PENGEMBANGAN APLIKASI
INTERNET-BASED KNOWLEDGE MARKET

SKRIPSI
KONSENTRASI TEKNIK INFORMATIKA DAN KOMPUTER

Diajukan untuk memenuhi sebagian persyaratan
untuk memperoleh gelar Sarjana Teknik

Disusun Oleh:

ANDRE WIDIARTANTO
NIM. 0310630015 - 63

KEMENTERIAN PENDIDIKAN NASIONAL
UNIVERSITAS BRAWIJAYA
FAKULTAS TEKNIK
MALANG
2010

LEMBAR PERSETUJUAN

PENGEMBANGAN APLIKASI *INTERNET-BASED KNOWLEDGE MARKET*

SKRIPSI

KONSENTRASI TEKNIK INFORMATIKA DAN KOMPUTER

Diajukan untuk memenuhi sebagian persyaratan
untuk memperoleh gelar Sarjana Teknik

Disusun Oleh:

ANDRE WIDIARTANTO
NIM. 0310630015 - 63

Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
a.n.	
Herman Tolle, ST., MT.	**Himawat Aryadita, ST., MT., MSc.**
NIP. 19740823 200012 1 001	**NIP. 19801018 200801 1 003**

LEMBAR PENGESAHAN

PENGEMBANGAN APLIKASI *INTERNET-BASED KNOWLEDGE MARKET*

SKRIPSI

KONSENTRASI TEKNIK INFORMATIKA DAN KOMPUTER

Diajukan untuk memenuhi sebagian persyaratan
untuk memperoleh gelar Sarjana Teknik

Disusun Oleh:

ANDRE WIDIARTANTO

NIM. 0310630015 - 63

Skripsi ini telah diuji dan dinyatakan lulus
pada tanggal 26 Juli 2010

Dosen Penguji 1	Dosen Penguji 2
Dr. Ir. Harry S. Dachlan, M.Sc.	Waru Djuriatno, ST., MT.
NIP. 19490309 198602 1 001	NIP. 19690725 199702 1 001

Dosen Penguji 3

R. Arief Setyawan, ST., MT
NIP.19750819 199903 1 001

Mengetahui,
Ketua Jurusan Teknik Elektro

Rudy Yuwono, ST., M.Sc
NIP. 19710615 199802 1 003

PENGANTAR

Puji dan syukur kami panjatkan ke hadirat Tuhan Yang Maha Esa, karena berkat rahmat dan karunia-Nya sajalah penyusunan skripsi dengan judul "Pengembangan Aplikasi *Internet-Based Knowledge Market*" ini dapat diselesaikan. Penulis menyadari bahwa kajian ini tidak akan mencapai titik akhir penyelesaian tanpa bantuan dari berbagai pihak, oleh karena itu penulis mengucapkan terima kasih kepada:

1. Papa, Mama, dan seluruh keluarga besar kami untuk seluruh do a dan dukungannya yang telah diberikan kepada Ananda selama studi hingga terselesaikannya skripsi ini.
2. Bapak Rudy Yuwono, ST., MSc. selaku Ketua Jurusan Teknik Elektro, Fakultas Teknik, Universitas Brawijaya.
3. Bapak M. Azis Muslim, ST., MT., Ph.D. selaku Sekretaris Jurusan Teknik Elektro, Fakultas Teknik, Universitas Brawijaya.
4. Bapak Herman Tolle ST., MT. dan Bapak Himawat Aryadita ST., MT., MSc. selaku dosen pembimbing pada penyusunan skripsi ini.
5. Bapak Dr. Ir. Harry S. Dachlan, M.Sc., Bapak Waru Djuriatno, ST., MT., dan Bapak R. Arief Setyawan, ST., MT. selaku dosen penguji skripsi ini.
6. Bapak dan Ibu dosen serta karyawan Jurusan Teknik Elektro, dan Fakultas Teknik, Universitas Brawijaya.
7. Para sahabat dan teman-teman elektro semuanya yang telah bersama-sama menjalani hari dengan segala suka dan duka. Terima kasih atas segala bantuan dan dukungan yang telah diberikan selama ini.
8. Serta semua pihak yang tak dapat disebutkan satu per satu yang telah turut membantu baik secara langsung maupun tidak langsung dalam penyelesaian skripsi ini.

Tiada yang sempurna di dunia ini, tersadar bahwa skripsi ini sangat jauh dari kesempurnaan, segala kritik dan saran yang sifatnya membangun dari pembaca tentang isi skripsi ini akan diterima dengan senang hati. Akhir kata, penuli s berharap, semoga skripsi ini dapat bermanfaat bagi semua pihak yang membutuhkan.

Malang, Juli 2010

Penulis

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

RINGKASAN

Andre Widiartanto, Jurusan Teknik Elektro, Fakultas Teknik Universitas Brawijaya, Juli 2010, *Pengembangan Aplikasi Internet-Based Knowledge Market*, Dosen Pembimbing: Herman Tolle, dan Himawat Aryadita.

Aplikasi *Internet-Based Knowledge Market* adalah suatu aplikasi forum diskusi berbasis web berbentuk tanya jawab yang memanfaatkan media internet untuk memperbanyak jumlah masukan jawaban dari pengguna. Pada penggunaan di khalayak umum aplikasi ini dinilai cukup bermanfaat untuk memberikan alternatif-alternatif solusi untuk permasalahan dalam kehidupan sehari-hari. Kelebihan inilah yang cukup menarik penulis untuk mengembangkan aplikasi serupa untuk digunakan di lingkup universitas yang memiliki potensi sumber daya yang besar yang mana selama ini belum dimanfaatkan secara optimal.

Aplikasi dikembangkan dengan menggunakan bahasa pemrograman PHP, sistem basis data MySQL, sistem *web server* Apache, dan *framework* Codeigniter. Metode pengembangan dan pengujian yang digunakan adalah metode pemrograman berorientasi-objek.

Hasil pengujian menggambarkan bahwa aplikasi ini berhasil dibuat dan semua fungsionalitas yang direncanakan telah dapat berjalan dengan baik dalam dua lingkungan yang telah ditentukan, yaitu pada *local server* (*localhost*) maupun pada *remote server*.

Kata kunci: internet, *knowledge market*, codeigniter, pemrograman berorientasi-objek

BAB I
PENDAHULUAN

1.1 Latar Belakang

Internet saat ini sudah menjadi kebutuhan sehari-hari di beberapa kelompok masyarakat Indonesia. Para remaja dan kaum muda mengambil porsi terbesar sebagai pengguna internet di Indonesia. Masyarakat dalam kelompok ini sebagian besar memiliki profesi sebagai pelajar, baik tingkat menengah maupun tingkat perguruan tinggi. Pelajar seringkali memanfaatkan media internet untuk mencari informasi terkait dengan pengerjaan tugas-tugas sekolah.

Berbeda dengan pelajar tingkat menengah, mahasiswa sebagai pelajar tingkat tinggi dituntut untuk lebih banyak melakukan pembelajaran secara mandiri. Di dalam perguruan tinggi juga terdapat komunitas civitas academica yang memiliki anggota yang demikian besar bila dibandingkan dengan sekolah menengah. Jumlah masyarakat terpelajar yang besar ini, baik mahasiswa maupun tenaga pengajarnya (dosen), menyimpan potensi pengetahuan yang dapat digali lebih mendalam, dan salah satu caranya adalah dengan memanfaatkan media internet.

Saat ini teknologi internet telah memasuki era baru yang dikenal dengan era Web 2.0. Pada era Web 1.0, orang hanya dapat mencari dan membaca informasi dari Internet, namun pada era Web 2.0 masyarakat dapat membuat konten sendiri, bahkan berdiskusi melalui internet. Analogi yang memudahkan untuk membandingkan Web 1.0 dengan Web 2.0 adalah analogi *permission* pada sistem file. Pada Web 1.0 pengguna hanya memiliki hak *read* (baca), sedangkan pada Web 2.0 pengguna memiliki hak *read and write* (baca dan tulis).

Perkembangan Web 2.0 mengakibatkan perubahan kebiasaan komunitas pengguna internet yang begitu pesat. Para pakar meyakini bahwa saat ini telah terjadi perubahan dari era informasi (*information age*) menjadi era partisipasi (*participation age*) yang diakibatkan oleh adanya pemanfaatan internet yang begitu besar. Pengguna internet lebih mempercayai konten-konten asli buatan pengguna (*user generated*) dibanding dengan konten-konten resmi (*official*). Hal ini kemudian menciptakan prinsip "cari, minta, dan buat". Prinsip ini berarti jika pengguna internet membutuhkan informasi, usaha pertama yang dilakukan adalah mencari informasi tersebut melalui *search engine* atau media lainnya. Ketika informasi tidak didapatkan, maka pengguna internet berusaha meminta (*request*) kepada penyedia konten atau komunitas pengguna

internet untuk membuatkan artikel yang dibutuhkan. Usaha terakhir, apabila informasi yang sudah ada di internet dirasa masih kurang memuaskan, pengguna internet tingkat lanjut biasanya tergerak untuk memberikan kontribusi dalam komunitas dengan cara menulis artikel baru untuk melengkapi informasi yang sudah ada. Proses kontribusi pembuatan konten inilah yang mengakibatkan komunitas pengguna internet saat ini menjadi sumber informasi, bukan lagi sekedar menjadi pengguna informasi.

Baik dalam era Web 1.0 maupun Web 2.0 terdapat beberapa jenis teknologi internet yang mendukung adanya komunitas *online*. Pada era Web 1.0 terdapat aplikasi *mailing list* dan forum diskusi, dan pada Web 2.0 terdapat berbagai aplikasi yang dikelompokkan dalam suatu kategori khusus yang disebut dengan *social media*. *Social media* pada dasarnya adalah suatu aplikasi yang digunakan untuk berbagi atau mendiskusikan suatu informasi dengan orang lain. *Social media* dapat diaplikasikan dalam berbagai bentuk aplikasi, mulai dari yang tidak terikat waktu seperti forum diskusi hingga aplikasi yang *real time* seperti *instant messagging*. Adapun skripsi ini akan membahas tentang *knowledge market* yang merupakan salah satu bentuk aplikasi *social media*.

Dalam skripsi ini akan dibahas *knowledge market* yang memanfaatkan media internet sebagai perantara dalam penyebaran informasi, sehingga aplikasi ini disebut dengan istilah *internet-based knowledge market*. Beberapa aplikasi *knowledge market* yang populer di antaranya adalah *Yahoo! Answers* (http://answer.yahoo.com) dan *Answer Bag* (http://www.answerbag.com). Aplikasi ini populer di antaranya dikarenakan pengguna dapat mencari jawaban atas permasalahan dengan rentang yang sangat luas, mulai dari masalah yang ringan, seperti mencari produk printer, hingga masalah yang berat, seperti masalah rumah tangga. Banyak pengguna yang mengakui sangat terbantu dengan adanya aplikasi jenis ini.

Aplikasi *internet-based knowledge market* juga dibagi menjadi dua jenis yaitu dapat bersifat komersial atau non-komersial. Aplikasi yang dipilih dalam skripsi ini adalah aplikasi yang bersifat non-komersial. Aplikasi yang demikian sering pula disebut dengan istilah *community-based knowledge market*.

Aplikasi *knowledge market* memiliki kelebihan dalam hal fokus terhadap permasalahan bila dibandingkan dengan *social media* yang lainnya, khususnya forum diskusi. Selain itu data tersimpan secara terstruktur berdasarkan kategori permasalahan tertentu sehingga memudahkan untuk pencarian solusi untuk pengguna yang lainnya.

Permasalahan yang diperkirakan akan muncul dalam pengembangan aplikasi ini di antaranya adalah kompleksitas struktur basis data dan bagaimana merancang antar-muka (*interface*) yang mudah digunakan oleh pengguna.

Pengembangan web (*web development*) dilakukan dengan menggunakan pendekatan berorientasi-objek (*object-oriented*) dan menggunakan teknologi PHP dan MySQL. Agar lebih memudahkan, dipilih pula *framework* CodeIgniter sebagai *tool* pengembangan karena sifatnya yang *free* dan mendukung pendekatan berorientasi-objek. Pendekatan dan teknologi tersebut dipilih karena pengembangan dalam lingkungan ini banyak diklaim sangat memudahkan pengembang, baik untuk tahap pengembangan maupun pada tahap perawatan (*maintenance*). Selain itu walaupun kombinasi PHP dan MySQL sudah banyak digunakan, namun masih sangat jarang dokumentasi tentang pengembangan web yang menggunakan pendekatan berorientasi-objek untuk dua teknologi ini.

1.2 Rumusan Masalah

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

1. Berapa jumlah use case yang dihasilkan dari proses analisis kebutuhan?
2. Berapa jumlah tabel yang diperlukan untuk mengimplementasikan aplikasi *Internet-Based Knowledge Market*?
3. Mana di antara hasil pengujian local server dan remote server yang menunjukkan hasil yang lebih baik?
4. Apakah hasil seluruh pengujian telah sesuai dengan spesifikasi sistem yang telah ditentukan?

1.3 Batasan Masalah

Dalam perencanaan dan pembuatan skripsi ini perlu dilakukan pembatasan masalah. Pembatasan masalah yang diajukan dalam penyusunan tugas akhir ini antara lain:

1. Aplikasi knowledge market terdiri dari sistem layanan untuk dimanfaatkan pengguna, serta sistem administrasi untuk perawatan situs.
2. Aplikasi dibuat dengan menggunakan sistem operasi Microsoft Windows XP Professional, aplikasi server web Apache, bahasa pemrograman PHP, sistem basis data MySQL, dan *framework* CodeIgniter.

3. Pembahasan difokuskan pada perancangan, pembuatan, dan pengujian aplikasi dengan pendekatan metode pemrograman berorientasi -objek.
4. Layanan pencarian data pada sistem hanya dapat melakukan pencarian berdasarkan kata atau frase yang terdapat pada blok pertanyaan (judul dan isi).
5. Pengkategorian permasalahan difokuskan pada lingkup kegiatan akademis dan non-akademis di lingkungan Universitas Bra wijaya, Malang.

1.4 Tujuan

Tujuan dari pembuatan tugas akhir ini adalah membuat aplikasi *Internet-based knowledge market* dengan menggunakan bahasa pemrograman PHP dan sistem basis data MySQL yang dapat diakses melalui jaringan Internet. Aplikasi yang dih asilkan, dengan beberapa perbaikan, dapat digunakan untuk bertukar informasi seputar permasalahan di lingkungan Universitas Brawijaya, Malang.

1.5 Manfaat

Manfaat yang diharapkan dapat diperoleh dari tugas akhir ini adalah:

1. Memberikan dokumentasi tentang pengembangan aplikasi web dengan menggunakan pendekatan berorientasi -objek dan menggunakan PHP berorientasi-objek.
2. Aplikasi dapat digunakan untuk mempermudah pertukaran pengetahuan atau informasi.
3. Menambah pengalaman penulis dalam melakukan peng embangan perangkat lunak.
4. Melatih penulis untuk mengimplementasikan ilmu -ilmu yang telah didapat di bangku perkuliahan.

