan nomer 22
Tujuan pengujian	Menguji bahwa sistem dapat menampilkan subkategori per kategori barang yang tercatat di dalam sistem. Data yang ditampilkan berupa kode dan nama.
Prosedur uji	1. Sistem telah berjalan dan pengguna telah login sebagai administrator. 2. *Use case* melihat daftar kategori barang telah dijalankan. 3. Memilih satu kategori barang.
Hasil yang diharapkan	Daftar subkategori barang per kategori dapat ditampilkan.

Sumber: [Pengujian]

23. Kasus Uji Menambah Subkategori Barang

Tabel 6.43, 6.44, dan 6.45 menunjukkan kasus uji menambah subkategori barang.

Tabel 6. 43 Kasus uji menambah subkategori barang

Nama kasus uji	Kasus uji menambah subkategori barang
Obyek uji	Kebutuhan nomer 23 (aliran utama)
Tujuan pengujian	Menguji bahwa sistem dapat menyediakan fasilitas untuk menambah subkategori barang. Data yang dicatat berupa kode dan nama.
Prosedur uji	1. *Use case* melihat data kategori barang telah dijalankan 2. Memilih menu tambah subkategori barang. 3. Memasukkan data-data pada *form* (kode= ”01” dan nama= ”TV”) 4. Menekan tombol simpan.
Hasil yang diharapkan	Subkategori barang berhasil ditambahkan.

Sumber: [Pengujian]

Tabel 6. 44 Kasus uji menambah subkategori barang dimana masukkan kode atau nama kosong

Nama kasus uji	Kasus uji menambah subkategori barang dimana masukkan kode atau nama kosong
Obyek uji	Kebutuhan nomer 23 (aliran alternatif 1)
Tujuan pengujian	Menguji sistem dapat menampilkan pesan bahwa masukkan kode atau nama kosong.
Prosedur uji	1. *Use case* melihat data kategori barang telah dijalankan

	2. Memilih menu tambah subkategori barang. 3. Memasukkan data-data pada *form* (kode= ”” dan nama= ””). 4. Menekan tombol simpan.
Hasil yang diharapkan	Sistem kembali ke *form* dan menampilkan pesan bahwa masukkan kode atau nama kosong.

Sumber: [Pengujian]

Tabel 6. 45 Kasus uji menambah subkategori barang dimana kode tersebut telah ada di dalam sistem

Nama kasus uji	Kasus uji menambah subkategori barang dimana kode tersebut telah ada di dalam sistem
Obyek uji	Kebutuhan nomer 23 (aliran alternatif 2)
Tujuan pengujian	Menguji bahwa sistem dapat menampilkan pesan kesalahan bahwa kode tersebut telah ada di dalam sistem.
Prosedur uji	1. *Use case* melihat data kategori barang telah dijalankan 2. Memilih menu tambah subkategori barang. 3. Memasukkan data-data pada *form* (kode= ”01” dan nama= ”TV”). 4. Menekan tombol simpan.
Hasil yang diharapkan	Sistem menampilkan pesan kesalahan bahwa kode tersebut telah ada di dalam sistem.

Sumber: [Pengujian]

24. Kasus Uji Mengubah Subkategori Barang

Tabel 6.46, 6.47, dan 6.48 menunjukkan kasus uji mengubah data subkategori barang.

Tabel 6. 46 Kasus uji mengubah data subkategori barang

Nama kasus uji	Kasus uji mengubah data subkategori barang
Obyek uji	Kebutuhan nomer 24 (aliran utama)
Tujuan pengujian	Menguji bahwa sistem dapat menyediakan fasilitas untuk mengubah data subkategori barang. Data yang dapat diubah adalah kode dan nama.
Prosedur uji	1. *Use case* melihat data kategori barang telah dijalankan 2. Memilih tombol ubah pada salah satu baris. 3. Memasukkan data-data pada *form* (kode= ”01”, nama= ”TV”). 4. Menekan tombol simpan.
Hasil yang diharapkan	Data subkategori barang dapat diubah dan disimpan.

Sumber: [Pengujian]

Tabel 6. 47 Kasus uji mengubah data subkategori barang dimana masukkan kode atau nama kosong

Nama kasus uji	Kasus uji mengubah data subkategori barang dimana masukkan kode atau nama kosong
Obyek uji	Kebutuhan nomer 24 (aliran alternatif 1)
Tujuan pengujian	Menguji sistem dapat menampilkan pesan bahwa masukkan kode atau nama kosong.
Prosedur uji	1. *Use case* melihat data kategori barang telah dijalankan 2. Memilih tombol ubah pada salah satu baris. 3. Memasukkan data-data pada *form* (kode= ”” dan nama=””). 4. Menekan tombol simpan.
Hasil yang diharapkan	Sistem kembali ke *form* dan menampilkan pesan bahwa masukkan kode atau nama kosong.

Sumber: [Pengujian]

Tabel 6. 48 Kasus uji mengubah data subkategori barang dimana kode tersebut telah ada di dalam sistem

Nama kasus uji	Kasus uji mengubah data subkategori barang dimana kode tersebut telah ada di dalam sistem
Obyek uji	Kebutuhan nomer 24 (aliran alternatif 2)
Tujuan pengujian	Menguji sistem dapat menampilkan pesan kesalahan bahwa kode tersebut telah ada di dalam sistem.

Prosedur uji	1. *Use case* melihat data kategori barang telah dijalankan 2. Memilih tombol ubah pada salah satu baris. 3. Memasukkan data-data pada *form* (kode= ”01” dan nama= ”TV”). 4. Menekan tombol simpan.
Hasil yang diharapkan	Sistem menampilkan pesan kesalahan bahwa kode tersebut telah ada di dalam sistem.

Sumber: [Pengujian]

25. Kasus Uji Menghapus Subkategori Barang

Tabel 6.49 menunjukkan kasus uji menghapus subkategori barang.

Tabel 6. 49 Kasus uji menghapus subkategori barang

Nama kasus uji	Kasus uji menghapus subkategori barang
Obyek uji	Kebutuhan nomer 25
Tujuan pengujian	Menguji bahwa sistem dapat menghapus data subkategori barang tertentu.
Prosedur uji	1. *Use case* melihat data kategori barang telah dijalankan. 2. Memilih tombol hapus pada salah satu baris subkategori barang.
Hasil yang diharapkan	Data subkategori barang dapat dihapus.

Sumber: [Pengujian]

26. Kasus Uji Melihat Daftar Suplier

Tabel 6.50 menunjukkan kasus uji melihat daftar suplier.

Tabel 6. 50 Kasus uji melihat daftar suplier

Nama kasus uji	Kasus uji melihat daftar suplier
Obyek uji	Kebutuhan nomer 26
Tujuan pengujian	Menguji bahwa sistem dapat menampilkan semua suplier yang tercatat di dalam sistem. Data yang ditampilkan berupa kode, nama, dan no. telp.
Prosedur uji	1. Sistem telah berjalan dan pengguna telah login sebagai administrator. 2. Memilih menu suplier.
Hasil yang diharapkan	Semua daftar suplier dapat ditampilkan.

Sumber: [Pengujian]

27. Kasus Uji Melihat Data Suplier

Tabel 6.51 menunjukkan kasus uji melihat data suplier.

Tabel 6. 51 Kasus uji melihat data suplier

Nama kasus uji	Kasus uji melihat data suplier
Obyek uji	Kebutuhan nomer 27
Tujuan pengujian	Menguji bahwa sistem dapat menampilkan data suplier yang tercatat di dalam sistem. Data yang ditampilkan berupa kode, nama, alamat, dan no. telp.
Prosedur uji	1. Sistem telah berjalan dan pengguna telah login sebagai administrator. 2. *Use case* melihat daftar suplier telah dijalankan. 3. Memilih kode salah satu suplier dari daftar suplier.
Hasil yang diharapkan	Data suplier dapat ditampilkan.

Sumber: [Pengujian]

28. Kasus Uji Menambah Suplier

Tabel 6.52, 6.53, dan 6.54 menunjukkan kasus uji menambah suplier.

Tabel 6. 52 Kasus uji menambah suplier

Nama kasus uji	Kasus uji menambah suplier
Obyek uji	Kebutuhan nomer 28 (aliran utama)

Tujuan pengujian	Menguji bahwa sistem dapat menyediakan fasilitas untuk menambah suplier. Data yang dicatat berupa kode, nama, alamat, dan no. telp.
Prosedur uji	1. *Use case* melihat daftar suplier telah dijalankan 2. Memilih menu tambah suplier. 3. Memasukkan data-data pada *form* (kode= ”SUP001”, nama= ”Unilever Indonesia”, alamat= ”Jl. Raya Malang-Surabaya”, no. telp= ”0341-999999”). 4. Menekan tombol simpan.
Hasil yang diharapkan	Suplier berhasil ditambahkan.

Sumber: [Pengujian]

Tabel 6. 53 Kasus uji menambah suplier dimana masukkan kode, nama, alamat, atau no.telp kosong

Nama kasus uji	Kasus uji menambah suplier barang dimana masukkan kode, nama, alamat, atau no.telp kosong
Obyek uji	Kebutuhan nomer 28 (aliran alternatif 1)
Tujuan pengujian	Menguji sistem dapat menampilkan pesan bahwa masukkan kode, nama, alamat, atau no.telp kosong.
Prosedur uji	1. *Use case* melihat daftar suplier telah dijalankan 2. Memilih menu tambah suplier. 3. Memasukkan data-data pada *form* (kode= ””, nama= ””, alamat= ””, no. telp= ””). 4. Menekan tombol simpan.
Hasil yang diharapkan	Sistem kembali ke *form* dan menampilkan pesan bahwa masukkan kode, nama, alamat, atau no.telp kosong.

Sumber: [Pengujian]

Tabel 6. 54 Kasus uji menambah suplier dimana kode tersebut telah ada di dalam sistem

Nama kasus uji	Kasus uji menambah suplier dimana kode tersebut telah ada di dalam sistem
Obyek uji	Kebutuhan nomer 28 (aliran alternatif 2)
Tujuan pengujian	Menguji bahwa sistem dapat menampilkan pesan kesalahan bahwa kode tersebut telah ada di dalam sistem.
Prosedur uji	1. *Use case* melihat daftar suplier telah dijalankan 2. Memilih menu tambah suplier. 3. Memasukkan data-data pada *form* (kode= ”SUP001”, nama= ”Unilever Indonesia”, alamat= ”Jl. Raya Malang-Surabaya”, no. telp= ”0341-999999”). 4. Menekan tombol simpan.
Hasil yang diharapkan	Sistem menampilkan pesan kesalahan bahwa kode tersebut telah ada di dalam sistem.

Sumber: [Pengujian]

29. Kasus Uji Mengubah Data Suplier

Tabel 6.55, 6.56 dan 6.57 menunjukkan kasus uji mengubah data suplier.

Tabel 6. 55 Kasus uji mengubah data suplier

Nama kasus uji	Kasus uji mengubah data suplier
Obyek uji	Kebutuhan nomer 29 (aliran utama)
Tujuan pengujian	Menguji bahwa sistem dapat menyediakan fasilitas untuk mengubah data suplier. Data yang dapat diubah adalah kode, nama, alamat, dan no.telp.
Prosedur uji	1. *Use case* melihat daftar suplier telah dijalankan 2. Memilih tombol ubah pada salah satu baris. 3. Memasukkan data-data pada *form* (kode= ”SUP001”, nama= ”Unilever Indonesia”, alamat= ”Jl. Raya Malang-Surabaya”, no. telp= ”0341-999999”). 4. Menekan tombol simpan.
Hasil yang diharapkan	Data suplier dapat diubah dan disimpan.

Sumber: [Pengujian]

Tabel 6. 56 Kasus uji mengubah data suplier dimana masukkan kode, nama, alamat, atau no.telp kosong

Nama kasus uji	Kasus uji mengubah data suplier dimana masukkan kode, nama, alamat, atau no.telp kosong
Obyek uji	Kebutuhan nomer 29 (aliran alternatif 1)
Tujuan pengujian	Menguji sistem dapat menampilkan pesan bahwa masukkan kode, nama, alamat, atau no.telp kosong.
Prosedur uji	1. *Use case* melihat daftar suplier telah dijalankan. 2. Memilih tombol ubah pada salah satu baris. 3. Memasukkan data-data pada *form* (kode= ””, nama= ””, alamat= ””, no. telp= ””). 4. Menekan tombol simpan.
Hasil yang diharapkan	Sistem kembali ke *form* dan menampilkan pesan bahwa masukkan kode, nama, alamat, atau no.telp kosong.