1.6 Sistematika Penulisan

Sistematika yang digunakan dalam penyusunan skripsi ini adalah sebagai berikut:

Bab I Pendahuluan

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

Bab II **Dasar Teori**

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

Bab III **Metode Penelitian**

Berisi tentang metode penelitian, perancangan, dan pengujian aplikasi.

Bab IV **Analisis dan Perancangan Sistem**

Membahas analisis aplikasi yang meliputi kebutuhan sistem, perancangan sistem, perancangan proses, perancangan basis data, dan perancangan antarmuka sistem.

Bab V **Implementasi Perangkat Lunak**

Membahas tentang implementasi hasil perancangan basis data dan antar-muka sistem.

Bab VI **Pengujian dan Analisis Perangkat Lunak**

Membahas pengujian hasil implementasi basis data dan sistem secara keseluruhan.

Bab VII **Penutup**

Memuat kesimpulan penelitian dan saran-saran.

BAB II
TINJAUAN PUSTAKA

Untuk menunjang penulisan skripsi dan pengembangan perangkat lunak *knowledge market* dengan pemrograman berorientasi objek ini, dibutuhkan kajian terhadap dasar teori yang relevan. Beberapa dasar teori yang dimaksud diantaranya adalah teori tentang *knowledge market*, rekayasa perangkat lunak, basis data, pemrograman berorientasi objek, teknologi berbasis web dan HTML yang akan digunakan untuk mengembangkan sistem tersebut. Bab ini akan mengulas secara lebih detail dasar teori tersebut.

2.1 Web 2.0

Istilah Web 2.0 pertama kali muncul dalam forum diskusi yang diadakan oleh penerbit O'Reilly dan MediaLive International. Istilah ini digunakan untuk menggambarkan terjadinya perubahan yang drastis dalam perkembangan teknologi internet pada kisaran tahun 2000.

Diskusi tersebut mengambil beberapa kesimpulan yang salah satunya menggambarkan adanya beberapa tingkatan aplikasi yang masuk dalam kategori Web 2.0, yaitu: [ORE-06]

- Level-3: yaitu aplikasi yang hanya bersifat *online* dan menggunakan prinsip "Semakin berguna ketika penggunanya semakin banyak". Aplikasi yang termasuk dalam level ini antara lain *eBay* dan *Wikipedia*.
- Level-2: yaitu aplikasi yang tadinya bersifat *offline* (berbasis desktop), namun dikembangkan menjadi aplikasi *online* (berbasis web) untuk memperbanyak jumlah pengguna. Contoh aplikasi yang termasuk dalam level ini adalah *Flickr*.
- Level-1: yaitu aplikasi yang tadinya bersifat *offline* (berbasis desktop), namun dikembangkan menjadi aplikasi *online* (berbasis web) yang menyebabkan dimungkinkannya penambahan fitur baru. Contoh aplikasi yang termasuk dalam level ini adalah *iTunes* dengan fitur toko musik *online* yang sudah terintegrasi di dalamnya.
- Level-0: yaitu aplikasi yang diedarkan baik dalam versi *online* maupun *offline*, dan keduanya tetap memiliki fungsi yang sama. Contoh aplikasi yang termasuk dalam level ini adalah *Google Maps*.

2.1.1 Collaboration

Salah satu unsur penting yang berperan dalam mempopulerkan teknologi Web 2.0 adalah kolaborasi (*collaboration*). Kolaborasi memiliki berbagai pengertian tergantung dari sudut pandang penggunanya. Salah satu pengertian dari kolaborasi adalah sebuah metode yang diyakini sebagai kunci yang akan memunculkan solusi - solusi atas permasalahan global seperti peperang an, kekerasan, kemiskinan, rasisme, krisis lingkungan hidup, dan pelanggaran hak asasi manusia. Dengan semakin berkembangnya teknologi, khususnya teknologi komunikasi dan komputer, kolaborasi berkembang pesat hingga terdapat pengkategorian sebagai berikut:

- Kolaborasi sinkron (*Synchronous Collaboration*), yaitu interaksi berbasis komputer yang diikuti beberapa orang dengan jeda waktu (*delay*) tidak lebih dari 5 detik. Contoh aplikasi tipe ini adalah *instant messaging*.
- Kolaborasi Semi-Sinkron (*Semi-Synchronous Collaboration*), yaitu interaksi yang dapat melebihi jeda waktu lebih dari 5 detik, namun terikat pada urutan waktu (*time frame*). Contoh aplikasi tipe ini adalah web seminar yang telah direkam dan dapat diputar ulang.
- Kolaborasi Asinkron (*Asynchronous Collaboration*), yaitu interaksi melalui suatu media tanpa batasan waktu. Contoh aplikasi tipe ini adalah *e-mail*, dan forum diskusi.

Terdapat beberapa keunggulan kolaborasi [COL-08], antara lain:

- Efisien dalam hal biaya dan waktu.
- Kualitas solusi cenderung lebi h baik.
- Dukungan terhadap suatu keputusan dapat diberikan dengan lebih efektif dan inovatif.
- Memudahkan akses para ahli pada permasalahan yang sedang dibahas.

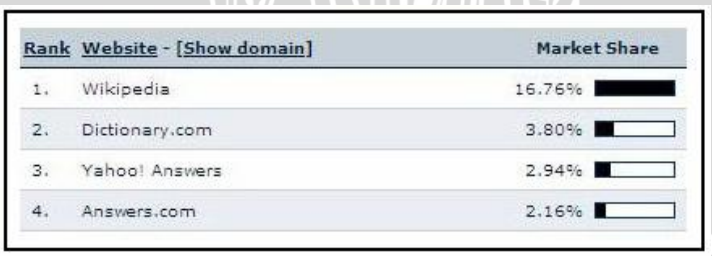
2.1.2 Knowledge Market

Para pakar, baik dari dunia pendidikan maupun dari dunia bisnis, semakin menyadari pentingnya suatu informasi. Para pakar pendidikan berusaha merumuskan informasi sehingga menjadi suatu hal yang bersifat ilmiah. Suatu bentuk paling mendasar dari teori informasi ini dikenal dengan istilah data, yaitu informasi yang direpresentasikan dalam satuan *bit* dan *byte*. Adapun istilah informasi dapat didefinisikan sebagai sekumpulan data yang dikaitkan dalam konteks tertentu oleh

manusia sehingga menjadikan data tersebut memiliki makna yang lebih berarti dan bermanfaat. Proses untuk memaknai suatu informasi dikenal dengan istilah pembelajaran (*learning*). Pada tingkatan yang lebih lanjut barulah didapatkan istilah pengetahuan (*knowledge*), yaitu pengaplikasian yang berupa aksi maupun komunikasi dari sekumpulan informasi. Pada perkembangan selanjutnya para akademisi dan pengusaha meletakkan pengetahuan sebagai bagian dari aset tak-nyata (*intangible assets*) yang mana hal ini semakin memperkuat pentingnya peran pengetahuan di dalam aktivitas masyarakat umum. [COL-08] [MAI-07:98]

Seiring dengan perkembangan teknologi internet, penyimpanan dan penyebaran informasi semakin mudah dilakukan, terlebih pada era Web 2.0. Terdapat berbagai jenis aplikasi Web 2.0 yang telah beredar di internet saat ini, di antaranya adalah wiki, *social media*, dan lain-lain. Jenis aplikasi yang dibahas dalam skripsi ini adalah jenis *knowledge market*, yaitu suatu media yang digunakan untuk melakukan pertukaran pengetahuan (*knowledge sharing*). Aplikasi yang telah beredar di internet dan termasuk dalam jenis ini adalah Yahoo! Answers (http://answer.yahoo.com) dan Answer Bag (http://www. answerbag.com).

Para pengamat tren teknologi internet menilai aplikasi jenis *knowledge market* semakin diminati oleh para pengguna internet, bahkan mengalahkan jenis wiki. Data dari Hitwise yang diambil pada 13 Mei 2006 menunjukkan bahwa *Yahoo! Answers* (yang merupakan jenis *knowledge market*), sedikit lebih unggul dibanding *Answers.com* (yang merupakan jenis wiki) dalam hal popularitas pasar [SUL-06]. Bahkan menurut *Comscore*, *Yahoo! Answers* merupakan referensi internet terpopuler kedua setelah *Wikipedia*.



Rank	Website - [Show domain]	Market Share
1.	Wikipedia	16.76%
2.	Dictionary.com	3.80%
3.	Yahoo! Answers	2.94%
4.	Answers.com	2.16%

Gambar 2.1 Data Market Share Situs Populer oleh Hitwise

(Sumber: [SUL-06])

Aplikasi jenis *knowledge market* dinilai lebih populer karena banyak memiliki keunggulan bila dibandingkan dengan jenis media lain seperti forum diskusi atau wiki. Keunggulan yang paling utama adalah faktor kemudahan dan efektivitas dalam melakukan pertukaran informasi. Sebagai contoh adalah kasus kepuasan pengguna

dalam menggunakan suatu produk. Dalam forum diskusi hal ini memang dapat diakomodasi, namun forum diskusi tidak memiliki aturan yang mengikat pengguna untuk tetap mengirimkan informasi (*posting*) yang terfokus untuk menjawab suatu permasalahan karena forum diskusi lebih sering dimanfaatkan untuk menyebarkan suatu informasi dan mengharapkan komentar dari pengguna yang lainnya. Selain itu, pada forum diskusi sering terjadi kiriman OOT (*Out of Topic*), yang mana pada *knowledge market* posting bernada OOT dapat ditekan jumlahnya.

Knowledge market dapat dikategorikan menurut beberapa faktor seperti disebutkan di bawah ini [SKY-01]:

- Berdasar tipe produk atau layanan. Sebagai contoh publikasi, konsultasi, dan hak kepemilikan intelektual.
- Berdasar tipe kemasan. Memiliki rentang dari proyek konsultasi, hingga saran pendek untuk menjawab pertanyaan yang spesifik.
- Berdasar mekanisme komunikasi. Sebagai contoh melalui suatu jaringan, telepon atau e-mail, hingga model pencocokan proyek.
- Berdasar lingkup dan rentang cakupan (scope and coverage). Memiliki rentang mulai dari layanan lokal hingga internasional.
- Berdasar tingkat kerahasiaan. Memiliki rentang mulai dari dapat diakses semua kalangan salah satunya melalui jalur internet, hingga hanya digunakan secara internal saja.

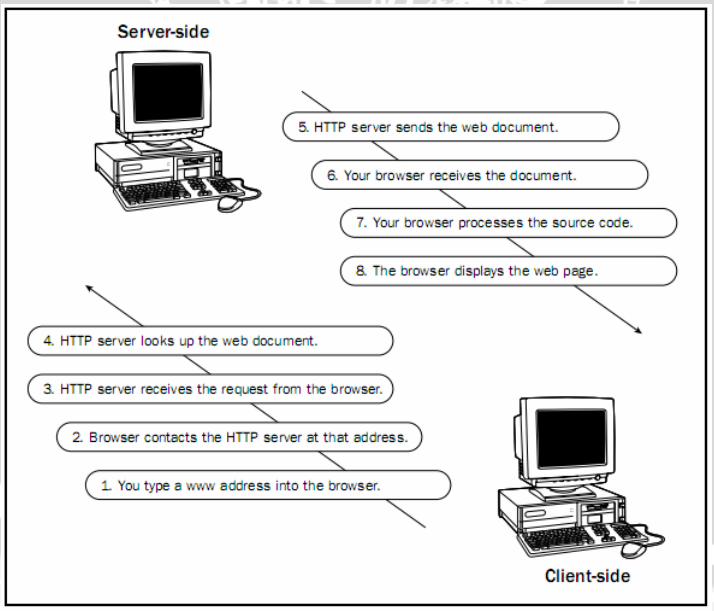
Berdasarkan pengkategorian tersebut, terdapat beberapa pilihan target yang dapat dijadikan sasaran. Target yang dipilih dalam skripsi ini adalah pengguna yang menganggap pengetahuan sebagai domain publik, sehingga aplikasi yang dikembangkan dapat disebut dengan aplikasi *free internet-based knowledge market*. Maksud dari *free* di sini adalah tidak terdapat kompensasi biaya tertentu untuk mendapatkan suatu jawaban dari pengguna lainnya (non-komersial).

2.2 Internet dan Web

Internet merupakan suatu jaringan komputer yang sangat kompleks. Pengguna internet, dan bahkan sebagian besar pengembang aplikasi web, tidak terlalu tertarik dengan proses pertukaran data yang terjadi di balik layar. Faktanya beberapa pengetahuan tentang proses di balik layar ini akan membantu pengembang web untuk mengembangkan aplikasi web yang lebih berkualitas.

Proses di balik layar yang cukup fundamental untuk dipahami oleh seorang pengembang web adalah tentang HTTP. Suatu proses HTTP pada intinya melibatkan dua tipe komputer, yaitu komputer server dan komputer klien. Komputer server adalah komputer yang di dalamnya tersimpan aplikasi web beserta kelengkapannya seperti server basis data. Komputer server harus selalu menyala karena difungsikan untuk selalu siap memberikan layanan jika diperlukan. Komputer klien adalah komputer yang akan meminta layanan, atau singkatnya adalah komputer yang akan kita gunakan untuk mengakses internet.

Langkah pertama yang memicu suatu rangkaian proses HTTP adalah ketika pengguna mengetikkan alamat pada browser. Setelah alamat dimasukkan, browser akan mengirimkan permintaan (*request*) melalui jaringan komputer yang kompleks hingga didapatkan komputer server dengan alamat yang dimaksud. Setelah server menerima permintaan dari *browser*, perintah dalam permintaan akan dieksekusi oleh server dan server akan membuat tanggapan (*response*) yaitu dengan menghasilkan sebuah halaman web yang akan dikirimkan kembali kepada komputer klien. Kemudian proses terakhir setelah halaman web diterima oleh *browser*, *browser* akan menampilkan halaman web dalam format dokumen tertentu kepada pengguna.



Gambar 2.2 Proses HTTP di Sisi Server dan di Sisi Klien

(Sumber: YOR-05: 11)

2.2.1 HTML dan xHTML

Setelah ditemukannya jaringan internet, para penemu internet mencari solusi dalam hal format apa dan bagaimana cara pertukaran data yang sesuai digunakan melalui internet. Akhirnya mereka mendapatkan format yang memudahkan suatu informasi berupa teks untuk dilakukan referensi-silang (*cross-reference*) yaitu dengan menggunakan *hypertext links*. Konsep format ini berkembang dan kemudian menjadi format baku yang diberi nama Hypertext Markup Language (HTML).

HTML sesungguhnya hanyalah file *plain text* yang didalamnya berisi *tag-tag* HTML. Karena merupakan file teks, maka file HTML dapat dibuka menggunakan editor teks pada umumnya. File HTML yang dibuka menggunakan ed itor teks akan sulit dibaca kontennya, namun ketika suatu file HTML dibuka menggunakan aplikasi yang disebut *web browser*, maka *tag-tag* HTML akan hilang dan konten halaman web akan ditampilkan lebih bermakna. [OLI-06]

Tag HTML memiliki ciri-ciri diawali dan diakhiri tanda lebih kecil dan lebih besar (<, >). Terkadang *tag* memiliki beberapa atribut yang mendefinisikan suatu fungsi tertentu.