Sumber: [Pengujian]

Tabel 6. 57 Kasus uji mengubah data suplier dimana kode tersebut telah ada di dalam sistem

Nama kasus uji	Kasus uji mengubah data suplier dimana kode tersebut telah ada di dalam sistem
Obyek uji	Kebutuhan nomer 29 (aliran alternatif 2)
Tujuan pengujian	Menguji sistem dapat menampilkan pesan kesalahan bahwa kode tersebut telah ada di dalam sistem.
Prosedur uji	1. *Use case* melihat daftar suplier telah dijalankan 2. Memilih tombol ubah pada salah satu baris. 3. Memasukkan data-data pada *form* (kode= ”SUP001”, nama= ”Unilever Indonesia”, alamat= ”Jl. Raya Malang-Surabaya”, no. telp= ”0341-999999”). 4. Menekan tombol simpan.
Hasil yang diharapkan	Sistem menampilkan pesan kesalahan bahwa kode tersebut telah ada di dalam sistem.

Sumber: [Pengujian]

30. Kasus Uji Menghapus Suplier

Tabel 6.58 menunjukkan kasus uji menghapus suplier.

Tabel 6. 58 Kasus uji menghapus suplier

Nama kasus uji	Kasus uji menghapus suplier
Obyek uji	Kebutuhan nomer 30
Tujuan pengujian	Menguji bahwa sistem dapat menghapus data suplier tertentu.
Prosedur uji	1. *Use case* melihat daftar suplier telah dijalankan. 2. Memilih tombol hapus pada salah satu baris suplier.
Hasil yang diharapkan	Data suplier dapat dihapus.

Sumber: [Pengujian]

31. Kasus Uji Mencari Suplier

Tabel 6.59 dan 6.60 menunjukkan kasus uji mencari suplier.

Tabel 6. 59 Kasus uji mencari suplier

Nama kasus uji	Kasus uji mencari suplier
Obyek uji	Kebutuhan nomer 31 (aliran utama)
Tujuan pengujian	Menguji bahwa sistem dapat mencari suplier tertentu yang ada di dalam sistem. Pencarian suplier berdasarkan nama suplier.
Prosedur uji	1. *Use case* melihat daftar suplier telah dijalankan. 2. Memasukkan nama suplier. 3. Menekan tombol cari.
Hasil yang diharapkan	Suatu suplier berhasil ditampilkan.

Sumber: [Pengujian]

Tabel 6. 60 Kasus uji mencari suplier dimana masukkan *keyword* kosong

Nama kasus uji	Kasus uji mencari suplier dimana masukkan *keyword* kosong
Obyek uji	Kebutuhan nomer 31 (aliran alternatif 1)
Tujuan pengujian	Menguji bahwa sistem dapat menampilkan pesan bahwa *keyword* pencarian tidak boleh kosong.
Prosedur uji	1. *Use case* melihat daftar suplier telah dijalankan. 2. Menekan tombol cari (*keyword* kosong).
Hasil yang diharapkan	Sistem menampilkan pesan bahwa *keyword* pencarian tidak boleh kosong.

Sumber: [Pengujian]

32. Kasus Uji Melihat Daftar Konsumen

Tabel 6.61 menunjukkan kasus uji melihat daftar konsumen.

Tabel 6. 61 Kasus uji melihat daftar konsumen

Nama kasus uji	Kasus uji melihat daftar konsumen
Obyek uji	Kebutuhan nomer 32
Tujuan pengujian	Menguji bahwa sistem dapat menampilkan semua konsumen yang tercatat di dalam sistem. Data yang ditampilkan berupa nama dan status validasi.
Prosedur uji	1. Sistem telah berjalan dan pengguna telah login sebagai administrator. 2. Memilih menu konsumen.
Hasil yang diharapkan	Semua daftar konsumen dapat ditampilkan.

Sumber: [Pengujian]

33. Kasus Uji Menghapus Konsumen

Tabel 6.62 menunjukkan kasus uji menghapus konsumen.

Tabel 6. 62 Kasus uji menghapus konsumen

Nama kasus uji	Kasus uji menghapus konsumen
Obyek uji	Kebutuhan nomer 33
Tujuan pengujian	Menguji bahwa sistem dapat menghapus data konsumen tertentu.
Prosedur uji	1. *Use case* melihat daftar konsumen telah dijalankan. 2. Memilih tombol hapus pada salah satu baris konsumen.
Hasil yang diharapkan	Data konsumen dapat dihapus.

Sumber: [Pengujian]

34. Kasus Uji Mencari Konsumen

Tabel 6.63 dan 6.64 menunjukkan kasus uji mencari konsumen.

Tabel 6. 63 Kasus uji mencari konsumen

Nama kasus uji	Kasus uji mencari konsumen
Obyek uji	Kebutuhan nomer 34 (aliran utama)
Tujuan pengujian	Menguji bahwa sistem dapat mencari konsumen tertentu yang ada di dalam sistem. Pencarian suplier berdasarkan status validasi.
Prosedur uji	1. *Use case* melihat daftar konsumen telah dijalankan. 2. Memasukkan status validasi. 3. Menekan tombol cari.
Hasil yang diharapkan	Daftar konsumen dengan suatu status validasi berhasil ditampilkan.

Sumber: [Pengujian]

Tabel 6. 64 Kasus uji mencari konsumen dimana masukkan *keyword* kosong

Nama kasus uji	Kasus uji mencari konsumen dimana masukkan *keyword* kosong
Obyek uji	Kebutuhan nomer 34 (aliran alternatif 1)
Tujuan pengujian	Menguji bahwa sistem dapat menampilkan pesan bahwa *keyword* pencarian tidak boleh kosong.
Prosedur uji	1. *Use case* melihat daftar konsumen telah dijalankan.

	2. Menekan tombol cari (*keyword* kosong).
Hasil yang diharapkan	Sistem menampilkan pesan bahwa *keyword* pencarian tidak boleh kosong.

Sumber: [Pengujian]

35. Kasus Uji Melihat Daftar Validasi Konsumen

Tabel 6.65 menunjukkan kasus uji melihat daftar validasi konsumen.

Tabel 6. 65 Kasus uji melihat daftar validasi konsumen

Nama kasus uji	Kasus uji melihat daftar validasi konsumen
Obyek uji	Kebutuhan nomer 35
Tujuan pengujian	Menguji bahwa sistem dapat menampilkan semua konsumen yang tercatat di dalam sistem. Data yang ditampilkan berupa nama, *no.handphone*, dan status validasi.
Prosedur uji	1. Sistem telah berjalan dan pengguna telah login sebagai operator. 2. Memilih menu validasi konsumen.
Hasil yang diharapkan	Semua daftar validasi konsumen dapat ditampilkan.

Sumber: [Pengujian]

36. Kasus Uji Mengubah Data Konsumen

Tabel 6.66 menunjukkan kasus uji mengubah data konsumen.

Tabel 6. 66 Kasus uji mengubah data konsumen

Nama kasus uji	Kasus uji mengubah data konsumen
Obyek uji	Kebutuhan nomer 36
Tujuan pengujian	Menguji bahwa sistem dapat menyediakan fasilitas untuk mengubah data konsumen. Data yang dapat diubah adalah status validasi.
Prosedur uji	1. *Use case* melihat daftar validasi konsumen telah dijalankan 2. Memilih menu salah satu status validasi dari daftar validasi konsumen. 3. Memasukkan data-data pada *form* (status validasi= ”Valid”). 4. Menekan tombol simpan.
Hasil yang diharapkan	Data konsumen dapat diubah dan disimpan.

Sumber: [Pengujian]

37. Kasus Uji Memesan Barang

Tabel 6.67 menunjukkan kasus uji memesan barang.

Tabel 6. 67 Kasus uji memesan barang

Nama kasus uji	Kasus uji memesan barang
Obyek uji	Kebutuhan nomer 37
Tujuan pengujian	Menguji bahwa sistem dapat menyediakan fasilitas untuk memesan barang.
Prosedur uji	1. Sistem telah berjalan dan pengguna telah login sebagai konsumen. 2. Memilih kategori dari daftar kategori barang. 3. Memilih barang dari daftar barang per kategori.
Hasil yang diharapkan	Data barang yang dipesan disimpan pada keranjang belanja.

Sumber: [Pengujian]

38. Kasus Uji Melihat Keranjang Belanja

Tabel 6.68 menunjukkan kasus uji melihat keranjang belanja.

Tabel 6. 68 Kasus uji melihat keranjang belanja

Nama kasus uji	Kasus uji melihat keranjang belanja
Obyek uji	Kebutuhan nomer 38
Tujuan pengujian	Menguji bahwa sistem dapat menampilkan semua barang yang tercatat di dalam keranjang belanja. Data barang yang ditampilkan berupa nama, harga,

	jumlah, dan subtotal.
Prosedur uji	1. *Use case* memesan barang telah dijalankan. 2. Memilih tombol keranjang belanja.
Hasil yang diharapkan	Semua daftar barang yang dipesan dapat ditampilkan dan dapat diubah.

Sumber: [Pengujian]

39. Kasus Uji Mengubah Data Keranjang Belanja

Tabel 6.69 menunjukkan kasus uji mengubah data keranjang belanja.

Tabel 6. 69 Kasus uji mengubah data keranjang belanja

Nama kasus uji	Kasus uji mengubah data keranjang belanja
Obyek uji	Kebutuhan nomer 39
Tujuan pengujian	Menguji bahwa sistem dapat menyediakan fasilitas untuk mengubah data keranjang belanja. Data yang dapat diubah adalah jumlah barang.
Prosedur uji	1. *Use case* melihat keranjang belanja telah dijalankan. 2. Memasukkan data pada *form* (jumlah= "8")
Hasil yang diharapkan	Data keranjang belanja dapat diubah dan disimpan.

Sumber: [Pengujian]

40. Kasus Uji Menghapus Keranjang Belanja

Tabel 6.70 menunjukkan kasus uji menghapus keranjang belanja.

Tabel 6. 70 Kasus uji menghapus keranjang belanja

Nama kasus uji	Kasus uji menghapus keranjang belanja
Obyek uji	Kebutuhan nomer 40
Tujuan pengujian	Menguji bahwa sistem dapat menghapus data barang dalam keranjang belanja.
Prosedur uji	1. *Use case* melihat keranjang belanja telah dijalankan. 2. Memilih tombol hapus pada salah satu baris barang.
Hasil yang diharapkan	Data konsumen dapat dihapus.

Sumber: [Pengujian]

41. Kasus Uji Melihat Detail Pesanan

Tabel 6.71 menunjukkan kasus uji melihat detail pesanan.

Tabel 6. 71 Kasus uji melihat detail pesanan

Nama kasus uji	Kasus uji melihat melihat detail pesanan
Obyek uji	Kebutuhan nomer 41
Tujuan pengujian	Menguji bahwa sistem dapat menampilkan semua detail pesanan yang tercatat di dalam sistem. Detail pesanan yang ditampilkan berupa nama, harga, jumlah, dan subtotal.
Prosedur uji	1. *Use case* melihat keranjang belanja telah dijalankan. 2. Memilih tombol detail pesanan.
Hasil yang diharapkan	Semua daftar barang yang dipesan dapat ditampilkan.

Sumber: [Pengujian]

42. Kasus Uji Check Out

Tabel 6.72 menunjukkan kasus uji check out.

Tabel 6. 72 Kasus uji check out

Nama kasus uji	Kasus uji check out
Obyek uji	Kebutuhan nomer 42
Tujuan pengujian	Menguji bahwa sistem dapat melakukan check out dan menampilkan data check out. Data yang ditampilkan berupa kode transaksi, waktu pesan,

	nama, alamat, no.handphone, dan total belanja.
Prosedur uji	1. *Use case* melihat detail pesanan telah dijalankan. 2. Memilih tombol check out.
Hasil yang diharapkan	Data check out dapat ditampilkan dan disimpan.

Sumber: [Pengujian]

43. Kasus Uji Melihat Daftar Pesanan

Tabel 6.73 menunjukkan kasus uji melihat daftar pesanan.

Tabel 6. 73 Kasus uji melihat daftar pesanan

Nama kasus uji	Kasus uji melihat daftar pesanan
Obyek uji	Kebutuhan nomer 43
Tujuan pengujian	Menguji bahwa sistem dapat menampilkan semua pesanan yang tercatat di dalam sistem. Data yang ditampilkan berupa kode transaksi, waktu pesan, dan status kirim.
Prosedur uji	1. Sistem telah berjalan dan pengguna telah login sebagai operator. 2. Memilih tombol daftar pesanan.
Hasil yang diharapkan	Semua daftar pesanan dapat ditampilkan.

Sumber: [Pengujian]

44. Kasus Uji Mencetak Nota

Tabel 6.74 menunjukkan kasus uji mencetak nota.