Extensible Hypertext Markup Language (XHTML) merupakan format yang menyempurnakan format HTML dengan menambahkan aturan-aturan XML yang mengikat. Adapun aturan yang membedakan XHTML dan HTML adalah sebagai berikut:

- XHTML memiliki aturan penulisan *case-sensitive* yaitu semua tag harus ditulis dengan huruf kecil.
- Nilai pada suatu atribut harus diapit tanda petik, boleh petik tu nggal ataupun petik ganda.
- Penyingkatan penulisan atribut tidak diperboolehkan.
- Semua tag harus memiliki penutup tag.
- Aturan mengikat pada XHTML dipertimbangkan oleh para pengamat sebagai standar format di masa mendatang karena:
- Penulisannya lebih bersih.
- Memiliki aturan yang konsisten.
- Lebih dinamis untuk proyek yang besar.
- Dapat lebih cepat di-load.
- Lebih mudah diatur dan di-*update*.

2.2.2 PHP (Hypertext Preprocessor)

PHP (dulu *Personal Home Page*, sekarang *PHP: Hypertext Preprocessor*) merupakan bahasa pemrograman sisi server yang dikembangkan secara bersama oleh para programmer dari seluruh dunia yang menekuni dunia *open-source*. PHP populer digunakan khususnya untuk melakukan koneksi antara halaman web dengan server basis data. [CON-04:ix]

PHP, khususnya versi 5 ke atas, telah mendukung pemrograman aplikasi web dengan pendekatan berorientasi objek secara penuh. Pada versi 3 PHP telah menerapkan konsep penggunaan objek, namun baru sebatas pada klas/objek yang bersifat *built-in* saja, artinya pengguna tidak dapat membuat klas/objek kreasinya sendiri. [LAV-06:11] Dengan diimplementasikannya konsep OO pada PHP, pengembangan aplikasi web dapat dilakukan dengan pendekatan yang lebih moduler, sehingga lebih memudahkan pengelola web untuk melakukan *maintenance* terhadap aplikasi web.

2.2.3 Apache Web Server

Untuk membuat sebuah pemrograman web dinamis, diperlukan sebuah *web server*. Ada banyak *web server* yang berkembang dan sering digunakan dalam membangun aplikasi berbasis web, salah satunya adalah Apache. Apache memiliki beberapa kelebihan antara lain adalah:

- *Free of Charge*, yang berarti pengguna tidak harus membayar lisensi kepada pembuat untuk menggunakannya.
- *Dapat diakses dan digabung dengan berbagai aplikasi lain (database server, ssl, dan sebagainya).*
- *Waktu pemrosesan lebih cepat dan tangguh dengan konfigurasi yang benar.*
- *Dapat dilakukan setting dan instalasi sesuai dengan kebutuhan dengan adanya modules dan DSO-nya.*
- *Memiliki kemampuan* advanced settings dan configuration support.

Dengan berbagai keunggulan tersebut, Apache sangat bagus jika dikombinasikan dengan aplikasi lainnya. Penggabungan yang paling sering dilakukan adalah dengan menggabungkan Apache, PHP, dan MySQL yang sama-sama bekerja, baik di server Linux maupun Windows.

2.2.4 CSS

Cascading Style Sheets (CSS) adalah bahasa desain yang sederhana yang ditujukan untuk mempermudah pengaturan tampilan dari suatu halaman web. CSS dapat digunakan untuk mengatur tata warna teks, jenis *font*, spasi antar paragraf, ukuran kolom teks, latar belakang baik berupa warna maupun gambar, dan efek lainnya yang berkaitan dengan tampilan halaman web.

CSS memiliki beberapa keunggulan, antara lain:

- Dengan menggunakan CSS, pengaturan tampilan situs menjadi terpusat pada satu atau beberapa dokumen saja, yang mana hal ini membe rikan konsistensi tampilan serta memudahkan untuk dilakukannya *update* terhadap tampilan situs. Hal ini dimungkinkan karena format-format tampilan dipisahkan dengan struktur dokumen dalam dua *file* yang berbeda.
- Pengguna situs dapat menggunakan style lain selain style standar yang diberikan oleh pembuat situs. Hal ini dapat meningkatkan akesibilitas dari situs tersebut, misalnya pengguna menggunakan style dengan tingkat kontras yang tinggi sehingga membuat konten menjadi lebih mudah untuk dibaca.
- CSS memungkinkan konten situs untuk diakses menggunakan berbagai media yang berbeda, misalnya dengan menggunakan perangkat telepon genggam, tanpa mengurangi makna konten tersebut.
- Penggunaan CSS mempercepat waktu *loading* suatu situs karena terjadi pengurangan penggunaan ruang di *hard disk* serta menggunakan *bandwidth* lebih kecil.

Seperti halnya HTML, penulisan CSS juga memiliki aturan tertentu. Contoh format penulisan skrip CSS disajikan pada gambar 2.3.

```
body {
     color: black;
}
```

Gambar 2.3 Format Penulisan Skrip CSS

Dalam aturan CSS penulisan spasi dan pergantian baris (line break) tidak berpengaruh pada makna aturan CSS, sehingga programmer cenderung membuat pola penulisan renggang untuk mempermudah pembacaan dokumen. Adapun aturan pada contoh di atas akan mengubah tampilan terkait pada *tag body* di dokumen HTML.

Aturan-aturan yang mengatur satu bagian atau terkait satu *tag* seperti ini disebut *selector*. Aturan penulisan yang berlaku yaitu *selector* diikuti tanda kurung kurawal yang didalamnya berisi aturan-aturan yang dipisahkan tanda titik koma. Sebuah aturan dikenal pula dengan istilah *declaration*. Penulisan *declaration* diikuti dengan tanda titik dua dan kemudian nilai yang terkait.

2.3 Dasar Basis Data

2.3.1 Structured Query Language (SQL)

Structured Query Language (SQL) merupakan bahasa yang digunakan pada basis data relasional. Bahasa ini dibuat seiring dengan semakin dibutuhkannya sistem basis data pada berbagai aplikasi, yang mana secara otomatis muncul kebutuhan akan suatu bahasa yang standar dan terstruktur untuk melakukan operasi-operasi pada basis data. [CON-04:246]

Perintah-perintah SQL dikelompokkan menjadi 5 macam, yaitu:

- *Data Definition Language* (DDL), yaitu kelompok perintah SQL yang digunakan untuk membuat suatu objek pada basis data. Objek basis data yang dimaksud terdiri dari *database*, *table*, *index* dan *view*. Dengan kata lain DDL digunakan untuk mendefinisikan suatu kerangka basis data. Perintahnya adalah :
  - *create* : untuk membuat/menciptakan objek basis data.
  - *alter* : untuk memodifikasi/mengubah objek basis data.
  - *drop* : untuk menghapus objek basis data.
- *Data Manipulation Language* (DML), yaitu kelompok perintah yang digunakan untuk mengoperasikan atau memanipulasi isi basis data. SQL menyediakan 4 perintah DML, yaitu:
  - *select* : digunakan untuk mengambil data dari basis data.
  - *delete* : digunakan untuk menghapus data dari basis data
  - *insert* : digunakan untuk menambahkan data pada basis data.
  - *update* : digunakan untuk memodifikasi data pada basis data.
- *Security*, yaitu perintah-perintah yang digunakan untuk menjamin keamanan data. Perintah *security* terdiri atas:
  - *Grant* : memberi akses kepada pengguna tertentu untuk melakukan akses ke basis data.
  - *Revoke*: mencabut hak akses dari pengguna.

- *Integrity*, yatiu perintah-perintah yang digunakan untuk menjaga kesatuan data. Contohnya adalah perintah *recover table*, untuk memperbaiki tabel pada basis data.
- *Auxilary*, yaitu perintah-perintah pelengkap atau tambahan seperti *unload* dan *rename*.

2.3.2 MySQL

Basis data (*database*) telah menjadi bagian yang tak terpisahkan di hampir seluruh bagian hidup manusia. Tanpanya, banyak hal yang kita lakukan akan menjadi sangat membosankan, atau bahkan tidak mungkin untuk dilakukan. Bank, universitas, dan perpustakaan adalah tiga contoh organisasi yang sangat bergantung pada sistem basis data. Demikian pula aplikasi-aplikasi di internet seperti *search engine*, *online shopping*, dan lainnya dapat menjadi mustahil untuk dibangun tanpa adanya basis data. Sebuah basis data yang diimplementasikan dan diberi antarmuka pada sebuah komputer sering disebut sebagai sebuah server basis data.

MySQL merupakan *Relational Database Management System* (RDBMS) yang didistribusikan secara gratis, dimana setiap orang bebas untuk menggunakannya. Sebagai server basis data, MySQL dapat dikatakan lebih unggul dibandingkan server basis data lainnya dalam hal proses *query* data. [NOR-04:1-3]

2.4 MVC Framework

Software framework adalah abstraksi yang di dalamnya memuat kode-kode umum yang fungsinya dapat diubah oleh penggunanya. *Framework* serupa dengan *library*, namun aliran kontrol program pada *framework* diatur oleh *framework* sendiri, bukan oleh pemanggil fungsi.

Dalam arsitekturnya, *framework* memiliki dua komponen utama. Komponen tersebut adalah *frozen spots* dan *hot spots*. *Frozen spots* adalah bagian dari *framework* yang memuat komponen dasar dari sistem, bagian ini tidak dapat diubah oleh pemrogram. Bagian kedua yaitu *hot spots* adalah bagian dari *framework* yang memuat kode-kode yang dibuat oleh pemrogram sesuai dengan kebutuhan yang telah dirancang sebelumnya untuk menambah fungsi pada perangkat l unak yang akan dibuat.

Beberapa *framework* pemrograman berbasis web telah mengimplementasikan konsep *model-view-controller* (MVC) di dalam arsitektur sistemnya. Konsep MVC bertujuan untuk memudahkan dilakukannya modifikasi pada perangkat lunak. Dengan

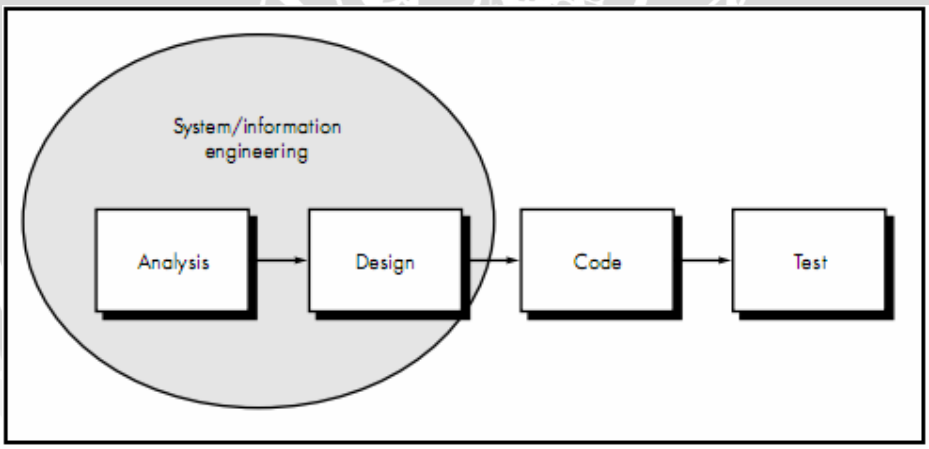
penggunaan konsep ini, modifikasi pada satu bagian diharapkan tidak akan mempengaruhi performansi bagian lainnya. Sesuai dengan namanya, MVC membagi sistem menjadi tiga bagian, yaitu *model* (menangani proses manipulasi data), *view* (menangani tampilan), dan *controller* (mengelola masukan dari pengguna).

2.4.1 CodeIgniter

Skripsi ini menggunakan *framework* pemrograman aplikasi PHP dengan pendekatan berorientasi objek yaitu CodeIgniter. Framework ini bersifat *open source* dan memiliki dokumentasi yang jelas dan lengkap .CodeIgniter merupakan salah satu *framework* yang menggunakan konsep MVC di dalamnya.

2.5 Teori Rekayasa Perangkat Lunak

Pengembangan aplikasi *Internet-Based Knowledge Market* dilakukan dengan pendekatan model air terjun (*waterfall*). Pengembangan aplikasi model air terjun dimulai dari proses analisis (*analysis*), kemudian dilanjutkan dengan proses perancangan (*design*), pengkodean (*code*), dan diakhiri dengan proses pengujian (*test*). Urutan proses ini ditunjukkan pada gambar *berikut*.



Gambar 2.4 Urutan Proses Pengembangan Aplikasi Model Air Terjun [PRE-01: 56]

Proses pertama yang dilakukan dalam model ini adalah proses analisis kebutuhan aplikasi. Proses ini dilakukan dengan cara mengumpulkan informasi - informasi terkait dengan aplikasi yang nantinya akan dik embangkan dalam bentuk fungsi, alur, performansi, dan antar muka dari aplikasi tersebut.

Proses kedua yang dilakukan adalah proses perancangan aplikasi. Proses ini dilakukan dengan berfokus pada empat atribut acuan yaitu struktur data, arsitektur

aplikasi, representasi antar muka, dan perincian algoritma. Proses perancangan bertujuan untuk menerjemahkan kebutuhan-kebutuhan aplikasi ke dalam suatu representasi aplikasi yang dapat diuji kualitasnya sebelum melakukan proses pengkodean.

Proses ketiga yang dilakukan adalah proses pengkodean. Proses ini bertujuan untuk menerjemahkan representasi rancangan ke dalam bentuk kode yang dapat dibaca oleh mesin komputer. Proses ini sangat bergantung dari seberapa rinci representasi rancangan telah dibuat.

Proses terakhir yang dilakukan adalah proses pengujian. Proses ini secara garis besar dibagi menjadi dua bagian, yaitu pengujian logika di dalam aplikasi dan pengujian fungsional di luar aplikasi. Pengujian logika bertujuan untuk mendeteksi kesalahan logika yang terdapat dalam kode program, sedangkan pengujian fungsional menguji kecocokan antara keluaran program yang telah dibuat dengan keluaran program yang diharapkan. [PRE-01]

Adapun dikarenakan aplikasi dalam skripsi ini adalah jenis aplikasi berbasis web (*web-based application*), pada prakteknya perlu dilakukan perincian pada tiap-tiap proses di dalamnya. Perincian proses pengembangan aplikasi berbasis web didasarkan pada empat langkah utama [AND-06], yaitu:

1. Membangun model data. Pada langkah pertama ini dibahas informasi apa saja yang akan disimpan dan bagaimana informasi akan ditampilkan.
2. Membangun model transaksi. Pada langkah ini dibahas *query* apa saja yang dibutuhkan.
3. Merancang aliran halaman. Pada langkah ini dibahas bentuk interaksi antara pengguna dan sistem.
4. Mengimplementasi tiap-tiap halaman. Pada langkah terakhir ini dilakukan penulisan *script* untuk tiap-tiap halaman.

2.6 Analisis Berorientasi-Objek

Pengembangan perangkat lunak yang baik selalu didahului dengan serangkaian analisis kebutuhan perangkat lunak. Pengembangan perangkat lunak beroroientasi-objek memiliki analisis kebutuhan yang berbeda dengan analisis kebutuhan pada pemrograman terstruktur.

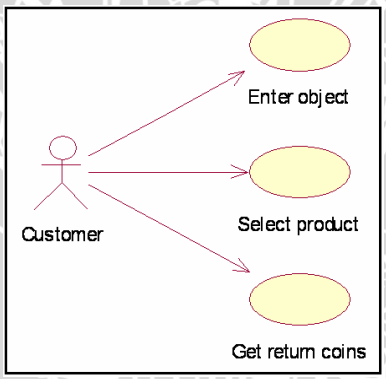
Dokumentasi analisis berorientasi-objek salah satunya dilakukan dengan menggunakan *Unified Modeling Language* (UML). Setidaknya terdapat empat diagram

utama yang digunakan dalam UML, yaitu *use case diagram*, *class diagram*, *sequence diagram*, dan *state diagram*. [KUR-05a:33]

2.6.1 Diagram Use Case

Diagram use case menggambarkan fungsionalitas yang diharapkan dari sebua h sistem. Diagram use case memberi tekanan pada "apa" yang diperbuat sistem, dan bukan "bagaimana" sistem bekerja. Sebuah use case merepresentasikan sebuah interaksi antara aktor dengan sistem. Use case merupakan sebuah pekerjaan tertentu, misalnya *login* ke sistem, membuat sebuah daftar belanja, dan sebagainya. Seorang/sebuah aktor adalah sebuah entitas manusia atau mesin yang berinteraksi dengan sistem untuk melakukan pekerjaan-pekerjaan tertentu.

Use case diagram dapat sangat membantu bila kita sedang men yusun kebutuhan sebuah sistem, mengkomunikasikan rancangan dengan klien, dan merancang test case untuk semua feature yang ada pada sistem.

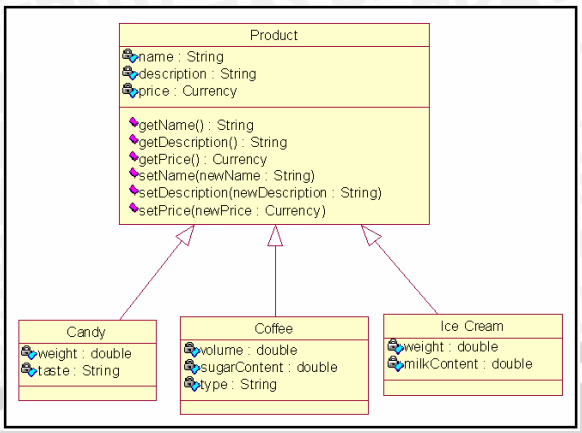


Gambar 2.5 Contoh Diagram *Use Case* (Sumber: [KUR-05a])

2.6.2 Diagram Klas

Klas (*class*) adalah sebuah spesifikasi y ang jika diinstansiasi akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan desain berorientasi objek. Klas menggambarkan keadaan suatu sistem (disebut atribut/properti), sekaligus menawarkan layanan untuk memanipulasi keadaan tersebut (disebut metoda/fungsi).

Diagram klas menggambarkan struktur dan deskripsi klas, *package* dan objek beserta hubungan satu sama lain seperti *containment*, pewarisan, asosiasi, dan lain-lain.



Gambar 2.6 Diagram Klas
(Sumber: [KUR-05a])

Klas memiliki tiga area pokok, yaitu nama (dan *stereotype*), atribut, dan metoda. Atribut dan metoda dapat memiliki salah satu sifat berikut:

- *Private*, tidak dapat dipanggil dari luar klas yang bersangkutan.
- *Protected*, hanya dapat dipanggil oleh klas yang bersangkutan dan anak -anak yang mewarisinya.
- *Public*, dapat dipanggil oleh siapa saja.
- Terdapat hubungan antar klas, antara lain:
- Asosiasi, yaitu hubungan statis antar klas. Umumnya menggambarkan klas yang memiliki atribut berupa klas lain, atau klas yang harus mengetahui eksistensi klas lain. Panah *navigability* menunjukkan arah *query* antar klas.
- Agregasi, yaitu hubungan yang menyatakan bagian ("terdiri atas...").
- Pewarisan, yaitu hubungan hirarkis antar klas. Klas dapat diturunkan dari klas lain dan mewarisi semua atribut dan metoda klas asalnya dan menambahkan fungsionalitas baru, sehingga ia disebut anak dari klas yang diwarisinya. Kebalikan dari pewarisan adalah generalisasi.
- Hubungan dinamis, yaitu rangkaian pesan (*message*) yang di-*passing* dari satu klas kepada klas lain. Hubungan dinamis dapat digambarkan dengan menggunakan *sequence diagram* yang akan dijelaskan kemudian.

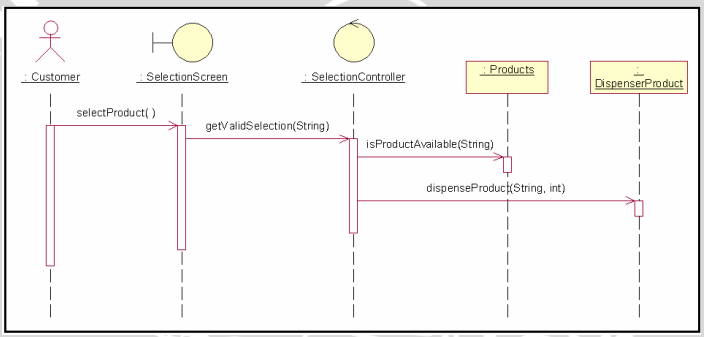
2.6.3 Diagram Sequence

Sequence diagram menggambarkan interaksi antar objek di dalam dan di sekitar sistem (termasuk pengguna, *display*, dan sebagainya) berupa pesan yang digambarkan

terhadap waktu. Diagram *sequence* terdiri atas dimensi vertikal (waktu) dan dimensi horizontal (objek-objek yang terkait).

Diagram *sequence* biasa digunakan untuk menggambarkan skenario atau rangkaian langkah-langkah yang dilakukan sebagai respon dari sebuah *event* untuk menghasilkan *output* tertentu. Diawali dari apa yang men-*trigger* aktivitas tersebut, proses dan perubahan apa saja yang terjadi secara internal dan *output* apa yang dihasilkan.

Masing-masing objek, termasuk aktor, memiliki *lifeline* vertikal. Pesan digambarkan sebagai garis berpanah dari satu objek ke objek lainnya. Pada fase desain berikutnya, pesan akan dipetakan menjadi operasi/metoda dari klas. *Activation bar* menunjukkan lamanya eksekusi sebuah proses, biasanya diawali dengan diterimanya sebuah *message*.



Gambar 2.7 Diagram *Sequence* (Sumber: [KUR-05a])

2.6.4 Diagram State

Setiap objek di dalam sistem memiliki daur hidup. Secara sederhana, daur hidup itu meliputi bagaimana objek tersebut dibuat, bagaimana objek tersebut berinteraksi dengan objek lain, dan bagaimana objek tersebut mati.

Dalam diagram state, ada 2 hal penting yang harus diperhatikan, yaitu *states* dan *events*. State adalah tingkah laku utama objek dalam kurun waktu tertentu. Sedangkan *event* adalah sesuatu yang menstimulasi *state*.

2.6.5 Pengujian Berorientasi-Objek

Pengujian perangkat lunak merupakan elemen yang sangat penting dalam daur hidup pengembangan perangkat lunak yakni di antaranya untuk menjamin kualitas perangkat lunak, serta menggambarkan peninjauan terhadap spesifikasi, perancangan, dan implementasi perangkat lunak.

2.6.6 Strategi Pengujian

Strategi untuk melakukan pengujian perangkat lunak dimulai dari "pengujian kecil" menuju ke "pengujian besar". Pengujian berorientasi objek dimulai dari *unit testing*, bergerak menuju *integration testing*, dan berakhir pada *validation testing*. [KUR-05b:30]

2.6.7 Teknik Pengujian

Pengujian perangkat lunak memerlukan perancangan kasus uji (*test case*) agar dapat menemukan kesalahan dalam waktu yang singkat dan dengan usaha yang minimum. Metode perancangan kasus uji menyediakan mekanisme penting yang dapat membantu menjamin kompleksitas pengujian dan keandalan yang tinggi untuk menemukan kesalahan. Pengujian yang digunakan meliputi pengujian kotak putih dan pengujian kotak hitam.

2.7 Dasar Teori Keamanan Aplikasi Web

Salah satu poin penting yang perlu diperhatikan dalam pembuatan aplikasi web adalah masalah keamanan. Pada dasarnya terdapat dua kategori keamanan seputar web, yaitu keamanan jaringan dan keamanan aplikasi. Dalam kategori keamanan jaringan dibahas kajian-kajian keamanan yang terkait dengan akses pada server, sedangkan dalam kategori kemanan aplikasi dibahas kajian-kajian keamanan yang terkait dengan akses pada aplikasi.

2.7.1 Keamanan Jaringan

- *Denial of Service* (DoS)

  Denial of Service (DoS) atau *Distributed Denial of Service* (DDoS) adalah tipe serangan yang berbentuk kiriman data permintaan layanan berjumlah besar yang yang membanjiri server sehingga pengguna lain yang ingin mengakses layanan pada web tersebut menjadi tidak tertangani. Masalah DoS tingkat dasar dapat dicegah dengan melakukan pengaturan yang benar pada aplikasi server yang digunakan (misalnya aplikasi Apache).

- Virus dan Worm

  Sama halnya dengan *personal computer* (PC), server juga merupakan suatu sistem komputer yang rentan terhadap serangan virus dan worm. Virus

adalah skrip program tereksekusi yang dapat memperbanyak diri dan diprogram untuk melakukan suatu manipulasi terhadap suatu bagian dari sistem komputer yang seringkali bersifat merusak. Worm adalah skrip program tereksekusi yang memperbanyak diri untuk dijalankan dalam memori komputer sehingga berdampak pada lebih lambatnya kerja sistem komputer dibandingkan pada keadaan normal. Masalah virus dan worm dapat dicegah dengan melakukan instalasi program antivirus pada komputer server. [SPE -03]

- *Eavesdropping* (dari sisi jaringan)

  Eavesdropping adalah proses pembacaan data oleh pihak yang tidak memiliki hak. Berbeda dengan masalah lainnya, masalah *eavesdropping* ini dapat dimasukkan ke dalam dua kategori keamanan sekaligus, baik kategori keamanan jaringan maupun keamanan aplikasi. Masalah *eavesdropping* dari sisi jaringan dapat dicegah dengan mengimplementasikan sistem *Secure Sockets Layer* (SSL). SSL adalah protokol jaringan komputer yang akan melakukan proses enkripsi pada suatu *session* layanan dalam server sehingga mempersulit pembacaan data oleh pengguna tanpa hak.

2.7.2 Keamanan Aplikasi

- *Eavesdropping* (dari sisi aplikasi)

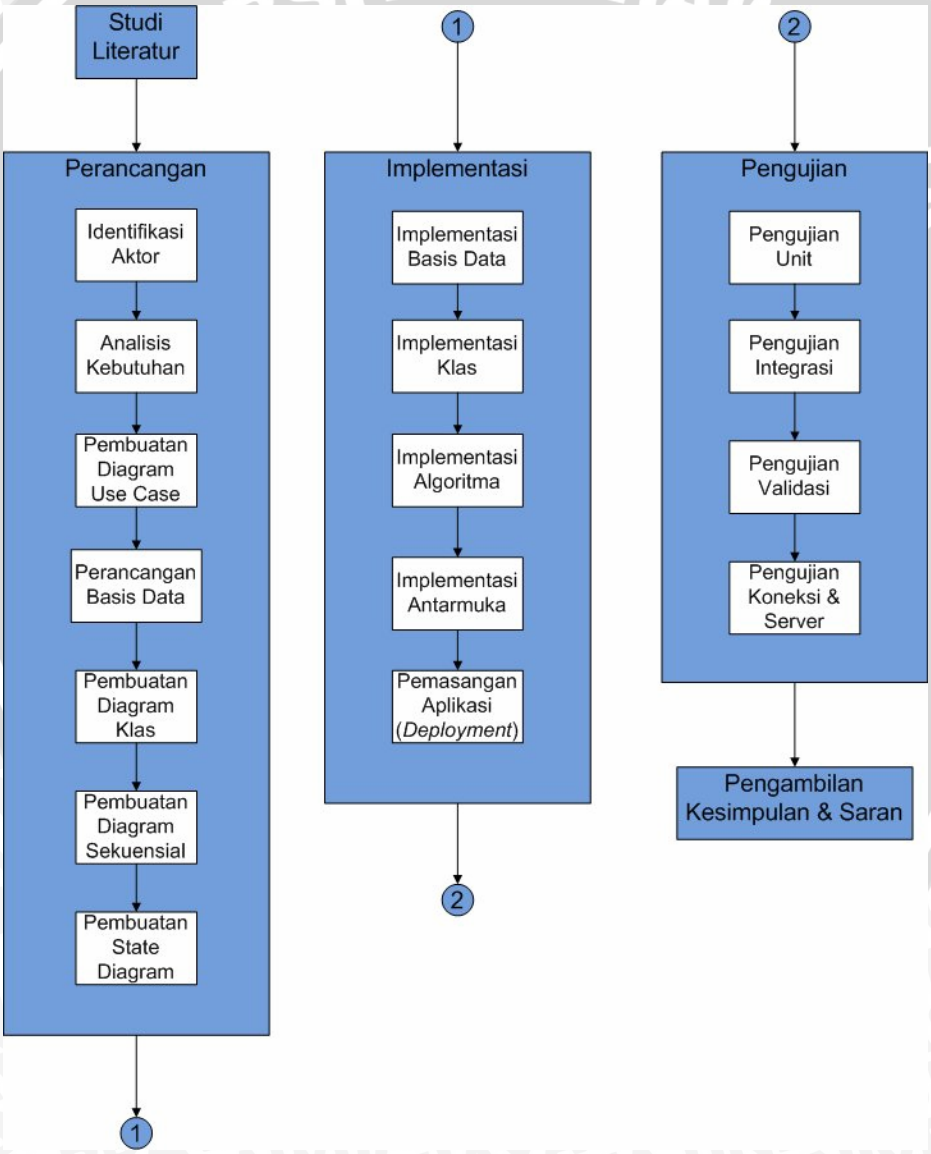
  Dari sisi keamanan aplikasi, masalah *eavesdropping* dapat dicegah dengan menerapkan proses autentikasi (*authentication*) dan autorisasi (*authorization*) secara benar. Proses autentikasi adalah proses di mana sistem dapat mengenali pengguna dari identitas yang dimilikinya, sedangkan autorisasi adalah proses membedakan antara pengguna yang memiliki hak akses dan yang tidak terhadap suatu *resource* dalam sistem. Kedua proses ini diimplementasikan dengan penggunaan nama pengguna dan kata sandi untuk semua pengguna sistem. Kata sandi juga dapat lebih diamankan dengan disimpan dalam bentuk terenkripsi.