Tabel 6. 74 Kasus uji mencetak nota

Nama kasus uji	Kasus uji mencetak nota
Obyek uji	Kebutuhan nomer 44
Tujuan pengujian	Menguji bahwa sistem dapat mencetak nota.
Prosedur uji	1. *Use case* melihat daftar pesanan telah dijalankan. 2. Memilih status kirim pada satu baris dari daftar pesanan. 3. Mengubah status kirim menjadi proses. 4. Memilih tombol simpan.
Hasil yang diharapkan	Nota dapat dicetak.

Sumber: [Pengujian]

6.3.2. Hasil Pengujian Validasi dan Analisis

Dari kasus uji yang telah dilaksanakan sesuai dengan prosedur pengujian pada sub pokok bahasan 6.3.1, didapatkan hasil seperti ditunjukkan pada Tabel 6.75.

Tabel 6. 75 Hasil pengujian validasi

No.	Kasus Uji	Hasil Yang Didapatkan	Status
1.	Kasus uji login	Pengguna berhasil login pada sistem.	Valid
2.	Kasus uji *login* dimana pasangan *username* dan *password* tidak ada dalam sistem	Kembali ke halaman *login*.	Valid
3.	Kasus uji logout	Pengguna berhasil *logout* dari sistem.	Valid
4.	Kasus uji melihat daftar pegawai	Semua daftar pegawai dapat ditampilkan.	Valid
5.	Kasus uji melihat data pegawai	Data pegawai dapat ditampilkan.	Valid
6.	Kasus uji menambah pegawai	Pegawai berhasil ditambahkan.	Valid
7.	Kasus uji menambah pegawai dimana masukkan kode, nama, *username*, *password*, *sex*, tempat lahir, alamat, no.	Sistem kembali ke *form* dan menampilkan pesan bahwa masukkan kode, nama, *username*, *password*, *sex*, tempat lahir, alamat, no. telepon, *email*, atau tugas kosong.	Valid

	telepon, *email*, atau tugas kosong		
8.	Kasus uji menambah pegawai dimana kode, *username*, atau *email* tersebut telah ada di dalam sistem	Sistem menampilkan pesan kesalahan bahwa kode, *username*, atau *email* tersebut telah ada di dalam sistem.	Valid
9.	Kasus uji mengubah data pegawai	Data pegawai dapat diubah dan disimpan.	Valid
10.	Kasus uji mengubah data pegawai dimana masukkan kode, nama, tempat lahir, alamat, no. telepon, atau *email* kosong	Sistem kembali ke *form* dan menampilkan pesan bahwa masukkan kode, nama, tempat lahir, alamat, no. telepon, atau *email* kosong.	Valid
11.	Kasus uji mengubah data pegawai dimana kode, *username*, atau *email* tersebut telah ada di dalam sistem	Sistem menampilkan pesan kesalahan bahwa kode, *username*, atau *email* tersebut telah ada di dalam sistem.	Valid
12.	Kasus uji menghapus pegawai	Data pegawai dapat dihapus.	Valid
13.	Kasus uji mencari pegawai	Seorang pegawai berhasil ditampilkan.	Valid
14.	Kasus uji mencari pegawai dimana masukkan *keyword* kosong	Sistem menampilkan pesan bahwa *keyword* pencarian tidak boleh kosong.	Valid
15.	Kasus uji melihat daftar barang	Semua daftar barang dapat ditampilkan.	Valid
16.	Kasus uji melihat data barang	Data barang dapat ditampilkan.	Valid
17.	Kasus uji menambah barang	Barang berhasil ditambahkan.	Valid
18.	Kasus uji menambah barang dimana masukkan kode, nama atau harga kosong	Sistem kembali ke *form* dan menampilkan pesan bahwa masukkan kode, nama, atau harga kosong.	Valid
19.	Kasus uji menambah barang dimana kode tersebut telah ada di dalam sistem	Sistem menampilkan pesan kesalahan bahwa kode tersebut telah ada di dalam sistem.	Valid
20	Kasus uji menambah stok barang	Stok barang berhasil ditambahkan.	Valid
21.	Kasus uji mengubah data barang	Data barang dapat diubah dan disimpan.	Valid
22.	Kasus uji mengubah data barang dimana masukkan kode, nama atau harga kosong	Sistem kembali ke *form* dan menampilkan pesan bahwa masukkan kode, nama, atau harga kosong.	Valid
23.	Kasus uji mengubah data barang dimana kode tersebut telah ada di dalam sistem	Sistem menampilkan pesan kesalahan bahwa kode tersebut telah ada di dalam sistem.	Valid
24.	Kasus uji menghapus barang	Data barang dapat dihapus.	Valid
25.	Kasus uji mencari barang	Suatu barang berhasil ditampilkan.	Valid
26.	Kasus uji mencari barang dimana masukkan *keyword* kosong	Sistem menampilkan pesan bahwa *keyword* pencarian tidak boleh kosong.	
27.	Kasus uji melihat daftar kategori barang	Semua daftar kategori barang dapat ditampilkan.	Valid
28.	Kasus uji melihat data kategori barang	Data kategori barang dapat ditampilkan.	Valid
29.	Kasus uji menambah kategori barang	Kategori barang berhasil ditambahkan.	Valid
30.	Kasus uji menambah kategori barang dimana masukkan kode atau nama kosong	Sistem kembali ke *form* dan menampilkan pesan bahwa masukkan kode atau nama kosong.	Valid
31.	Kasus uji menambah kategori barang dimana kode tersebut telah ada di dalam sistem	Sistem menampilkan pesan kesalahan bahwa kode tersebut telah ada di dalam sistem.	Valid
32.	Kasus uji mengubah data	Data kategori barang dapat diubah dan disimpan.	Valid

	kategori barang		
33.	Kasus uji mengubah data kategori barang dimana masukkan kode atau nama kosong	Sistem kembali ke *form* dan menampilkan pesan bahwa masukkan kode atau nama kosong.	Valid
34.	Kasus uji mengubah data kategori barang dimana kode tersebut telah ada di dalam sistem	Sistem menampilkan pesan kesalahan bahwa kode tersebut telah ada di dalam sistem.	Valid
35.	Kasus uji menghapus kategori barang	Data kategori barang dapat dihapus.	Valid
36.	Kasus uji mencari kategori barang	Suatu kategori barang berhasil ditampilkan.	Valid
37.	Kasus uji mencari kategori barang dimana masukkan *keyword* kosong	Sistem menampilkan pesan bahwa *keyword* pencarian tidak boleh kosong.	Valid
38.	Kasus uji melihat daftar subkategori barang	Semua daftar subkategori barang dapat ditampilkan.	Valid
39.	Kasus uji menambah subkategori barang	Subkategori barang berhasil ditambahkan.	Valid
40.	Kasus uji menambah subkategori barang dimana masukkan kode atau nama kosong	Sistem kembali ke *form* dan menampilkan pesan bahwa masukkan kode atau nama kosong.	Valid
41.	Kasus uji menambah subkategori barang dimana kode tersebut telah ada di dalam sistem	Sistem menampilkan pesan kesalahan bahwa kode tersebut telah ada di dalam sistem.	Valid
42.	Kasus uji mengubah data subkategori barang	Data subkategori barang dapat diubah dan disimpan.	Valid
43.	Kasus uji mengubah data subkategori barang dimana masukkan kode atau nama kosong	Sistem kembali ke *form* dan menampilkan pesan bahwa masukkan kode atau nama kosong.	Valid
44.	Kasus uji mengubah data subkategori barang dimana kode tersebut telah ada di dalam sistem	Sistem menampilkan pesan kesalahan bahwa kode tersebut telah ada di dalam sistem.	Valid
45.	Kasus uji menghapus subkategori barang	Data subkategori barang dapat dihapus.	Valid
46.	Kasus uji melihat daftar suplier	Semua daftar suplier dapat ditampilkan.	Valid
47.	Kasus uji melihat data suplier	Data suplier dapat ditampilkan.	Valid
48.	Kasus uji menambah suplier	Suplier berhasil ditambahkan.	Valid
49.	Kasus uji menambah suplier barang dimana masukkan kode, nama, alamat, atau no.telp kosong	Sistem kembali ke *form* dan menampilkan pesan bahwa masukkan kode, nama, alamat, atau no.telp kosong.	Valid
50.	Kasus uji menambah suplier dimana kode tersebut telah ada di dalam sistem	Sistem menampilkan pesan kesalahan bahwa kode tersebut telah ada di dalam sistem.	Valid
51.	Kasus uji mengubah datasuplier	Data suplier dapat diubah dan disimpan.	Valid
52.	Kasus uji mengubah data suplier dimana masukkan kode, nama, alamat, atau no.telp kosong	Sistem kembali ke *form* dan menampilkan pesan bahwa masukkan kode, nama, alamat, atau no.telp kosong.	Valid
53.	Kasus uji mengubah data suplier dimana kode tersebut telah ada di dalam sistem	Sistem menampilkan pesan kesalahan bahwa kode tersebut telah ada di dalam sistem.	Valid
54.	Kasus uji menghapus suplier	Data suplier dapat dihapus.	Valid

55.	Kasus uji mencari suplier	Suatu suplier berhasil ditampilkan.	Valid
56.	Kasus uji mencari suplier dimana masukkan *keyword* kosong	Sistem menampilkan pesan bahwa *keyword* pencarian tidak boleh kosong.	Valid
57.	Kasus uji melihat daftar konsumen	Semua daftar konsumen dapat ditampilkan.	Valid
58.	Kasus uji menghapus konsumen	Data konsumen dapat dihapus.	Valid
59.	Kasus uji mencari konsumen	Daftar konsumen dengan suatu status validasi berhasil ditampilkan.	Valid
60.	Kasus uji mencari konsumen dimana masukkan *keyword* kosong	Sistem menampilkan pesan bahwa *keyword* pencarian tidak boleh kosong.	Valid
61.	Kasus uji melihat daftar validasi konsumen	Semua daftar validasi konsumen dapat ditampilkan.	Valid
62.	Kasus uji mengubah data konsumen	Data konsumen dapat diubah dan disimpan.	Valid
63.	Kasus uji memesan barang	Data barang yang dipesan disimpan pada keranjang belanja.	Valid
64.	Kasus uji melihat keranjang belanja	Semua daftar barang yang dipesan dapat ditampilkan dan dapat diubah.	Valid
65.	Kasus uji mengubah data keranjang belanja	Data keranjang belanja dapat diubah dan disimpan.	Valid
66.	Kasus uji menghapus keranjang belanja	Data konsumen dapat dihapus.	Valid
67.	Kasus uji melihat detail pesanan	Semua daftar barang yang dipesan dapat ditampilkan.	Valid
68.	Kasus uji check out	Data check out dapat ditampilkan dan disimpan.	Valid
69.	Kasus uji melihat daftar pesanan	Semua daftar pesanan dapat ditampilkan.	Valid
70.	Kasus uji mencetak nota	Nota dapat dicetak.	Valid

Sumber: [Pengujian]

Dari hasil pengujian validasi pada Tabel 6.75 dapat diketahui bahwa sistem yang dibangun valid, yang mana sistem dapat berjalan dengan baik dan memberikan hasil yang didapatkan *user* sesuai dengan kebutuhannya.

6.4. Pengujian Waktu Akses Query

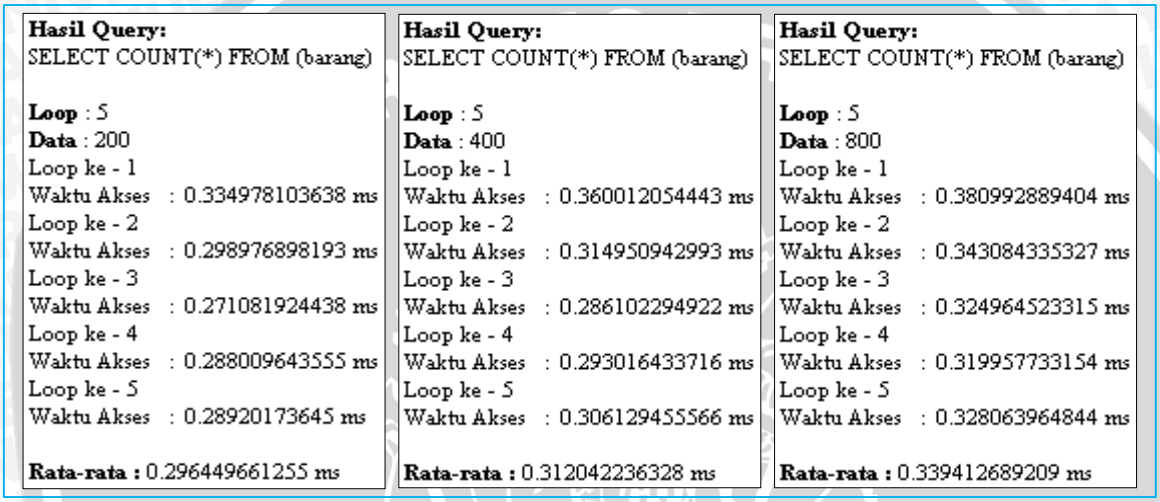
Pengujian ini dilakukan untuk mengetahui waktu yang dibutuhkan dalam melakukan *query* secara *offline* maupun secara *online* dengan menggunakan teknologi GPRS (*provider* indosatgprs), pada basis data test (tabel sama dengan basis data smarton). Prosedur yang dijalankan dengan mengakses testQuery.php. *File* testQuery.php dibuat khusus untuk melakukan pengujian waktu akses *query* dengan menggunakan fungsi `microtime()` pada PHP.