- *Cross*-site *scripting* (XSS)

  Cross-site *scripting* adalah proses memasukkan skrip berbahaya oleh pengguna pada halaman web yang ditampilkan secara dinamis. Dampak dari XSS dapat bervariasi, mulai dari pencurian informasi vital sampai pada perusakan data pada server. Masalah XSS tingkat dasar dapat dicegah dengan melakukan *tag filtering* dalam tiap *form* yang dikirimkan ke server. [CRO-07]

BAB III
METODOLOGI PENELITIAN

Pada bab ini akan dibahas berbagai tahapan penelitian yang digunakan untuk mengembangkan aplikasi *Internet-Based Knowledge Market*. Tahapan penelitian yang digunakan di antaranya adalah studi literatur, metode perancangan, implementasi, dan pengujian. Pada akhir proses pengembangan akan ditarik kesimpulan dan saran untuk proses perbaikan aplikasi di masa yang akan datang. Tahapan penelitian secara ringkas digambarkan pada gambar 3.1.



Gambar 3.1 Tahapan Pembuatan Aplikasi *Internet-Based Knowledge Market*

3.1 Studi Literatur

Studi literatur dilakukan untuk mendapatkan pengetahuan yang dapat mendukung penyusunan skripsi, terlebih sebagai dasar dalam melakukan perancangan. Teori-teori pendukung tersebut meliputi: dasar teori *knowledge market*, rekayasa perangkat lunak, pemrograman berorientasi-objek, *web 2.0*, internet, basis data, *web server* apache, dan *framework* CodeIgniter.

3.2 Perancangan Perangkat Lunak

Berdasarkan dari kajian literatur, tahap perancangan perangkat lunak dilakukan dengan melakukan perancangan kebutuhan sistem secara global, kemudian tiap-tiap kebutuhan diperdetail ke dalam suatu blok diagram sesuai dengan metode perancangan perangkat lunak berorientasi-objek.

Perancangan diawali dengan melakukan identifikasi aktor, kemudian dilanjutkan dengan analisis kebutuhan, pembuatan diagram *use case*, perancangan basis data, pembuatan diagram klas, pembutan diagram sekuensial, dan diakhiri dengan pembuatan *state diagram*.

3.3 Implementasi Perangkat Lunak

Implementasi perangkat lunak pada dasarnya sudah dapat dilakukan ketika suatu blok diagram dinilai cukup lengkap untuk dikerjakan dalam bentuk kode (*coding*). Setelah selesai, bila dimungkinkan langsung dilakukan pengujian blok untuk mencegah terjadinya *error* yang lebih rumit.

Tahap implementasi yang pertama kali dilakukan adalah implementasi basis data, kemudian dilanjutkan dengan implementasi klas ke dalam *file* PHP. Setelah itu dilakukan implementasi algoritma dan antarmuka. Tahap terakhir adalah melakukan pemasangan (*deployment*) aplikasi pada *local server* dan *remote server*. Pemasangan pada *local server* dilakukan dengan cara meletakkan seluruh *file* pada lokasi yang ditentukan oleh aplikasi server Apache yang digunakan, sedangkan pemasangan pada *remote server* dilakukan dengan cara memindahkan seluruh *file* ke server dengan menggunakan *File Transfer Protocol* (FTP).

3.4 Pengujian Perangkat Lunak

Pengujian perangkat lunak pertama-tama bertujuan untuk mencegah terjadinya kesalahan, khususnya kesalahan perintah (syntax error) pada awal pembuatan blok kode.

Selanjutnya pengujian dilakukan untuk mengukur kesesuaian kinerja perangkat lun ak dengan spesifikasi pada perancangan yang sudah dibuat sebelumnya.

Tahapan pengujian terdiri dari pengujian unit dengan metode pengujian kotak putih (*white box*), pengujian integrasi dan validasi dengan metode pengujian kotak hitam (*black box*), dan terakhir dilakukan pengujian koneksi untuk mengukur performa server yang digunakan dalam penelitian.

3.5 Pengambilan Kesimpulan dan Saran

Pengambilan kesimpulan didasarkan pada kesesuaian antara tujuan penelitian dengan hasil yang berupa perangkat lunak. Penuli san saran bertujuan untuk penyempurnaan dan pengembangan perangkat lunak di masa yang akan datang.

BAB IV
PERANCANGAN

Bab ini membahas mengenai perancangan perangkat lunak menggunakan metode berorientasi objek. Perancangan perangkat lunak ini terdiri at as dua tahap, tahap pertama adalah analisis kebutuhan *(requirement analysis)* perangkat lunak, dan tahap kedua adalah perancangan perangkat lunak. Tahap analisis kebutuhan perangkat lunak dimodelkan dengan pemodelan UML *(Unified Modelling Language)* menggunakan diagram *use case*. Tahap perancangan perangkat lunak dimodelkan dengan pemodelan UML menggunakan diagram klas (*class diagram*) dan diagram sekuensial (*sequence diagram*).

4.1 Analisis Kebutuhan (*Requirement Analysis*)

Pada tahap analisis kebutuhan ini dil akukan proses identifikasi, pemahaman dan penetapan kebutuhan-kebutuhan pengguna (*user requirements*). Hasil dari proses analisis kebutuhan akan dimodelkan dengan pemodelan UML menggunakan diagram *use case*. Analisis kebutuhan bertujuan untuk mendeskripsikan kebutuhan-kebutuhan fungsional maupun non fungsional yang harus disediakan oleh sistem agar dapat memenuhi kebutuhan-kebutuhan pengguna (*user*).

4.1.1 Identifikasi Aktor

Tahap ini bertujuan untuk melakukan identifikasi dan analisis terhadap aktor - aktor yang berinteraksi dengan aplikasi *knowledge market*. Tabel 4.1 memperlihatkan empat aktor beserta deskripsinya yang merupakan hasil dari proses identifikasi aktor. Empat jenis aktor yang terdapat dalam sistem dibagi menjadi dua kategori yaitu *basic user* dan *super user*. *Basic user* terdiri dari tamu dan anggota, *super user* terdiri dari *sweeper* dan *admin*.

Tabel 4.1 Deskripsi Aktor

Aktor	Deskripsi Aktor
Tamu	Tamu merupakan semua aktor pengguna yang belum sah (belum melakukan *log in*) dan tidak dapat mengakses beberapa fungsionalitas sistem, khusunya fungsi yang berhubungan dengan pengiriman data (*posting* dan *rating*).
Anggota	Anggota merupakan aktor pengguna yang sah dan sudah divalidasi melalui proses *log in* dan dapat mengakses sistem sesuai dengan otoritasnya . Aktor ini dapat menjalankan semua fungsi yang berkaitan dengan pemanfaatan aplikasi.

Aktor	Deskripsi Aktor
Sweeper	*Sweeper* merupakan aktor yang memiliki hak akses lebih luas dibanding anggota biasa. Sweeper bertugas untuk membantu rutin pekerjaan -pekerjaan *admin* yang berkaitan dengan manajemen data (kiriman pertanyaan dan jawaban).
Administrator	*Administrator* (juga disebut *Admin*) merupakan aktor yang memiliki hak akses paling luas. Aktor ini dapat menjalankan semua fungsi manajemen data dan manajemen keanggotaan.

Sumber: Analisis Kebutuhan

4.1.2 Daftar Kebutuhan

Daftar kebutuhan ditampilkan dengan sebuah tabel yang berfungsi untuk mendefinisikan kebutuhan-kebutuhan yang harus disediakan oleh sistem perangkat lunak. Tabel daftar kebutuhan mempunyai atribut ID kebutuhan, nama akt or yang berhubungan dengan kebutuhan dan nama *use case* yang merepresentasikan fungsionalitas untuk memenuhi kebutuhan tersebut. Daftar kebutuhan fungsional (F) dan non-fungsional (N) sistem yang dikembangkan pada tugas akhir ini ditunjukkan pada Tabel 4.2.

Tabel 4.2 Daftar Kebutuhan Sistem Informasi

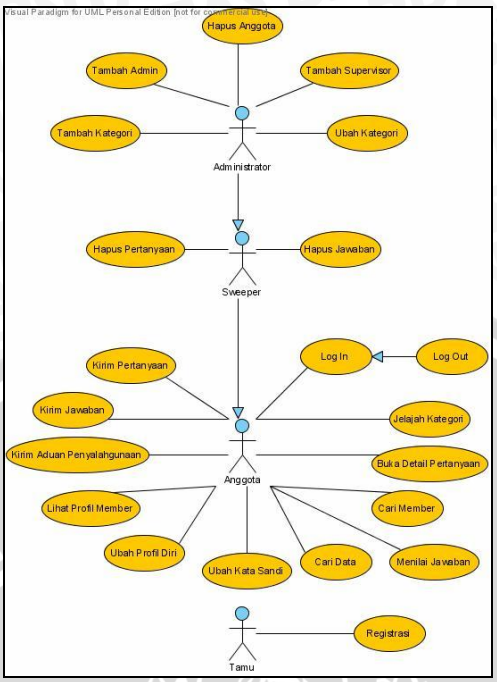
ID.	Kebutuhan	Aktor	*Use Case*
F1	Sistem harus dapat menyediakan fasilitas registrasi bagi tamu yang ingin menjadi anggota dalam situs. Fasilitas registrasi berupa beberapa form isian yang harus dilengkapi oleh c alon anggota.	Tamu	Registrasi
F2	Sistem harus dapat menyediakan fasilitas *log in* bagi *member* (anggota, *sweeper*, dan *admin*) yang berfungsi untuk melakukan proses autentifikasi terhadap *member*. Sistem harus dapat melakukan validasi terhadap *member* yang sah.	Anggota, *Sweeper*, *Admin*	*Log In*
F3	Sistem harus dapat menyediakan fasilitas *log out* sebagai pasangan dari fasilitas *log in*. *Log out* berfungsi untuk mengakhiri suatu sesi penggunaan aplikasi oleh *member*.	Anggota, *Sweeper*, *Admin*	*Log Out*
F4	Untuk mengurangi anonimitas *member*, sistem dilengkapi dengan kemampuan untuk melihat detail profil, baik profil diri maupun profil pengguna lainnya.	Tamu, Anggota, *Sweeper*, *Admin*	Lihat Profil Member
F5	Sistem harus dapat menyediakan fasilitas ubah profil diri untuk memberikan keterangan lebih lengkap tentang jati diri member.	Anggota, *Sweeper*, *Admin*	Ubah Profil Diri
F6	Sistem harus dapat menyediakan fasilitas ubah kata sandi.	Anggota, *Sweeper*, *Admin*	Ubah Kata Sandi
F7	Sistem harus dapat menyediakan fasilitas untuk mengirimkan pertanyaan yang merupakan inti dari aplikasi. Setelah pertanyaan diterima, sistem akan menyimpan data dalam basis data.	Anggota, *Sweeper*, *Admin*	Kirim Pertanyaan

ID.	Kebutuhan	Aktor	Use Case
F8	Sistem harus dapat menyediakan fasilitas untuk mengirimkan jawaban sebagai tanggapan atas suatu pertanyaan.	Anggota, *Sweeper*, *Admin*	Kirim Jawaban
F9	Sistem harus dapat menyediakan fasilitas untuk mengirimkan pengaduan penyalahgunaan, baik yang berupa pertanyaan maupun jawaban yang dikirimkan oleh *member*.	Anggota, *Sweeper*, *Admin*	Kirim Pengaduan Penyalahgunaan
F10	Sistem harus dapat menyediakan fasilitas pencarian data yang berfokus pada kata kunci yang terdapat dalam pertanyaan yang telah disimpan dalam basis data.	Anggota, *Sweeper*, *Admin*	Cari Pertanyaan
F11	Sistem harus dapat menyediakan fasilitas jelajah kategori pertanyaan untuk melihat seluruh pertanyaan yang masuk ke dalam suatu kategori.	Tamu, Anggota, *Sweeper*, *Admin*	Jelajah Kategori
F12	Sistem harus dapat menyediakan fasilitas penilaian (*rating*) pertanyaan dan jawaban sebagai penanda bahwa seorang *member* menyetujui atau menyukai kiriman *member* lainnya.	Anggota, *Sweeper*, *Admin*	Mengirim Rating
F13	Sistem harus dapat menyediakan fasilitas penambahan *admin*.	*Admin*	Tambah *Admin*
F14	Sistem harus dapat menyediakan fasilitas penambahan *sweeper*.	*Admin*	Tambah *Sweeper*
F15	Sistem harus dapat menyediakan fasilitas untuk menghapus *member* karena tidak tertutup kemungkinan terdapat *member* yang berulang kali melanggar aturan dan membuat *member* lain terganggu dengan *posting* yang dibuatnya.	*Admin*	Hapus Member
F16	Sistem harus dapat menyediakan fasilitas untuk menghapus pertanyaan bilamana pertanyaan tersebut oleh beberapa member dirasa melanggar peraturan, seperti menyinggung isu SARA, dll.	*Sweeper* dan *Admin*	Hapus Pertanyaan
F17	Sistem harus dapat menyediakan fasilitas untuk menghapus jawaban bilamana pertanyaan tersebut oleh beberapa member dirasa melanggar peraturan seperti menyinggung isu SARA.	*Sweeper* dan *Admin*	Hapus Jawaban
N1	Sistem menyediakan fasilitas pemberian skor dengan harapan dapat meningkatkan aktifitas dan kualitas penggunaan situs.	-	-
N2	Sistem dikembangkan dengan menggunakan bahasa pemrograman web PHP, menggunakan sistem manajemen basis data MySQL, dan menggunakan framework CodeIgniter.	-	-
N3	Sistem harus dapat dijalankan pada jaringan komputer dengan protokol TCP/IP.	-	-

Sumber: Analisis Kebutuhan

4.1.3 Diagram *Use Case*

Kebutuhan-kebutuhan fungsional yang diperlukan oleh pengguna dan harus dapat dipenuhi oleh sistem, akan dimodelkan dengan UML menggunakan diagram *use case*. Pada sistem ini terdapat dua puluh tiga *use case* dengan empat aktor yaitu tamu, anggota, *sweeper*, dan *admin*. Gambar 4.1 merupakan diagram *use case* dari sistem yang akan dibangun.



Gambar 4.1 Diagram *Use Case* Knowledge Market

Sumber: Analisis Kebutuhan

Spesifikasi dari masing-masing *use case* yang terdapat pada diagram *use case* pada Gambar 4.1, akan dijabarkan secara mendetail pada deskripsi *use case*. Di dalam deskripsi *use case* akan dijelaskan nama *use case*, nama aktor yang berinteraksi dengan *use case* tersebut, tujuan *use case*, deskripsi global *use case*, pra-kondisi atau kondisi awal yang harus dipenuhi sebelum *use case* dijalankan dan pasca-kondisi atau kondisi akhir yang diharapkan dari *use case* tersebut. Selain itu, juga akan dijabarkan aliran kejadian utama (*main flow of events*) yang menggambarkan tanggapan sistem (*system response*) atas suatu aksi yang diberikan oleh aktor (*actor action*), serta aliran kejadian alternatif (*alternative flow of events*) yang mungkin terjadi jika suatu kondisi tidak bisa terpenuhi (*exception*) dan kebutuhan khusus (*special requirement*) yang harus dipenuhi untuk menjalankan suatu fungsi tertentu.