6.4.1. Proses Pengujian

1. Pengujian Waktu Akses *Query* Terhadap Tabel `barang`

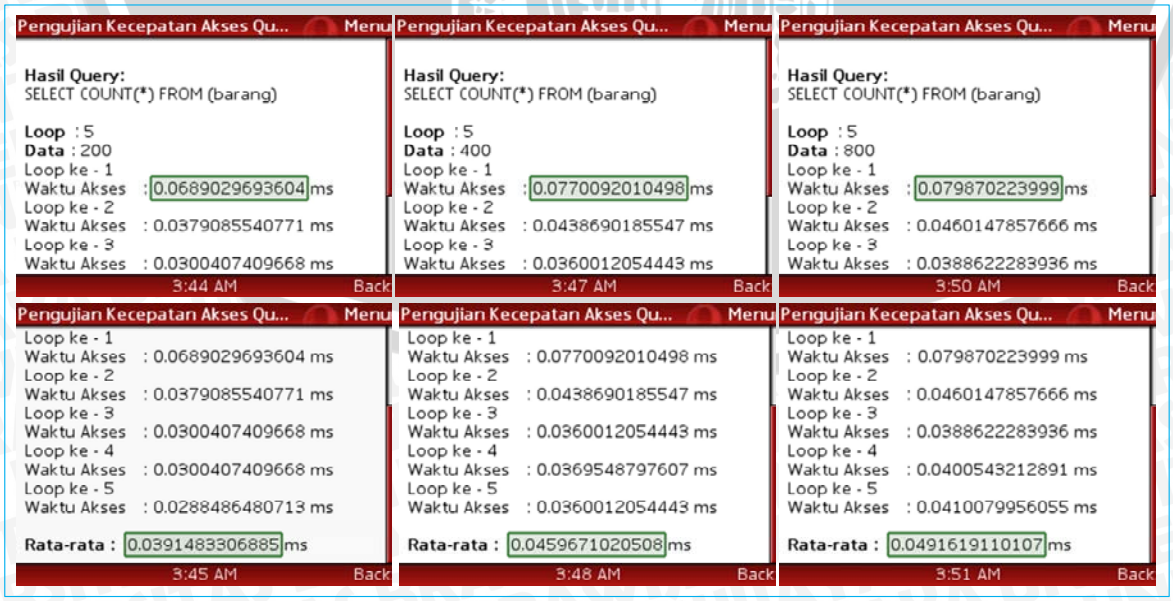
Hasil pengujian waktu akses *query* dalam detik terhadap tabel barang yang mempunyai 9 *field* (idbarang, suplier_idsuplier, kategori_idkategori, subkategori_idsubkategori, kode_barang, nama_barang, gambar, harga, deskripsi_barang, stok) dengan jumlah data *entry* sebanyak 200, 400, dan 800 *record* dengan konfigurasi *query* `SELECT COUNT(*) FROM (barang)` selama 5 kali perulangan.

Pengujian waktu akses *query* tabel barang secara *offline* melalui *localhost* ditunjukkan pada Gambar 6.6.



Gambar 6. 6 Hasil pengujian waktu akses *query* terhadap tabel barang secara *offline*
Sumber: [Pengujian]

Pengujian waktu akses *query* tabel barang secara *online* ditunjukkan pada Gambar 6.7.

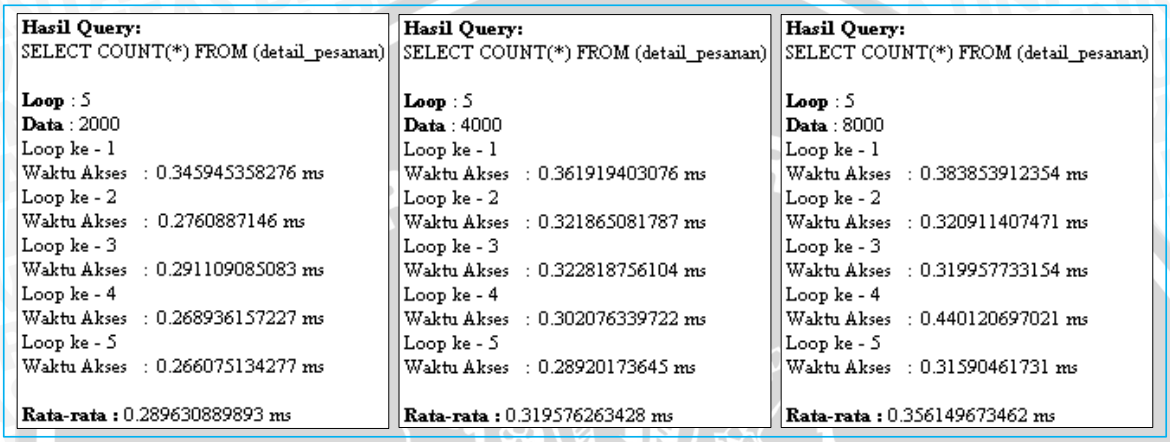


Gambar 6. 7 Hasil pengujian waktu akses *query* terhadap tabel barang secara *online*
Sumber: [Pengujian]

2. Pengujian Waktu Akses *Query* Terhadap Tabel `detail_pesanan`

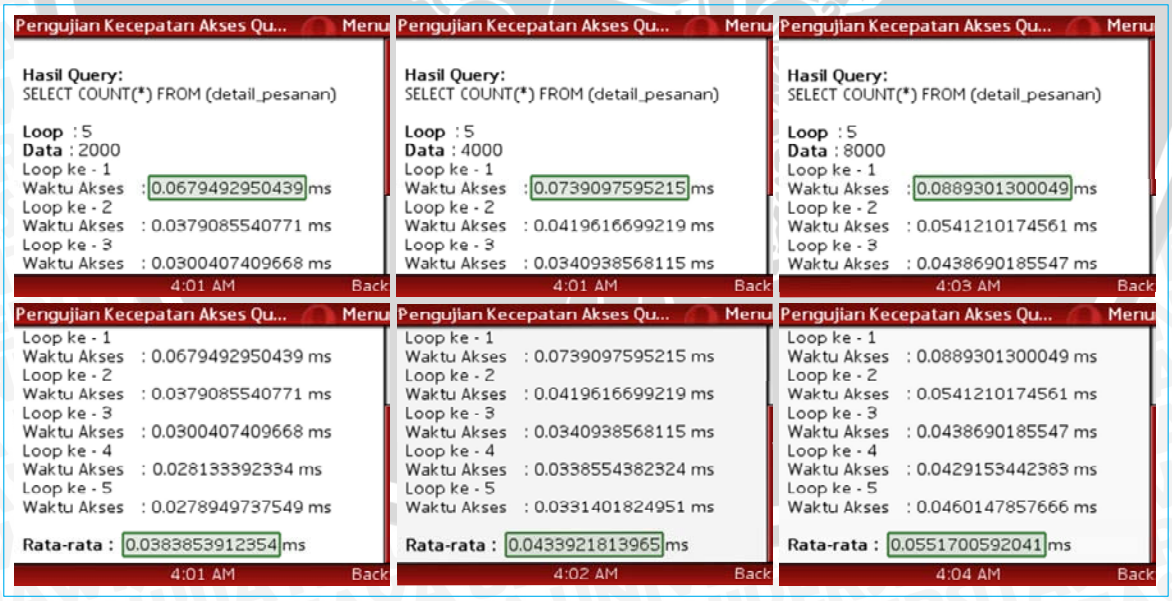
Hasil pengujian waktu akses *query* dalam detik terhadap tabel detail_pesanan yang mempunyai 4 *field* (iddetail_pesanan, barang_idbarang, transaksi_idtransaksi, jumlah_barang) dengan jumlah data *entry* sebanyak 2000, 4000, dan 8000 dengan konfigurasi *query* `SELECT COUNT(*) FROM (detail_pesanan)` selama 5 kali perulangan.

Pengujian waktu akses *query* tabel detail_pesanan secara *offline* melalui *localhost* ditunjukkan pada Gambar 6.8.



Gambar 6. 8 Hasil pengujian waktu akses *query* terhadap tabel detail_pesanan secara *offline*
Sumber: [Pengujian]

Pengujian waktu akses *query* tabel detail_pesanan secara *online* ditunjukkan pada Gambar 6.9.

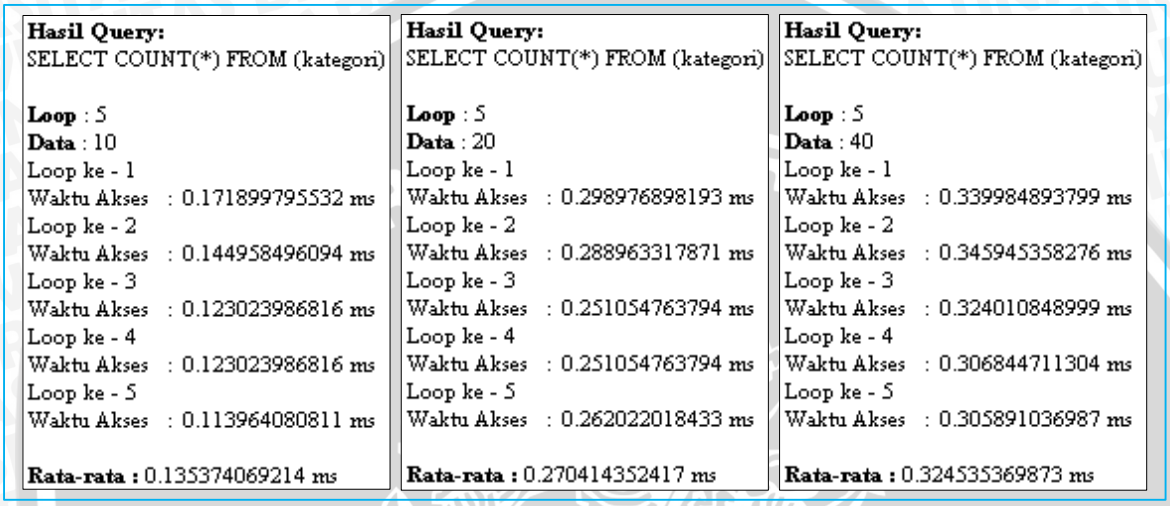


Gambar 6. 9 Hasil pengujian waktu akses *query* terhadap tabel detail_pesanan *online*
Sumber: [Pengujian]

3. Pengujian Waktu Akses *Query* Terhadap Tabel `kategori`

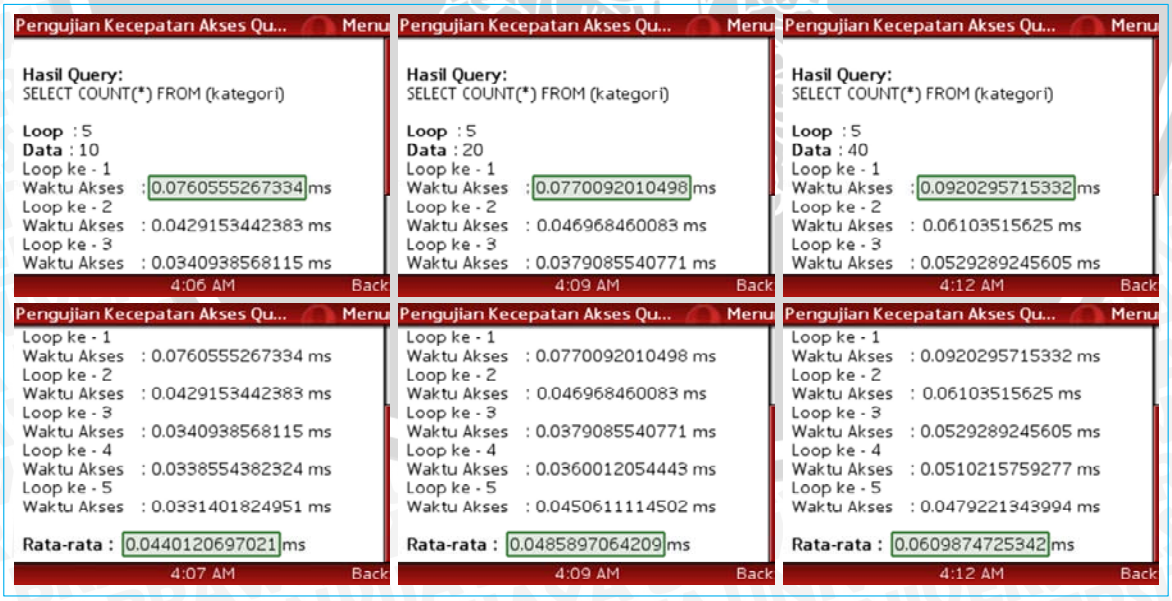
Hasil pengujian waktu akses *query* dalam detik terhadap tabel kategori yang mempunyai 4 *field* (idkategori, kode_kategori, nama_kategori, deskripsi_kategori) dengan jumlah data *entry* sebanyak 10, 20, dan 40 dengan konfigurasi *query* `SELECT COUNT(*) FROM (kategori)` selama 5 kali perulangan.