Kebutuhan fungsional yang pertama adalah fasilitas registrasi. Registrasi bertujuan untuk menjadikan tamu sebagai anggota yang sah dengan mencatat di basis data *nick* pengguna sebagai tanda identifikasi. Tabel 4.3 merupakan deskripsi dari *use case* registrasi.

Tabel 4.3 Deskripsi *Use Case* Registrasi

Use Case	Registrasi
Aktor	Tamu
Tujuan	Memberikan tamu yang ingin menjadi anggota *nick* pengguna yang digunakan untuk masuk ke dalam sistem (*log in*).
Deskripsi	Sistem memiliki dua tipe pengguna, yaitu tamu dan *registered user* (Anggota, *Sweeper*, dan *Admin*). Anggota adalah seorang user yang telah melakukan registrasi dan memiliki nama user sebagai bukti keanggotaan. Proses registrasi akan memberikan akses berupa nama user kepada tamu yang ingin menjadi anggota dengan syarat mengisi form yang kemudian akan divalidasi oleh sistem.
Pra-kondisi	Form registrasi dalam keadaan aktif
Pasca-kondisi	Sistem memberikan notifikasi bahwa proses registrasi telah selesai dan kemudian ditampilkan halaman utama.
Aliran Utama	
Aksi dari Aktor	**Tanggapan dari Sistem**
1. Anggota memasukkan nilai *nick*, kata sandi dan beberapa *form* isian lain pada ruang yang disediakan. Setelah selesai proses diakhiri dengan menekan tombol "Kirim".	2. Sistem melakukan validasi terhadap nilai *nick* pengguna. Jika nilai *nick* belum pernah ada dalam basis data (*unique*), maka semua data disimpan dalam basis data dan kemudian ditampilkan halaman utama (*home page*).
Aliran Alternatif 1: Eksepsi jika ada bagian form yang belum diisi	
	1. Sistem menampilkan pesan peringatan kesalahan bahwa terdapat bidang isian (*field*) yang kosong dan harus dilengkapi.
Aliran Alternatif 2: Eksepsi jika data nick sudah terpakai	
	1. Sistem menampilkan pesan peringatan kesalahan bahwa data *nick* sudah dipakai oleh anggota lainnya.
Aliran Alternatif 3: Eksepsi jika data kata sandi ulang tidak sama dengan data kata sandi ulang	
	1. Sistem menampilkan pesan peringatan kesalahan bahwa data kata sandi ulangan harus diisi data yang sama dengan data kata sandi.

Sumber: Analisis Kebutuhan

Kebutuhan fungsional yang kedua adalah fasilitas *log in*. *Log in* bertujuan untuk melakukan validasi kepada anggota yang sah. Tabel 4.4 merupakan deskripsi dari *use case log in*.

Tabel 4.4 Deskripsi *Use Case Log In*

Use Case	Login
Aktor	*Registered User* (Anggota, *Sweeper*, dan *Admin*)
Tujuan	Melakukan autentikasi terhadap *user*

Deskripsi	Untuk masuk ke dalam sistem, aktor terlebih dahulu harus melakukan proses *log in*. Aktor harus memasukkan data *nick* dan kata sandi yang benar dan diakhiri dengan menekan tombol "Log in".
Pra-kondisi	Form login dalam keadaan aktif
Pasca-kondisi	Sistem menampilkan halaman utama dan status pengguna berubah dari "Tamu" menjadi *nick* pengguna yang bersangkutan

Aliran Utama	
Aksi dari Aktor	**Tanggapan dari Sistem**
1. Aktor memasukkan nilai *nick* dan kata sandi pada ruang yang disediakan, kemudian mengakhiri proses *log in* dengan menekan tombol "Log in".	2. Sistem melakukan validasi terhadap nilai *nick* dan kata sandi yang dimasukkan oleh aktor. Jika data tersebut benar, maka sistem akan membuat satu *session* yang menyimpan data-data pengguna tersebut. Pada bagian atas halaman (*status bar*) terdapat tombol "Log out" untuk keluar dari sistem.
Aliran Alternatif 1: Eksepsi jika ada bagian form yang belum diisi	
	1. Sistem menampilkan pesan peringatan kesalahan bahwa terdapat bidang isian (*field*) yang kosong dan harus dilengkapi.
Aliran Alternatif 2: Eksepsi jika proses validasi pengguna tidak berhasil	
	1. Sistem menampilkan pesan peringatan kesalahan bahwa terdapat kesalahan penulisan data *nick* atau kata sandi yang tidak sesuai dengan data yang tersimpan dalam basis data (*invalid user*).

Sumber: Analisis Kebutuhan

Kebutuhan fungsional *log out* direpresentasikan dengan *use case log out*. Proses *log out* bertujuan untuk mengakhiri sesi yang telah dibuat oleh pengguna yang terdaftar. Tabel 4.5 merupakan deskripsi dari *use case log in*..

Tabel 4.5 Deskripsi *Use Case Log Out*

Use Case	*Log Out*
Aktor	*Registered User*
Tujuan	Mengakhiri suatu sesi yang telah dilakukan oleh *user*
Deskripsi	Aktor melakukan proses log out sebagai penanda akhir dari suatu sesi.
Pra-kondisi	User telah melakukan *log in*
Pasca-kondisi	User berubah status dari Anggota (ditunjukkan dengan nama) menjadi tamu dan ditampilkan notifikasi *log out*

Aliran Utama	
Aksi dari Aktor	**Tanggapan dari Sistem**
1. *User* menekan tombol **"Log out"**	2. Sistem mengakhiri/menghapus session dari aktor tersebut dan menampilkan notifikasi serta merubah status pengguna menjadi tamu.

Sumber: Analisis Kebutuhan

Kebutuhan fungsional lihat profil direpresentasikan dengan *use case* lihat profil. Fasilitas lihat profil bertujuan untuk memberikan informasi tentang detail identitas pengguna situs. Tabel 4.6 merupakan deskripsi dari *use case* lihat profil.

Tabel 4.6 Deskripsi *Use Case* Lihat Profil

Use Case	Lihat Profil
Aktor	Semua Tipe Aktor
Tujuan	Menyediakan informasi tentang detail identitas pengguna situs.
Deskripsi	Identitas pengirim pertanyaan atau jawaban dapat menjadi nilai tambah untuk pengguna situs dalam menilai tingkat kepercayaan kiriman (*posting*).
Pra-kondisi	Halaman hasil pencarian user atau halaman detail pertanyaan sedang aktif.
Pasca-kondisi	Halaman detail user ditampilkan.
Aliran Utama	
Aksi dari Aktor	**Tanggapan dari Sistem**
1. *User* menekan link dari nama user.	2. Sistem menampilkan halaman detail user yang dimaksud.

Sumber: Analisis Kebutuhan

Kebutuhan fungsional untuk mengubah profil diri direpresentasikan oleh *use case* Ubah Profil. Fasilitas ubah profil diri bertujuan untuk mengubah detail identitas yang telah ditentukan sebelumnya. Tabel 4.7 merupakan deskripsi dari *use case* ubah profil diri.

Tabel 4.7 Deskripsi *Use Case* Ubah Profil Diri

Use Case	Ubah Profil Diri
Aktor	*Registered User*
Tujuan	Memberikan fasilitas kepada user untuk melakukan pengubahan detail profil diri.
Deskripsi	Profil merupakan informasi pelengkap yang dapat menambah tingkat kepercayaan atas jawaban yang telah dikirimkan.
Pra-kondisi	Halaman edit profil diri sedang aktif.
Pasca-kondisi	Ditampilkan halaman detail profil diri yang telah diubah.

Aliran Utama	
Aksi dari Aktor	**Tanggapan dari Sistem**
1. *User* mengisi form dengan data baru dan menekan tombol simpan.	2. Sistem menyimpan data baru dan kemudian menampilkan halaman detail profil yang telah diubah.
Aliran Alternatif 1: User melakukan pembatalan	
1. *User* menekan tombol *batal*.	2. Sistem kembali menampilkan halaman detail profil.

Sumber: Analisis Kebutuhan

Kebutuhan fungsional pencarian anggota direpresentasikan dengan *use case* Cari Anggota. Fasilitas Cari Anggota bertujuan untuk mencari anggota berdasarkan nick yang digunakan. Tabel 4.8 merupakan deskripsi dari *use case* Cari Anggota.

Tabel 4.8 Deskripsi *Use Case* Cari Anggota

Use Case	Cari Anggota
Aktor	Semua tipe aktor
Tujuan	Menyediakan fasilitas pencarian anggota berdasarkan nama user.
Deskripsi	Adakalanya seorang user ingin mengetahui detail profil dari user lainnya, oleh karena itu sistem menyediakan fasilitas pencarian ini.
Pra-kondisi	Halaman pencarian sedang aktif.
Pasca-kondisi	Sistem menampilkan halaman hasil pencarian.
Aliran Utama	
Aksi dari Aktor	**Tanggapan dari Sistem**
1. *User* mengisi nama user yang hendak dicari dan kemudian menekan tombol **"Cari"**	2. Sistem menampilkan halaman yang berisi hasil pencarian dalam basis data.

Sumber: Analisis Kebutuhan

Kebutuhan fungsional kirim pertanyaan direpresentasikan dengan *use case* Kirim Pertanyaan. Fasilitas Kirim Pertanyaan bertujuan untuk menampung pertanyaan dari pengguna. Tabel 4.9 merupakan deskripsi dari *use case* Kirim Pertanyaan.

Tabel 4.9 Deskripsi *Use Case* Kirim Pertanyaan

Use Case	Kirim Petanyaan
Aktor	*Registered User*
Tujuan	Menampung pertanyaan dari user yang nantinya akan menjadi pemicu kiriman jawaban
Deskripsi	Pertanyaan merupakan inti dari aplikasi ini, oleh karena itu use case ini merupakan use case yang penting. Aplikasi mula-mula menerima data, kemudian menyimpannya ke dalam basis data.
Pra-kondisi	User telah melakukan log in, halaman kirim pertanyaan aktif.
Pasca-kondisi	Muncul notifikasi pengiriman dan redirect ke tampilan tanya.

Aliran Utama	
Aksi dari Aktor	**Tanggapan dari Sistem**
1. *User* mengisi lengkap form dan menekan tombol simpan/kirim.	2. User melakukan validasi, jika valid maka data disimpan ke dalam basis data, kemudian sistem menampilkan halaman detail pertanyaan.
Aliran Alternatif 1: Terdapat form yang tidak diisi	
	1. Sistem menampilkan notifikasi dan menampilkan halaman form kembali.

Sumber: Analisis Kebutuhan

Kebutuhan fungsional kirim jawaban direpresentasikan dengan *use case* Kirim Jawaban. Fasilitas Kirim Jawaban bertujuan untuk menampung jawaban dari pengguna. Tabel 4.10 merupakan deskripsi dari *use case* Kirim Jawaban.

Tabel 4.10 Deskripsi *Use Case* Kirim Jawaban

Use Case	Kirim Jawaban	
Aktor	*Registered User*	
Tujuan	Menampung jawaban sebagai tanggpan dari user atas dikirimnya suatu pertanyaan.	
Deskripsi	Sistem akan mengumpulkan jawaban-jawaban dari berbagai user untuk kemudian dinilai oleh user lain juga dipilih sebagai jawaban terbaik oleh pengirim jawaban.	
Pra-kondisi	Halaman detail pertanyaan aktif.	
Pasca-kondisi	Sistem menampilkan notifikasi penyimpanan jawaban dan redirect ke halaman detail pertanyaan.	
Aliran Utama		
Aksi dari Aktor		**Tanggapan dari Sistem**
1. User mengisi form jawaban yang terdapat dalam halaman detail pertanyaan kemudian menekan tombol kirim.		2. Sistem melakukan validasi terhadap jawaban, dan jika data tidak kosong maka jawaban disimpan ke dalam basis data.
Aliran Alternatif 1: Form jawaban kosong		
1. User menekan tombol kirim namun form jawaban tidak diisi.		2. Sistem memberikan notifikasi error dan redirect ke halaman detail pertanyaan.

Sumber: Analisis Kebutuhan

Kebutuhan fungsional kirim pengaduan direpresentasikan dengan *use case* Kirim Pengaduan. Fasilitas Kirim Pengaduan bertujuan untuk menampung keluhan penyalahgunaan pengisian data pertanyaan atau jawaban dari pengguna. Tabel 4.11 merupakan deskripsi dari *use case* Kirim Pengaduan.

Tabel 4.11 Deskripsi *Use Case* Kirim Pengaduan

Use Case	Kirim Pengaduan
Aktor	*Registered User*

Tujuan	Sistem mengumpulkan data penyalahgunaan dari user atas pertanyaan atau jawaban yang tidak etis atau tidak bermakna (spam) untuk selanjutnya ditanggapi *sweeper* dengan menghapusnya.
Deskripsi	Sistem menyediakan fungsi untuk meningkatkan kualitas jawaban dari user dengan cara membuat fasilitas pengaduan penyalahgunaan.
Pra-kondisi	Halaman kirim pengaduan aktif.
Pasca-kondisi	Notifikasi keberhasilan pengaduan ditampilkan kemudian redirect ke halaman detail pertanyaan.

Aliran Utama	
Aksi dari Aktor	**Tanggapan dari Sistem**
1. *User* mengisi jenis pengaduan yang sesuai atau mengisi pengaduan jenis lain, kemudian menekan tombol kirim.	2. Sistem menyimpan data pengaduan ke dalam basis data, memberikan notifikasi keberhasilan dan melakukan redirect ke halaman detail pertanyaan.
Aliran Alternatif 1: User melakukan pembatalan	
1. *User* menekan tombol batal.	2. Menutup form kirim pengaduan dan kembali ke halaman detail pertanyaan.

Sumber: Analisis Kebutuhan

Kebutuhan fungsional pencarian pertanyaan direpresentasikan dengan *use case* Cari Pertanyaan. Fasilitas Cari Pertanyaan bertujuan untuk mencari pertanyaan yang telah disimpan dalam basis data. Tabel 4.12 merupakan deskripsi dari *use case* Cari Pertanyaan.

Tabel 4.12 Deskripsi *Use Case* Cari Pertanyaan

Use Case	Cari Pertanyaan
Aktor	Semua tipe aktor
Tujuan	Memberikan fasilitas untuk mencari pertanyaan berdasarkan kata kunci tertentu
Deskripsi	Ada kalanya user tertarik dengan suatu kata kunci dan sebelum melakukan posting petanyaan, sangat dianjurkan untuk mencari posting serupa dengan menggunakan fasilitas pencarian pertanyaan ini.
Pra-kondisi	Halaman pencarian posting sedang aktif.
Pasca-kondisi	Sistem menampilkan halaman hasil pencarian.

Aliran Utama	
Aksi dari Aktor	**Tanggapan dari Sistem**
1. *User* mengisi kata kunci pada form dan kemudian menekan tombol **"Cari"**	2. Sistem menampilkan halaman dengan hasil pencarian dalam basis data.
Aliran Alternatif 1: Form kosong	
1. *User* menekan tombol Cari namun form dalam keadaaan kosong.	2. Sistem menampilkan notifikasi kesalahan.

Sumber: Analisis Kebutuhan

Kebutuhan fungsional penjelajahan (*browsing*) kategori direpresentasikan dengan *use case* Jelajah Kategori. Fasilitas Jelajah Kategori bertujuan untuk menampilkan daftar pertanyaan dengan kategori tertentu. Tabel 4.13 merupakan deskripsi dari *use case* Jelajah Kategori.

Tabel 4.13 Deskripsi *Use Case* Jelajah Kategori

Use Case	Jelajah Kategori	
Aktor	Semua tipe aktor	
Tujuan	Memberikan daftar kategori yang berisi berbagai posting yang sejenis	
Deskripsi	Untuk memudahkan penggunaan, situs mengelompokkan posting berdasarkan kategori bidang tertentu.	
Pra-kondisi	Halaman utama aktif.	
Pasca-kondisi	Halaman jelajah kategori ditampilkan.	
Aliran Utama		
Aksi dari Aktor		**Tanggapan dari Sistem**
1. *User* menekan tombol **"Jelajah"**		2. Sistem menampilkan halaman jelajah kategori yang berisi daftar kategori posting.

Sumber: Analisis Kebutuhan

Kebutuhan fungsional menilai (*rate*) jawaban direpresentasikan dengan *use case* Nilai Jawaban. Fasilitas Nilai Jawaban bertujuan untuk menilai apakah suatu jawaban relevan atau tidak terhadap suatu pertanyaan. Tabel 4.14 merupakan deskripsi dari *use case* Nilai Jawaban.

Tabel 4.14 Deskripsi *Use Case* Nilai Jawaban

Use Case	Menilai Jawaban
Aktor	*Registered User*
Tujuan	Memberikan fasilitas kepada user untuk melakukan penilaian terhadap suatu jawaban.
Deskripsi	Dengan fasilitas ini user lain yang bukan pengirim pertanyaan dapat ikut melakukan penilaian terhadap suatu jawaban yang telah dikirmkan.
Pra-kondisi	Halaman detail pertanyaan aktif.
Pasca-kondisi	Halaman detail pertanyaan ditampilkan kembali.

Aliran Utama	
Aksi dari Aktor	**Tanggapan dari Sistem**
1. *User* menekan salah satu dari tombol **OK** atau **Tidak OK.**	2. Sistem menyimpan penialaian ke dalam basis data, dan kemudian menampilkan kembali halaman detail pertanyaan.
Aliran Alternatif 1: User sudah melakukan penilaian sebelumnya	
	1. Sistem menampilkan notifikasi kesalahan dan menampilkan kembali halaman kesalahan.

Sumber: Analisis Kebutuhan

Kebutuhan fungsional tambah *admin* direpresentasikan dengan *use case* Tambah *Admin*. Fasilitas Tambah *Admin* bertujuan untuk menambah pengguna dengan hak akses khusus yaitu *admin* dan sweeper. Tabel 4.15 merupakan deskripsi dari *use case* Tambah *Admin*.

Tabel 4.15 Deskripsi *Use Case* Tambah *Admin*

Use Case	Tambah *Admin*
Aktor	*Admin*
Tujuan	Menambah user dengan hak akses sebagai *admin*
Deskripsi	*Admin* adalah tipe aktor dengan hak akses paling tinggi sehingga diperlukan prosedur khusus untuk melakukan penambahan *admin*.
Pra-kondisi	Halaman tambah *admin* aktif.
Pasca-kondisi	Sistem menampilkan notifikasi bahwa proses penamb ahan *admin* berhasil dilakukan.
Aliran Utama	
Aksi dari Aktor	**Tanggapan dari Sistem**
1. *Admin* mengisi form dengan lengkap dan kemudian menekan tombol Simpan.	2. Sistem menyimpan data ke dalam basis data kemudian menampilkan notifikasi keberhasilan (halaman utama *admin*).
Aliran Alternatif 1: Terdapat form kosong	
	1. Sistem menampilkan notifikasi kesalahan.

Sumber: Analisis Kebutuhan

Kebutuhan fungsional hapus anggota direpresentasikan dengan *use case* Hapus Anggota. Fasilitas Hapus Anggota bertujuan untuk menghap us data anggota karena banyaknya pengaduan dari user lainnya. Tabel 4.16 merupakan deskripsi dari *use case* Hapus Anggota.

Tabel 4.16 Deskripsi *Use Case* Hapus Anggota

Use Case	Hapus Anggota
Aktor	*Admin*
Tujuan	Menghapus anggota dari basis data
Deskripsi	Sistem berusaha menanggulangi adanya anggota "nakal" yang sering melanggar peraturan seperti melakukan *spamming* dengan menyediakan fasilitas untuk menghapus member ini.
Pra-kondisi	Halaman detail pengguna sedang aktif
Pasca-kondisi	Tampil notifikasi keberhasilan proses penghapusan
Aliran Utama	
Aksi dari Aktor	**Tanggapan dari Sistem**
1. *Admin* membuka detail anggota kemudian klik pada link "Hapus Anggota ini".	2. Sistem melakukan penghapusan data anggota tersebut dari basis data.

Sumber: Analisis Kebutuhan

Kebutuhan fungsional hapus pertanyaan direpresentasikan dengan *use case* Hapus Pertanyaan. Fasilitas Hapus Pertanyaan bertujuan untuk menghapus data pertanyaan karena banyaknya pengaduan dari user. Tabel 4.17 merupakan deskripsi dari *use case* Hapus Pertanyaan.

Tabel 4.17 Deskripsi *Use Case* Hapus Pertanyaan

Use Case	Hapus Pertanyaan
Aktor	*Sweeper*, *Admin*
Tujuan	Menghapus suatu pertanyaan dari basis data
Deskripsi	Sistem menyediakan fasilitas untuk menghapus pertanyaan bilamana terdapat pertanyaan yang melanggar aturan yang ada.
Pra-kondisi	Halaman detail pertanyaan sedang aktif
Pasca-kondisi	Tampil notifikasi keberhasilan proses penghapusan
Aliran Utama	
Aksi dari Aktor	**Tanggapan dari Sistem**
1. *Admin* membuka detail pertanyaan kemudian klik pada link "Hapus Pertanyaan ini".	2. Sistem melakukan penghapusan data pertanyaan tersebut dari basis data juga data jawaban yang terkait dengan pertanyaan tersebut.
Aliran Alternatif 1: User keluar dari sistem	
1. *User* menekan tombol *exit* pada *browser*.	2. Menutup form login dan keluar dari sistem.

Sumber: Analisis Kebutuhan

Kebutuhan fungsional hapus jawaban direpresentasikan dengan *use case* Hapus Jawaban. Fasilitas Hapus Jawaban bertujuan untuk menghapus data jawaban karena banyaknya pengaduan dari user. Tabel 4.18 merupakan deskripsi dari *use case* Hapus Jawaban.

Tabel 4.18 Deskripsi *Use Case* Hapus Jawaban

Use Case	Hapus Jawaban
Aktor	*Sweeper*, atau *Admin*
Tujuan	Melakukan penghapusan jawaban dari basis data
Deskripsi	Sistem menyediakan fasilitas penghapusan jawaban bilamana d irasa terdapat suatu kiriman jawaban tidak pantas/etis.
Pra-kondisi	Halaman detail pertanyaan sedang aktif
Pasca-kondisi	Tampil notifikasi keberhasilan proses penghapusan
Aliran Utama	
Aksi dari Aktor	**Tanggapan dari Sistem**
1. *Admin* membuka detail pertanyaan kemudian klik pada link "Hapus Jawaban ini".	2. Sistem melakukan penghapusan data jawaban tersebut dari basis.

Sumber: Analisis Kebutuhan

4.2 Perancangan Perangkat Lunak

Pada tahap perancangan perangkat lunak akan dilakukan proses perancangan berdasarkan pada hasil analisis kebutuhan. Pemodelan perancangan menggunakan diagram klas untuk memodelkan elemen-elemen klas dan relasi antar klas serta diagram sekuensial untuk menggambarkan aliran proses atau data antar klas berdasarkan urutan waktu.

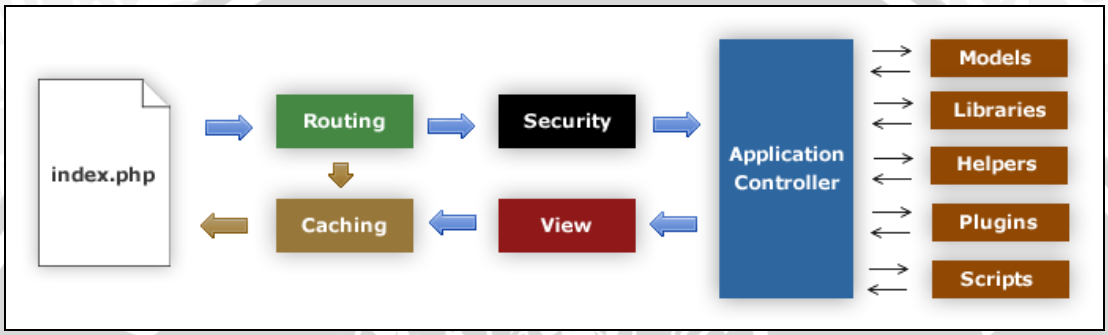
4.2.1 Arsitektur Perangkat Lunak

Arsitektur perangkat lunak sistem dirancang untuk dapat dimodifikasi menggunakan kerangka kerja (*framework*). *Framework* yang digunakan pada aplikasi ini adalah PHP *framework codeigniter* yang menerapkan pola (*pattern*) MVC (*Model, View, Controller*). MVC itu sendiri adalah konsep pemrograman yang memisahkan pemrograman *logic* aplikasi dengan presentasinya. MVC memungkinkan halaman web berisi sedikit sekali skrip PHP karena file presentasi terpisah dengan file skrip PHP.

- *Model* mewakili struktur data. Umumnya kelas model akan berisi fungsi - fungsi yang akan membantu untuk mengambil (*get*), menambah (*insert*), dan mengedit (*update*) data di *database*.

- *View* adalah informasi yang disajikan ke pengguna (*user*). Sebuah *view* normalnya adalah sebuah halaman web, namun di *CodeIgniter* sebuah *view* bisa juga menjadi bagian halaman seperti *header* atau *footer*. Bisa juga menjadi berbagai jenis halaman yang lain.
- *Controller* berfungsi sebagai penengah antara *Model, View* dan *resource* lain yang dibutuhkan untuk memroses *HTTP request* dan memproduksi sebuah halaman web.

Alur program MVC yang diterapkan pada PHP *Framework CodeIgniter* ditunjukkan pada gambar 4.2.



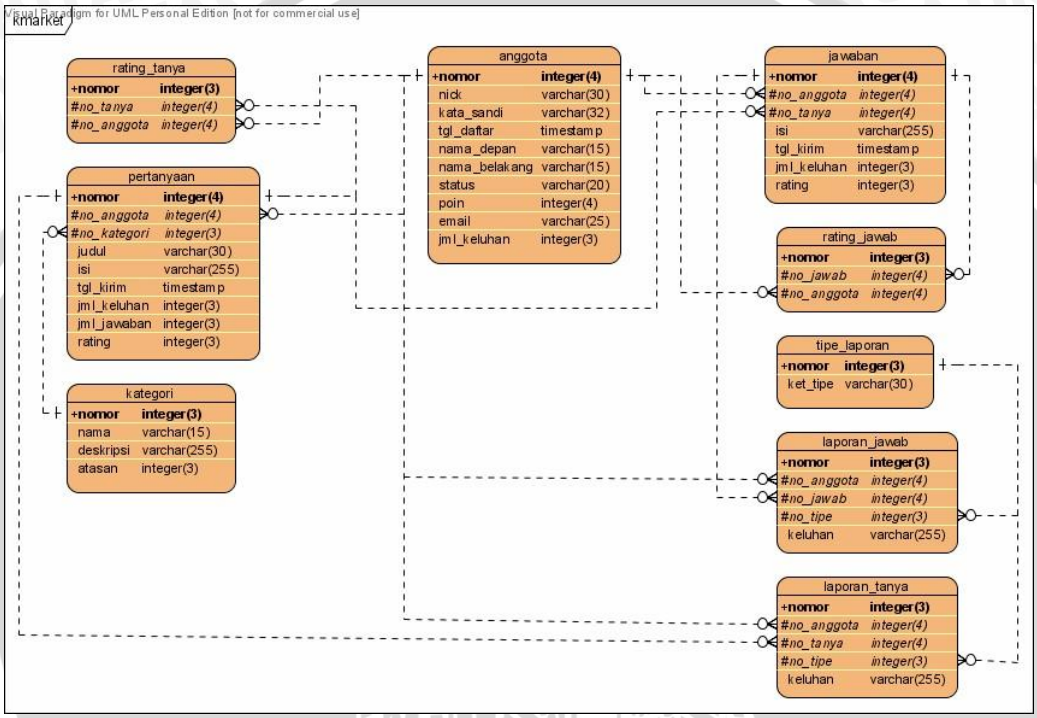
Gambar 4.2 Alur Program PHP Framework CodeIgniter

Sumber: CodeIgniter Help File

1. Index.php berfungsi sebagai *controller* depan, menginisialisasi *basic resource* yang dibutuhkah untuk menjalankan CodeIgniter.
2. *Router* menganalisa HTTP *request* untuk menentukan apa yang harus dilakukan dengan HTTP *request* itu.
3. Jika *file cache* masih ada, maka akan dikirim langsung ke *browser*, tanpa melewati eksekusi normal sistem.
4. *Security*, sebelum *controller* aplikasi di panggil, HTTP *request* dan data yang dikirim pengguna disaring terlebih dahulu untuk alasan keamanan.
5. *Controller* memanggil *model*, pustaka (*library*) inti, *plugin*, *helper*, dan *resource* lainnya yang di butuhkan untuk memroses permintaan (*request*) tertentu.
6. *View* yang sudah diproses, dikirim ke *browse*r sebagai hasil yang terlihat. Jika status *caching* hidup, *view* akan disimpan di *cache*, jadi jika ada permintaan *request* yang sama, *view* itu bisa ditampilkan lagi.

4.2.2 Perancangan Basis Data (*Database*)

Untuk menggambarkan rancangan basis digunakan diagram ERD (*Entity Relationship Diagram*). Aplikasi ini melibatkan beberapa entitas di antaranya adalah anggota, pertanyaan, jawaban, kategori, laporan tanya, laporan jawab, tipe laporan, rating tanya, dan rating jawab). Gambar 4.3 merupakan ERD untuk aplikasi *internet based knowledge market.*



Gambar 4.3 ERD Sistem Internet Based Knowledge Market

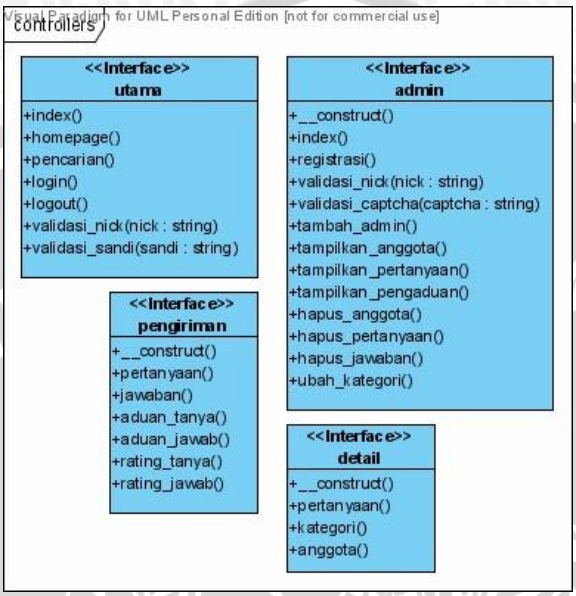
Sumber: Perancangan

4.2.3 Diagram Klas (*Class Diagram*)

Diagram klas memberikan gambaran elemen-elemen klas yang membentuk sebuah perangkat lunak. Klas akan merealisasikan fungsionalitas yang harus dipenuhi oleh setiap *use case*. Klas dibangun dengan menganalisis *use case* yang telah dimodelkan dan kemudian merealisasikannya (*Use case Realization*). Diagram klas menggambarkan nama, attribut dan metode yang dimiliki oleh sebuah klas yang dirancang sesuai dengan fungsi dan tanggung jawab dari setiap klas.