Pengujian waktu akses *query* tabel kategori secara *offline* melalui *localhost* ditunjukkan pada Gambar 6.10.



Gambar 6. 10 Hasil pengujian waktu akses *query* terhadap tabel kategori secara *offline*
Sumber: [Pengujian]

Pengujian waktu akses *query* tabel kategori secara *online* ditunjukkan pada Gambar 6.11.

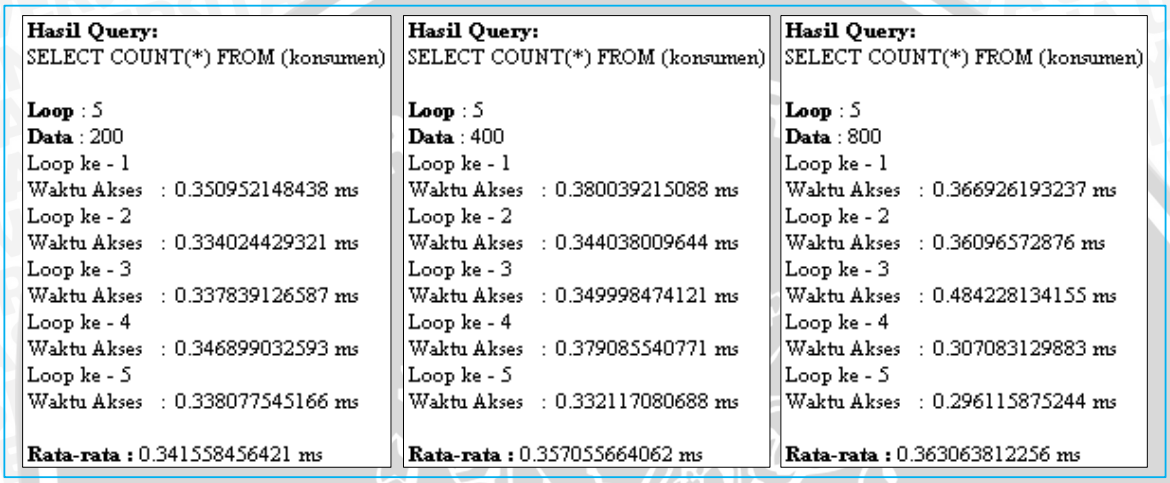


Gambar 6. 11 Hasil pengujian waktu akses *query* terhadap tabel kategori secara *online*
Sumber: [Pengujian]

4. Pengujian Waktu Akses *Query* Terhadap Tabel konsumen

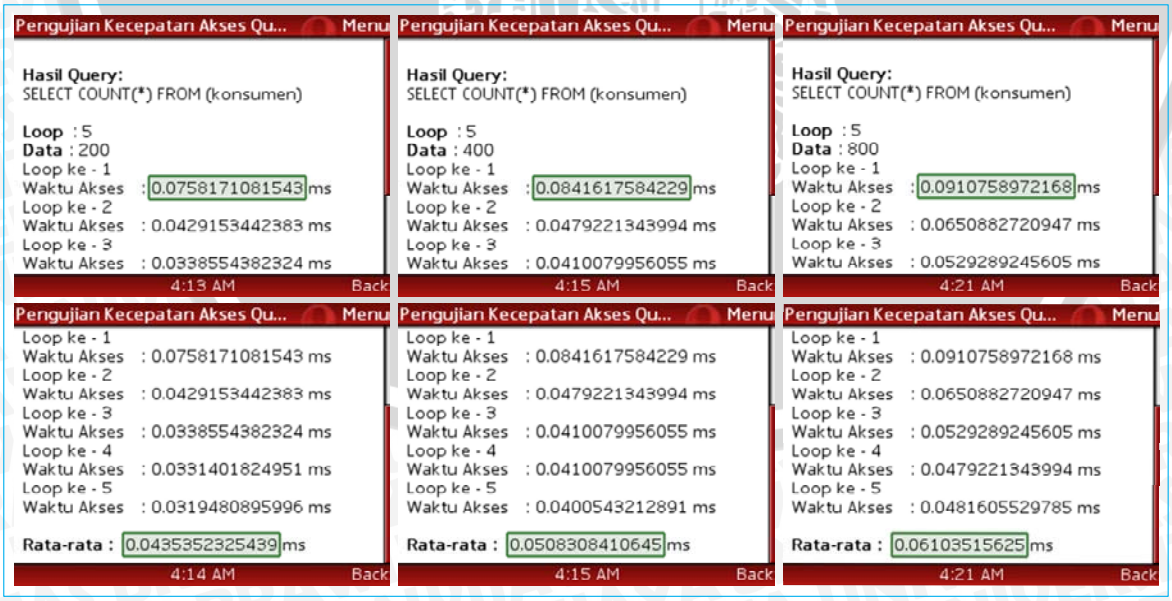
Hasil pengujian waktu akses *query* dalam detik terhadap tabel konsumen yang mempunyai 8 *field* (idkonsumen, nama_konsumen, username, password_konsumen, alamat_konsumen, hp_konsumen, email_konsumen, status_validasi) dengan jumlah data *entry* sebanyak 200, 400, dan 800 dengan konfigurasi *query* `SELECT COUNT(*) FROM (konsumen)` selama 5 kali perulangan.

Pengujian waktu akses *query* tabel konsumen secara *offline* melalui *localhost* ditunjukkan pada Gambar 6.12.



Gambar 6. 12 Hasil pengujian waktu akses *query* terhadap tabel konsumen secara *offline*
Sumber: [Pengujian]

Pengujian waktu akses *query* tabel konsumen secara *online* ditunjukkan pada Gambar 6.13.

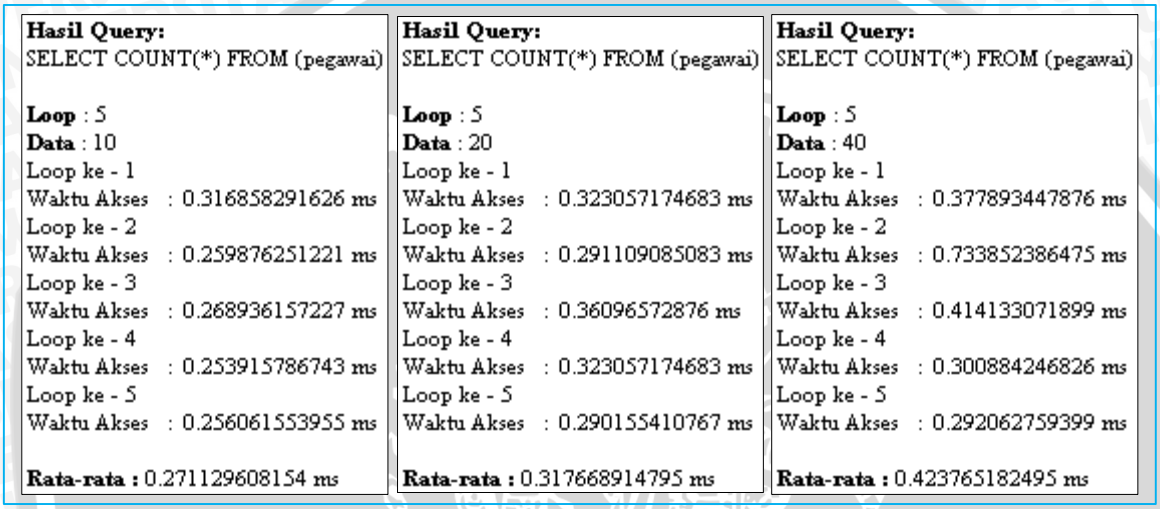


Gambar 6. 13 Hasil pengujian waktu akses *query* terhadap tabel konsumen secara *online*
Sumber: [Pengujian]

5. Pengujian Waktu Akses *Query* Terhadap Tabel `pegawai`

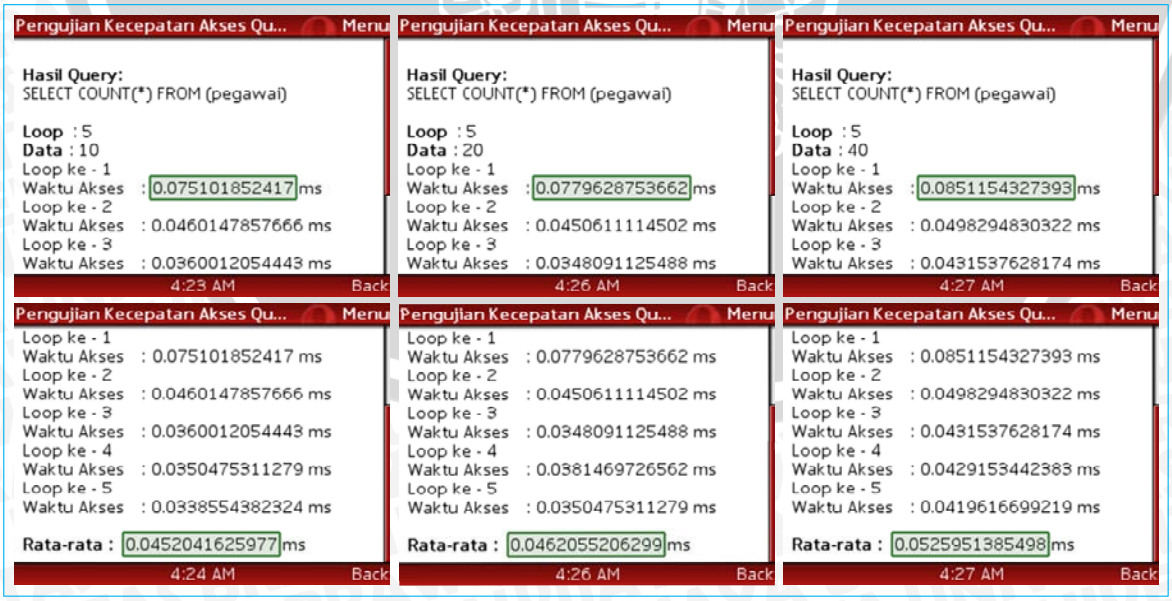
Hasil pengujian waktu akses *query* dalam detik terhadap tabel pegawai yang mempunyai 12 *field* (idpegawai, kode_pegawai, nama_pegawai, username, password_pegawai, sex_pegawai, tempat_lahir, tgl_lahir, alamat_pegawai, telp_pegawai, email_pegawai, tugas) dengan jumlah data *entry* sebanyak 10, 20, dan 40 dengan konfigurasi *query* `SELECT COUNT(*) FROM (pegawai)` selama 5 kali perulangan.

Pengujian waktu akses *query* tabel pegawai secara *offline* melalui *localhost* ditunjukkan pada Gambar 6.14.



Gambar 6. 14 Hasil pengujian waktu akses *query* terhadap tabel pegawai secara *offline*
Sumber: [Pengujian]

Pengujian waktu akses *query* tabel pegawai secara *online* ditunjukkan pada Gambar 6.15.

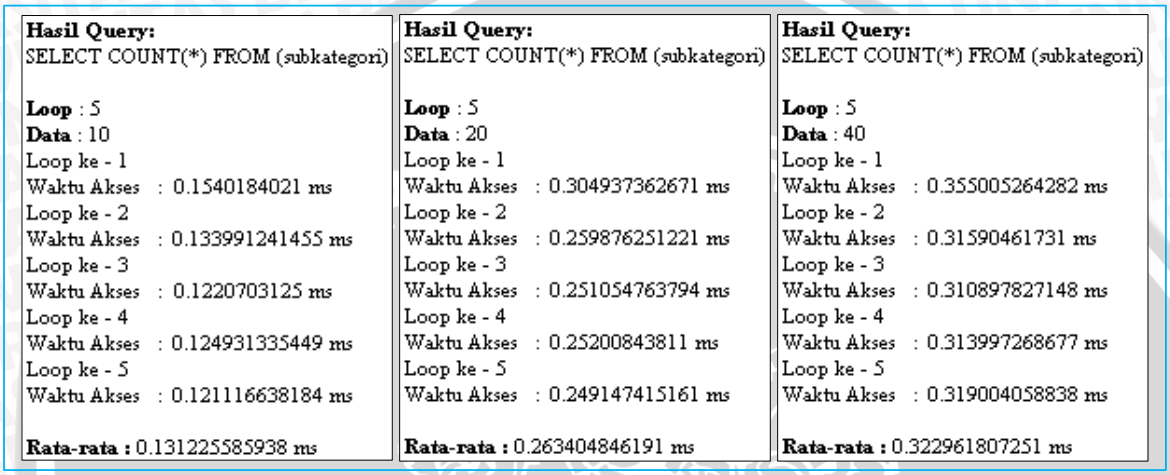


Gambar 6. 15 Hasil pengujian waktu akses *query* terhadap tabel pegawai secara *online*
Sumber: [Pengujian]

6. Pengujian Waktu Akses *Query* Terhadap Tabel `subkategori`

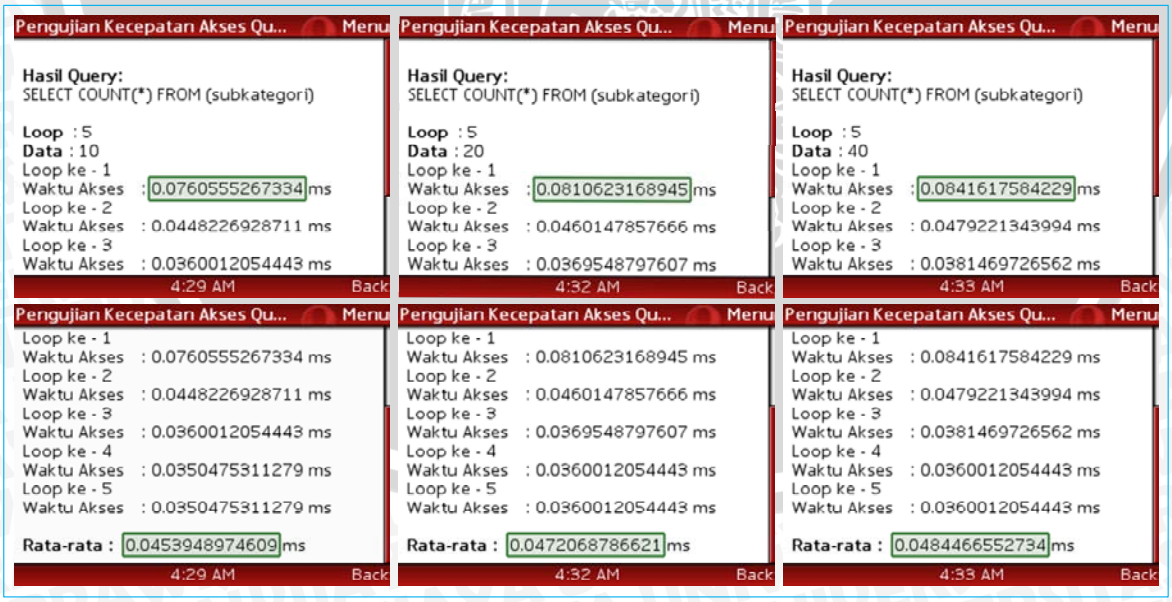
Hasil pengujian waktu akses *query* dalam detik terhadap tabel subkategori yang mempunyai 4 *field* (idsubkategori, kategori_idkategori, kode_subkategori, nama_subkategori) dengan jumlah data *entry* sebanyak 10, 20, dan 40 dengan konfigurasi *query* `SELECT COUNT(*) FROM (subkategori)` selama 5 kali perulangan.

Pengujian waktu akses *query* tabel subkategori secara *offline* melalui *localhost* ditunjukkan pada Gambar 6.16.



Gambar 6. 16 Hasil pengujian waktu akses *query* terhadap tabel subkategori secara *offline*
Sumber: [Pengujian]

Pengujian waktu akses *query* tabel subkategori secara *online* ditunjukkan pada Gambar 6.17.

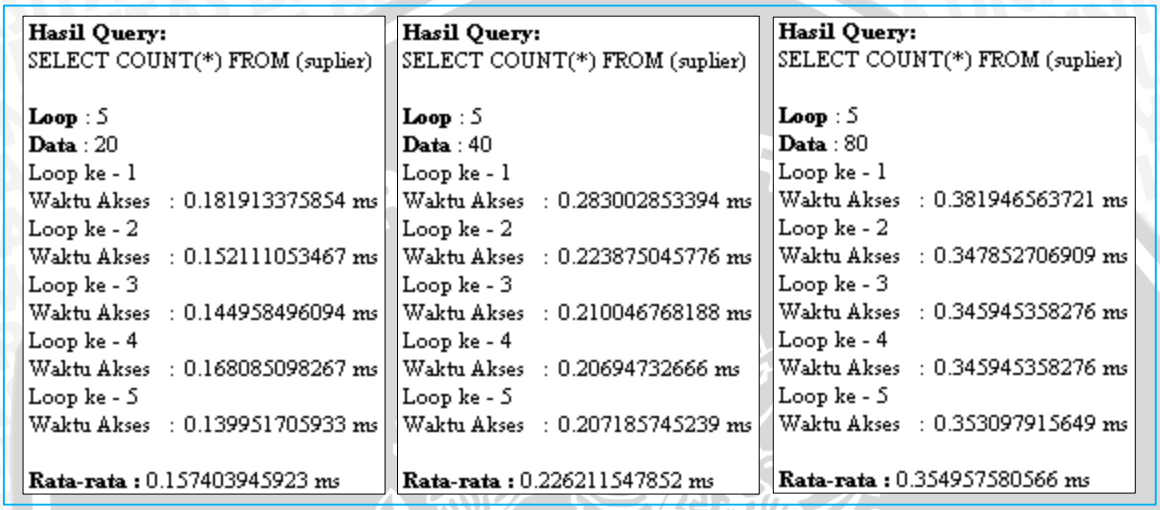


Gambar 6. 17 Hasil pengujian waktu akses *query* terhadap tabel subkategori secara *online*
Sumber: [Pengujian]

7. Pengujian Waktu Akses *Query* Terhadap Tabel `suplier`

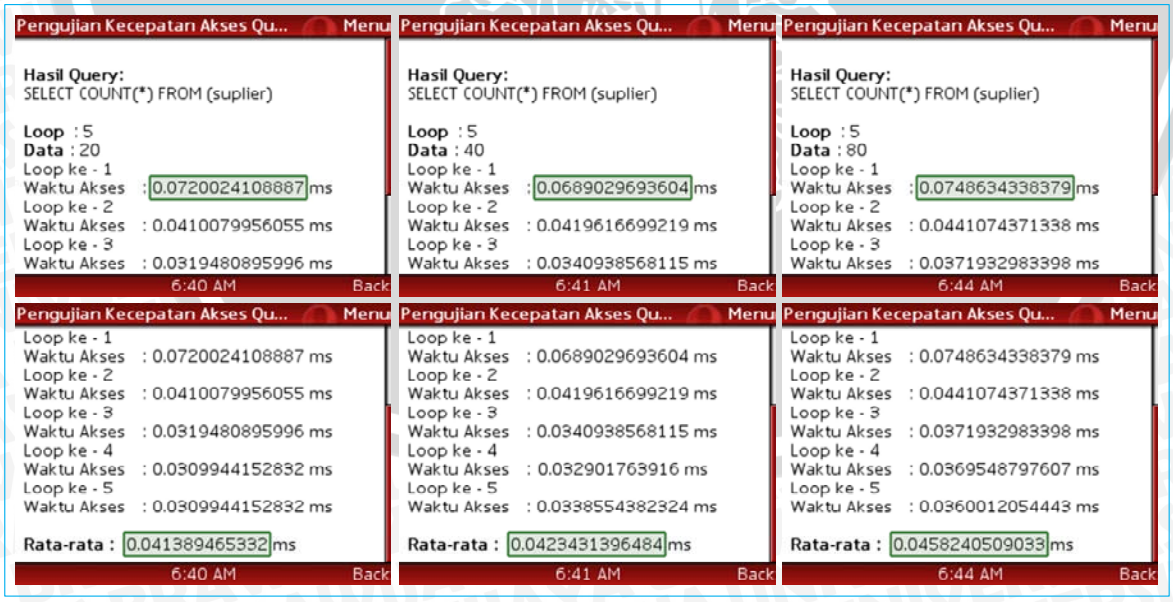
Hasil pengujian waktu akses *query* dalam detik terhadap tabel suplier yang mempunyai 5 *field* (idsuplier, kode_suplier, nama_suplier, alamat_suplier, telp_suplier) dengan jumlah data *entry* sebanyak 20, 40, dan 80 dengan konfigurasi *query* `SELECT COUNT(*) FROM (suplier)` selama 5 kali perulangan.

Pengujian waktu akses *query* tabel suplier secara *offline* melalui *localhost* ditunjukkan pada Gambar 6.18.



Gambar 6. 18 Hasil pengujian waktu akses *query* terhadap tabel suplier secara *offline*
Sumber: [Pengujian]

Pengujian waktu akses *query* tabel suplier secara *online* ditunjukkan pada Gambar 6.19.

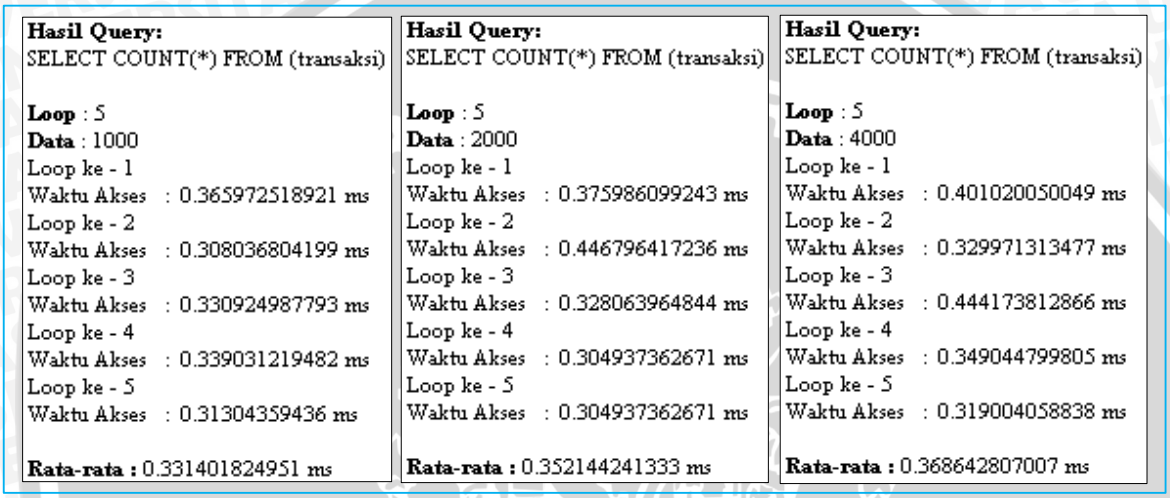


Gambar 6. 19 Hasil pengujian waktu akses *query* terhadap tabel suplier secara *online*
Sumber: [Pengujian]

8. Pengujian Waktu Akses *Query* Terhadap Tabel `transaksi`

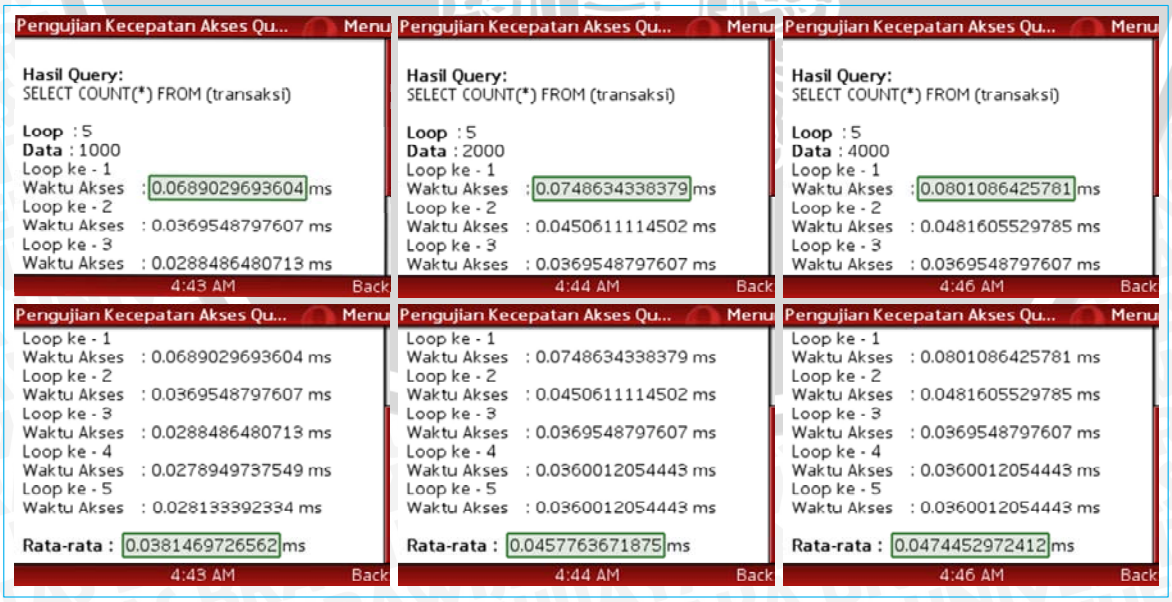
Hasil pengujian waktu akses *query* dalam detik terhadap tabel transaksi yang mempunyai 6 *field* (idtransaksi, pegawai_idpegawai, konsumen_idkonsumen, kode_transaksi, waktu_pesan, status_kirim) dengan jumlah data *entry* sebanyak 1000, 2000, dan 4000 dengan konfigurasi *query* `SELECT COUNT(*) FROM (transaksi)` selama 5 kali perulangan.

Pengujian waktu akses *query* tabel transaksi secara *offline* melalui *localhost* ditunjukkan pada Gambar 6.20.



Gambar 6. 20 Hasil pengujian waktu akses *query* terhadap tabel transaksi secara *offline*
Sumber: [Pengujian]

Pengujian waktu akses *query* tabel transaksi secara *online* ditunjukkan pada Gambar 6.21.



Gambar 6. 21 Hasil pengujian waktu akses *query* terhadap tabel transaksi secara *online*
Sumber: [Pengujian]

6.4.2. Hasil Pengujian Dan Analisis

Database server di-*instal* di *local server* sekaligus digunakan sebagai *client*. Basis data *test* pada *local server* dapat menangani permintaan *query* dari *client* secara *offline*. *Database* juga di-*instal* di *web server* sehingga dapat menangani permintaan *query* dari *client* secara *online*.

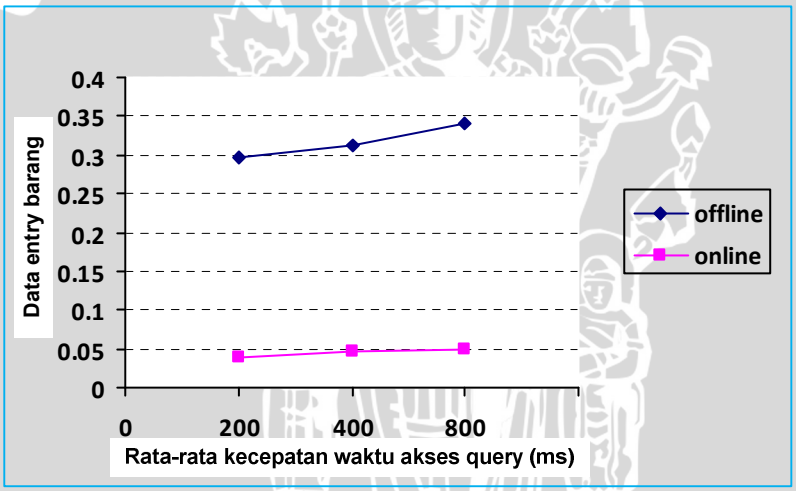
Tabel 6.76 menunjukkan rata-rata waktu akses *query* dalam *milisecond* dengan 200, 400, dan 800 untuk tabel barang.

Tabel 6. 76 Tabel rata-rata waktu akses *query* tabel barang

Akses	Rata-rata waktu akses dengan 200 data (ms)	Rata-rata waktu akses dengan 400 data (ms)	Rata-rata waktu akses dengan 800 data (ms)
offline	0.296449661255	0.312042236328	0.339412689209
online	0.0391483306885	0.0459671020508	0.0491619110107

Sumber: [Pengujian]

. Gambar 6.22 menunjukkan grafik hubungan antara data *entry* dan nilai rata-rata kecepatan waktu akses *query* tabel barang yang diakses secara *offline* dan *online*.



Gambar 6. 22 Grafik hubungan antara data *entry* dan rata-rata kecepatan waktu akses *query* tabel barang yang diakses secara *offline* dan *online*
Sumber: [Pengujian]

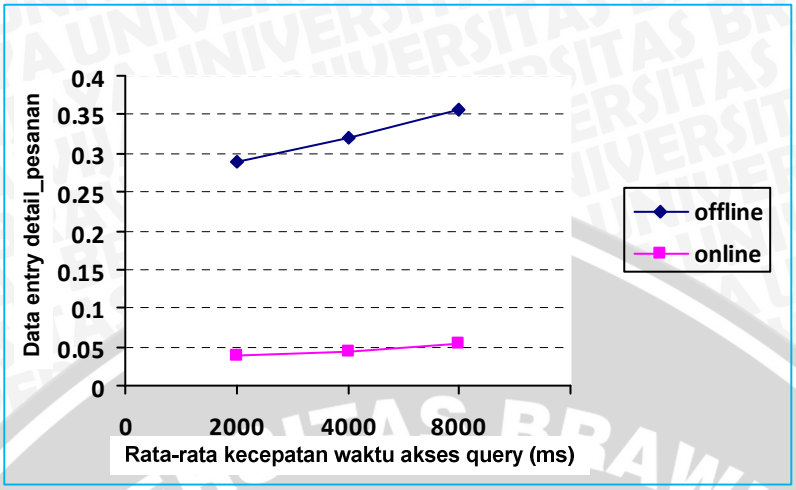
Tabel 6.77 menunjukkan rata-rata waktu akses *query* dalam *milisecond* dengan 2000, 4000, dan 8000 untuk tabel detail_pesanan.

Tabel 6. 77 Tabel rata-rata waktu akses *query* tabel detail_pesanan

Akses	Rata-rata waktu akses dengan 2000 data (ms)	Rata-rata waktu akses dengan 4000 data (ms)	Rata-rata waktu akses dengan 8000 data (ms)
offline	0.289630889893	0.319576263428	0.356149673462
online	0.0383853912354	0.0433921813965	0.0551700592041

Sumber: [Pengujian]

Gambar 6.23 menunjukkan grafik hubungan antara data *entry* dan nilai rata-rata kecepatan waktu akses *query* tabel detail_pesanan yang diakses secara *offline* dan *online*.



Gambar 6. 23 Grafik hubungan antara data *entry* dan rata-rata kecepatan waktu akses *query* tabel detail_pesanan yang diakses secara *offline* dan *online*
Sumber: [Pengujian]

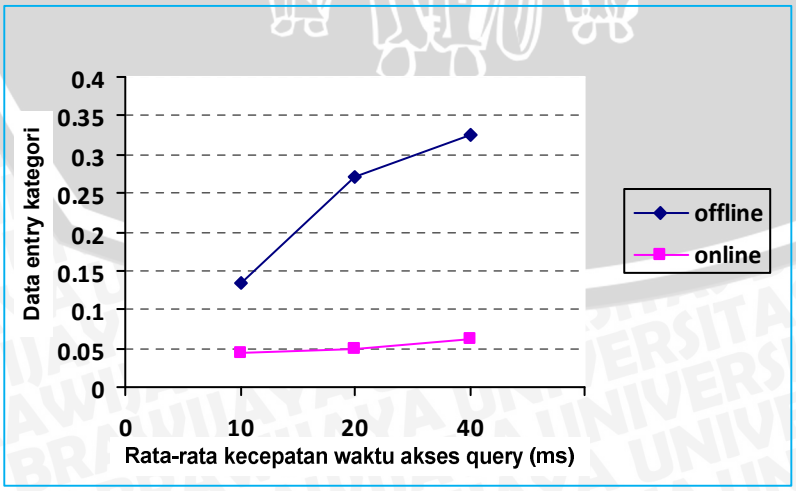
Tabel 6.78 menunjukkan rata-rata waktu akses *query* dalam *milisecond* dengan 10, 20, dan 40 untuk tabel kategori.

Tabel 6. 78 Tabel rata-rata waktu akses *query* tabel kategori

Akses	Rata-rata waktu akses dengan 10 data (ms)	Rata-rata waktu akses dengan 20 data (ms)	Rata-rata waktu akses dengan 40 data (ms)
offline	0.135374069214	0.270414352417	0.324535369873
online	0.0440120697021	0.0485897064209	0.0609874725342

Sumber: [Pengujian]

Gambar 6.24 menunjukkan grafik hubungan antara data *entry* dan nilai rata-rata kecepatan waktu akses *query* tabel kategori yang diakses secara *offline* dan *online*.



Gambar 6. 24 Grafik hubungan antara data *entry* dan rata-rata kecepatan waktu akses *query* tabel kategori yang diakses secara *offline* dan *online*
Sumber: [Pengujian]

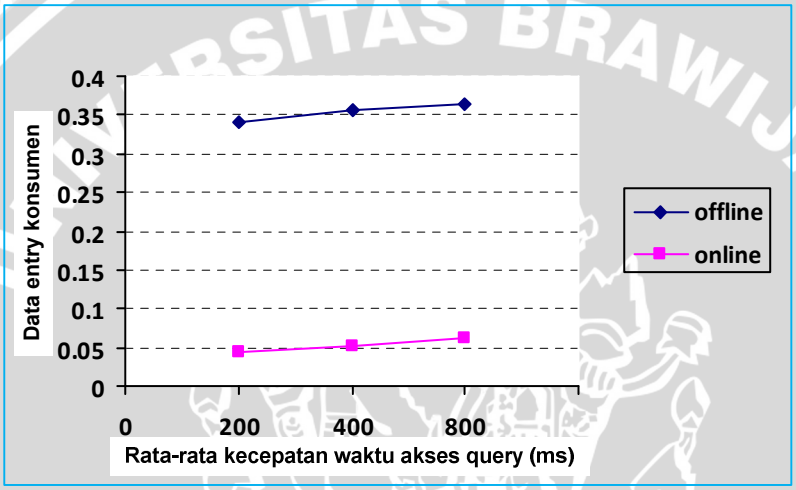
Tabel 6.79 menunjukkan rata-rata waktu akses *query* dalam *milisecond* dengan 200, 400, dan 800 untuk tabel konsumen.

Tabel 6. 79 Tabel rata-rata waktu akses *query* tabel konsumen

Akses	Rata-rata waktu akses dengan 200 data (ms)	Rata-rata waktu akses dengan 400 data (ms)	Rata-rata waktu akses dengan 800 data (ms)
offline	0.341558456421	0.357055664062	0.363063812256
online	0.0435352325439	0.0508308410645	0.06103515625

Sumber: [Pengujian]

Gambar 6.25 menunjukkan grafik hubungan antara data *entry* dan nilai rata-rata kecepatan waktu akses *query* tabel konsumen yang diakses secara *offline* dan *online*.



Gambar 6. 25 Grafik hubungan antara data *entry* dan rata-rata kecepatan waktu akses *query* tabel konsumen yang diakses secara *offline* dan *online*
Sumber: [Pengujian]

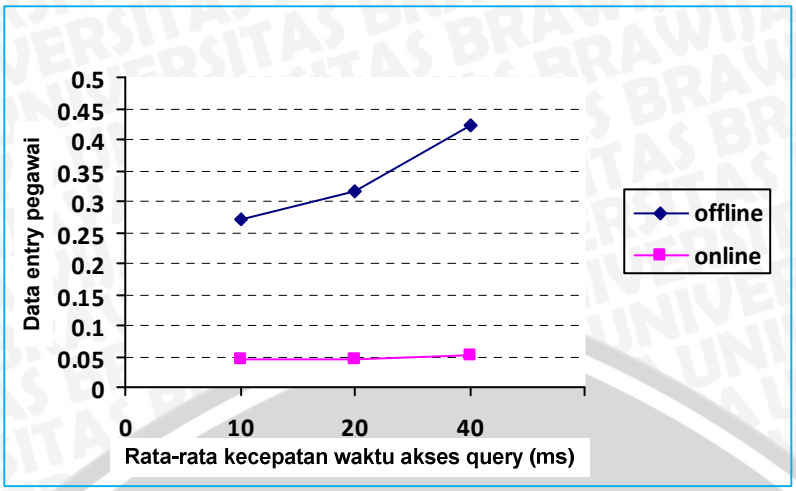
Tabel 6.80 menunjukkan rata-rata waktu akses *query* dalam *milisecond* dengan 10, 20, dan 40 untuk tabel pegawai.

Tabel 6. 80 Tabel rata-rata waktu akses *query* tabel pegawai

Akses	Rata-rata waktu akses dengan 10 data (ms)	Rata-rata waktu akses dengan 20 data (ms)	Rata-rata waktu akses dengan 40 data (ms)
offline	0.271129608154	0.317668914795	0.423765182495
online	0.0452041625977	0.0462055206299	0.0525951385498

Sumber: [Pengujian]

Gambar 6.26 menunjukkan grafik hubungan antara data *entry* dan nilai rata-rata kecepatan waktu akses *query* tabel pegawai yang diakses secara *offline* dan *online*.



Gambar 6. 26 Grafik hubungan antara data *entry* dan rata-rata kecepatan waktu akses *query* tabel pegawai yang diakses secara *offline* dan *online*
Sumber: [Pengujian]

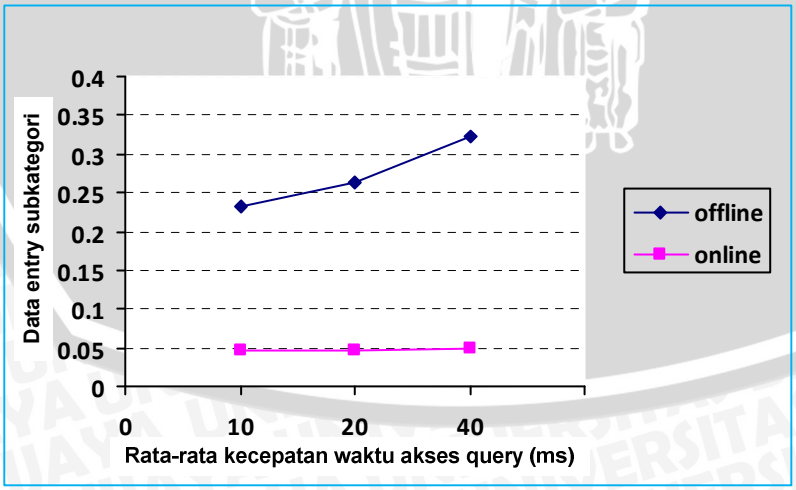
Tabel 6.81 menunjukkan rata-rata waktu akses *query* dalam *milisecond* dengan 10, 20, dan 40 untuk tabel subkategori.

Tabel 6. 81 Tabel rata-rata waktu akses *query* tabel subkategori

Akses	Rata-rata waktu akses dengan 10 data (ms)	Rata-rata waktu akses dengan 20 data (ms)	Rata-rata waktu akses dengan 40 data (ms)
offline	0.231225585938	0.263404846191	0.322961807251
online	0.0453948974609	0.0472068786621	0.0484466552734

Sumber: [Pengujian]

Gambar 6.27 menunjukkan grafik hubungan antara data *entry* dan nilai rata-rata kecepatan waktu akses *query* tabel subkategori yang diakses secara *offline* dan *online*.



Gambar 6. 27 Grafik hubungan antara data *entry* dan rata-rata kecepatan waktu akses *query* tabel subkategori yang diakses secara *offline* dan *online*
Sumber: [Pengujian]

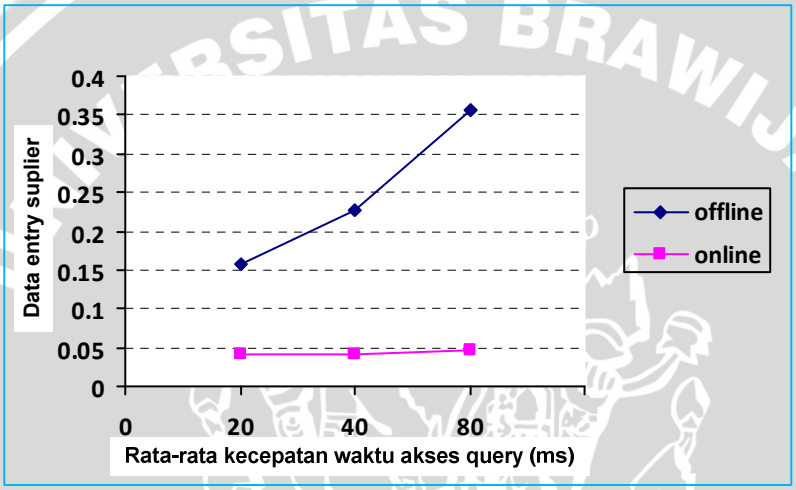
Tabel 6.82 menunjukkan rata-rata waktu akses *query* dalam *milisecond* dengan 20, 40, dan 80 untuk tabel suplier.

Tabel 6. 82 Tabel rata-rata waktu akses *query* tabel suplier

Akses	Rata-rata waktu akses dengan 20 data (ms)	Rata-rata waktu akses dengan 40 data (ms)	Rata-rata waktu akses dengan 80 data (ms)
offline	0.157403945923	0.226211547852	0.354957580566
online	0.041389465332	0.0423431396484	0.0458240509033

Sumber: [Pengujian]

Gambar 6.28 menunjukkan grafik hubungan antara data *entry* dan nilai rata-rata kecepatan waktu akses *query* tabel suplier yang diakses secara *offline* dan *online*.



Gambar 6. 28 Grafik hubungan antara data *entry* dan rata-rata kecepatan waktu akses *query* tabel suplier yang diakses secara *offline* dan *online*
Sumber: [Pengujian]

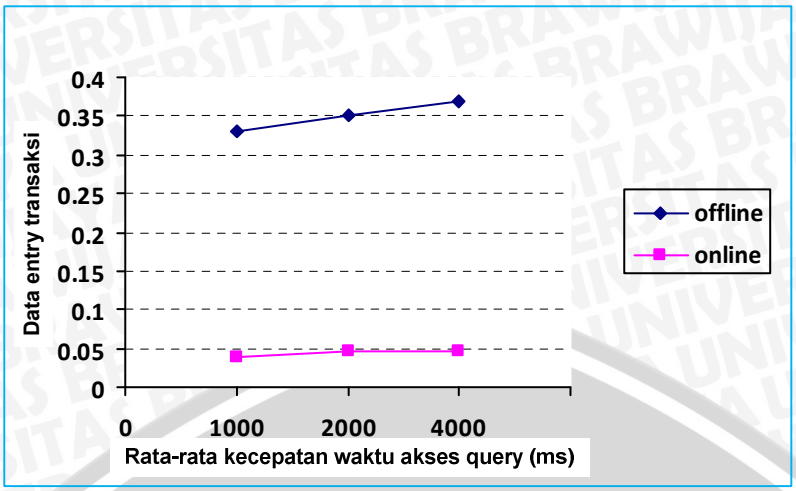
Tabel 6.83 menunjukkan rata-rata waktu akses *query* dalam *milisecond* dengan 1000, 2000, dan 4000 untuk tabel transaksi.

Tabel 6. 83 Tabel rata-rata waktu akses *query* tabel transaksi

Akses	Rata-rata waktu akses dengan 1000 data (ms)	Rata-rata waktu akses dengan 2000 data (ms)	Rata-rata waktu akses dengan 4000 data (ms)
offline	0.331401824951	0.352144241333	0.368642807007
online	0.0381469726562	0.0457763671875	0.0474452972412

Sumber: [Pengujian]

Gambar 6.29 menunjukkan grafik hubungan antara data *entry* dan nilai rata-rata kecepatan waktu akses *query* tabel transaksi yang diakses secara *offline* dan *online*.



Gambar 6. 29 Grafik hubungan antara data *entry* dan rata-rata kecepatan waktu akses *query* tabel transaksi yang diakses secara *offline* dan *online*
Sumber: [Pengujian]

Dari grafik-grafik hubungan antara data *entry* dan nilai rata-rata kecepatan waktu akses *query* dari tabel-tabel dalam basis data, menunjukkan perbedaan kecepatan waktu akses *query* yang dilakukan secara *offline* dan *online*, dimana kecepatan pada saat diakses secara *online* lebih cepat dari pada kecepatan saat diakses secara *offline*. Perbedaan pengujian kecepatan waktu akses *query* yang dilakukan secara *offline* dan *online* ini terjadi karena faktor *web server* yang digunakan dan faktor jaringan.

BAB VII
PENUTUP

7.1. Kesimpulan

Berdasarkan hasil perancangan, implementasi, dan pengujian yang dilakukan, maka diambil kesimpulan sebagai berikut :

1. Sistem *mobile online supermarket* ini *valid* sesuai dengan kebutuhan *user*, dapat bekerja dengan baik karena hasil yang didapatkan sesuai dengan hasil yang diharapkan.
2. Pengujian waktu akses *query* pada basis data *smarton* menunjukkan perbedaan kecepatan waktu akses *query* yang dilakukan secara *offline* dan *online*, dimana kecepatan pada saat diakses secara *online* lebih cepat dari pada kecepatan saat diakses secara *offline*. Perbedaan pengujian kecepatan waktu akses *query* yang dilakukan secara *offline* dan *online* ini terjadi karena faktor *web server* yang digunakan dan faktor jaringan.

7.2. Saran

Saran yang dapat diberikan untuk pengembangan sistem *mobile online supermarket* dengan teknologi XHTML MP dan WCSS sebagai berikut :

1. Menu dalam *mobile online supermarket* ini dibuat lebih variasi, misalnya dengan ditambahkan menu kotak saran dan menu info tentang *smart-on*.
2. Sistem *mobile online supermarket* dapat dikembangkan lagi sehingga mempunyai kemampuan untuk menyediakan informasi yang lebih lengkap dan menarik.

DAFTAR PUSTAKA

[AN1-09] Anonim. *Bab 9 Pengujian Perangkat Lunak*. http://www.pdf-search-engine.com/metrik-teknik-untuk-perangkat-lunak-pdf.html. 2009.

[AN1-08] Anonim. *Supermarket*. http://id.wikipedia.org/wiki/Supermarket. 2008.

[AN2-08] Anonim. *WCSS Tutorial*. www.developershome.com/wap/xhtmlmp/xhtml_mp_tutorial.asp?page=introduction. 2008.

[AN3-08] Anonim. *XHTML MP Tutorial*. www.developershome.com/wap/wcss/. 2008.

[DHA-03] Dharwiyanti, Sri dan Romi Satria Wahono. *Pengantar Unified Modeling Language (UML)*. http://setia.staff.gunadarma.ac.id/Downloads/files/6039/MateriSuplemenUml.pdf. 2003.

[FER-04] Feri. *Web Portal*. http://www.total.or.id/info.php?kk=Web%20portal. 2004.

[JMD-03] JM, Danang. *Pasar Modern Terus Geser Peran Pasar Tradisional*. http://www.sinarharapan.co.id/ekonomi/promarketing/2004/0622/prom1.html. 2003.

[JUS-07] Jusak. *Kreasi Situs Mobile Internet dengan XHTML MP*. Jakarta: Prestasi Pustaka, 2007.

[KAD-02] Kadir, Abdul. *Dasar Pemrograman Web Dinamis Menggunakan PHP*. Yogyakarta: Andi. 2002.

[KUR-02] Kurniawan, Yahya. *Aplikasi Web Database dengan PHP dan MySQL*. Jakarta: PT Elex Media Komputindo, 2002.

[MUN-05] Munawar. *Pemodelan Visual dengan UML*. Yogyakarta: Graha Ilmu. 2005.

[PRA-03] Prasetyo, Didik Dwi. *Administrasi Database Server MySQL*. Jakarta: PT Elex Media Komputindo, 2003.

[SIM-06] Simarmata, Janner dan Iman Paryudi. *Basis Data*. Yogyakarta: Andi, 2006.

[SUT-07] Sutisna, Dadan. *7 Langkah Mudah Menjadi Webmaster*. Jakarta: Mediakita, 2007.

[UTD-02] Utdirartatmo, Firrar. *Mengelola Database Server MySQL di Linux dan Windows*. Yogyakarta: Andi, 2002.