4.2.3.1 Diagram Klas Controller

Diagram Klas pada bagian controller terdapat empat klas controller diantaranya klas kontroler_utama, kontroler_admin, kontroler_user, dan kontroler_detail. Semua klas ini merupakan turunan dari klas controller yang terdapat pada *framework CodeIgniter*.



Gambar 4.4 Diagram Klas pada Bagian Controller

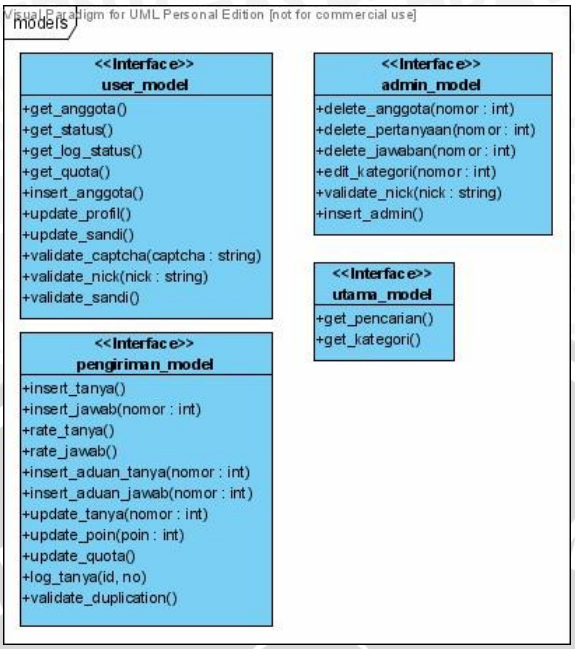
Sumber: Perancangan

4.2.3.2. Diagram Klas Model

Diagram Klas pada bagian model terdapat empat klas model diantaranya klas utama_model, admin_model, user_model, dan pengiriman_model. Semua klas ini merupakan turunan dari klas model yang terdapat pada *framework CodeIgniter*.

4.2.4 Diagram Sekuensial (*Sequential Diagram*)

Diagram sekuensial menggambarkan urutan kejadian atau proses yang berlangsung ketika menjalankan sebuah fungsionalitas sistem dan pola hubungan antar objek.



Gambar 4.5 Diagram Klas pada Bagian Model

Sumber: Perancangan

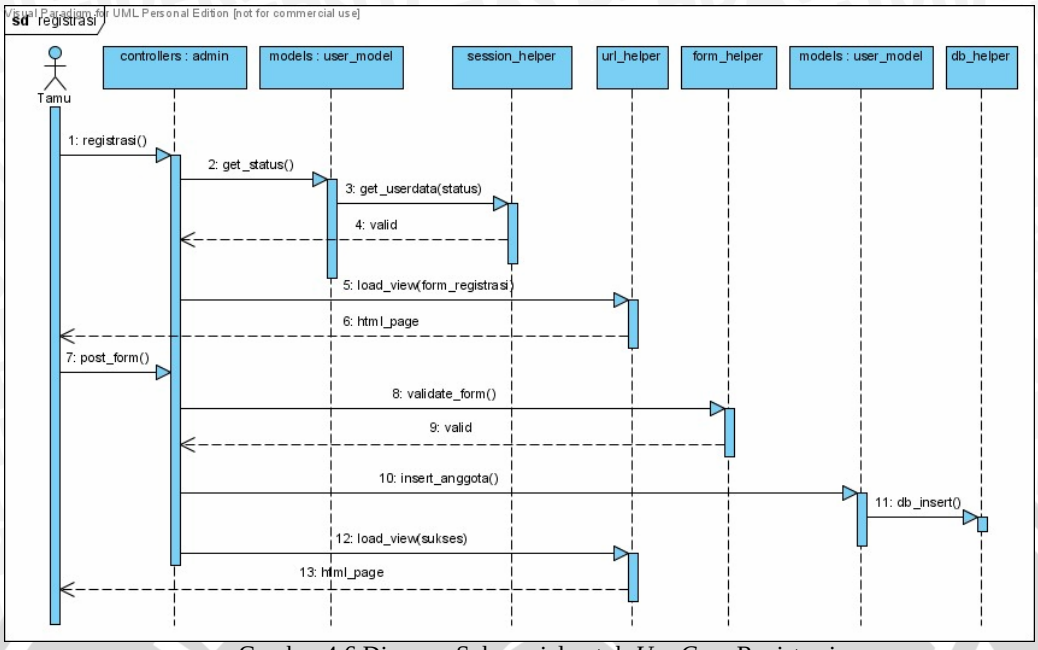
Secara umum, proses-proses yang digambarkan pada diagram sekuensial memanfaatkan helper yang dimiliki *framework* Codeigniter. *Helper* yang sering digunakan di antaranya adalah *session helper*, *url helper*, dan *database helper*.

4.2.4.1 Diagram Sekuensial Registrasi

Diagram sekuensial ini menggambarkan interaksi yang terjadi ketika seorang pengguna melakukan registrasi. Diagram sekuensial registrasi ditunjukkan pada gambar 4.6.

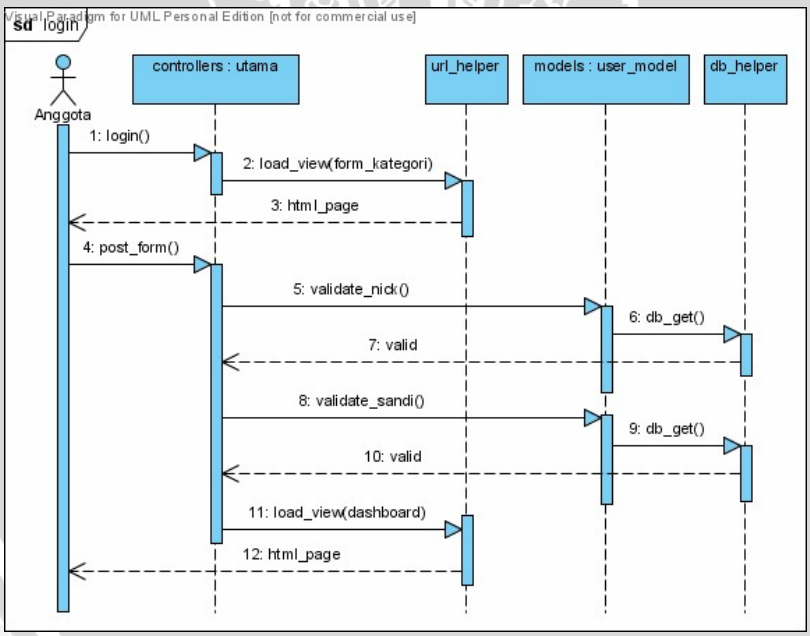
4.2.4.2 Diagram Sekuensial Login

Diagram sekuensial ini menggambarkan interaksi yang terjadi ketika seorang user melakukan login. Diagram sekuensial login ditunjukkan pada gambar 4.7.



Gambar 4.6 Diagram Sekuensial untuk *Use Case* Registrasi

Sumber: Perancangan

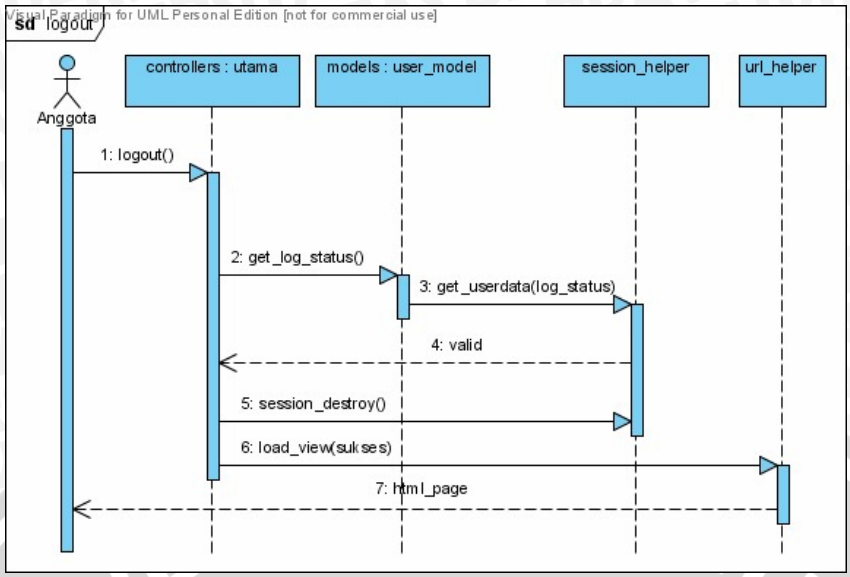


Gambar 4.7 Diagram Sekuensial untuk *Use Case* Login

Sumber: Perancangan

4.2.4.3 Diagram Sekuensial Logout

Diagram sekuensial ini menggambarkan interaksi yang terjadi ketika seorang user melakukan logout. Diagram sekuensial logout ditunjukkan pada gambar 4.8.

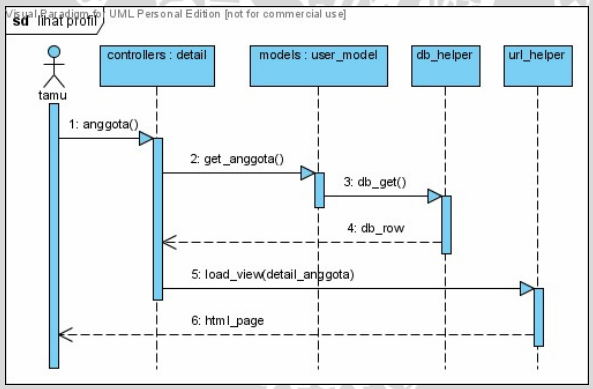


Gambar 4.8 Diagram Sekuensial untuk *Use Case* Logout

Sumber: Perancangan

4.2.4.4 Diagram Sekuensial Lihat Profil Anggota

Diagram sekuensial ini menggambarkan interaksi yang terjadi ketika seorang user melakukan aksi lihat profil anggota. Diagram sekuensial lihat profil anggota ditunjukkan pada gambar 4.9.

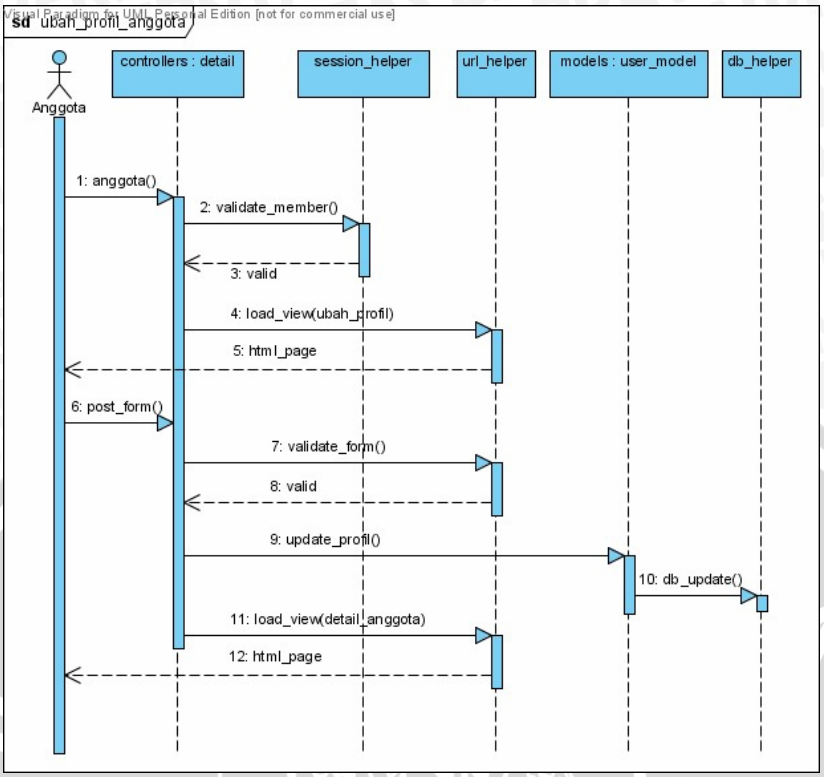


Gambar 4.9 Diagram Sekuensial untuk *Use Case* Lihat Profil Anggota

Sumber: Perancangan

4.2.4.5 Diagram Sekuensial Ubah Detail Profil

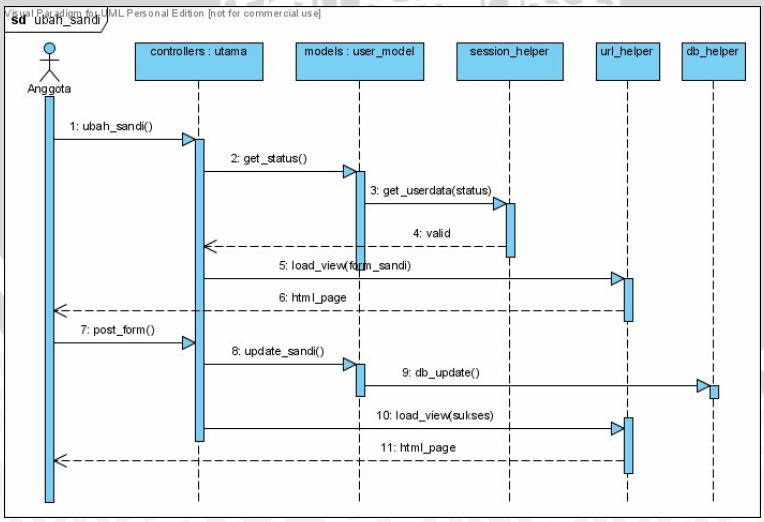
Diagram sekuensial ini menggambarkan interaksi yang terj adi ketika seorang user melakukan pengubahan detail profil diri. Diagram sekuensial lihat profil diri ditunjukkan pada gambar 4.10.



Gambar 4.10 Diagram Sekuensial untuk *Use Case* Ubah Detail Profil. Sumber: Perancangan

4.2.4.6 Diagram Sekuensial Ubah Kata Sandi

Diagram sekuensial ini menggambarkan interaksi yang terjadi ketika seorang user melakukan pengubahan kata sandi. Diagram sekuensial ubah kata sandi ditunjukkan pada gambar 4.11.

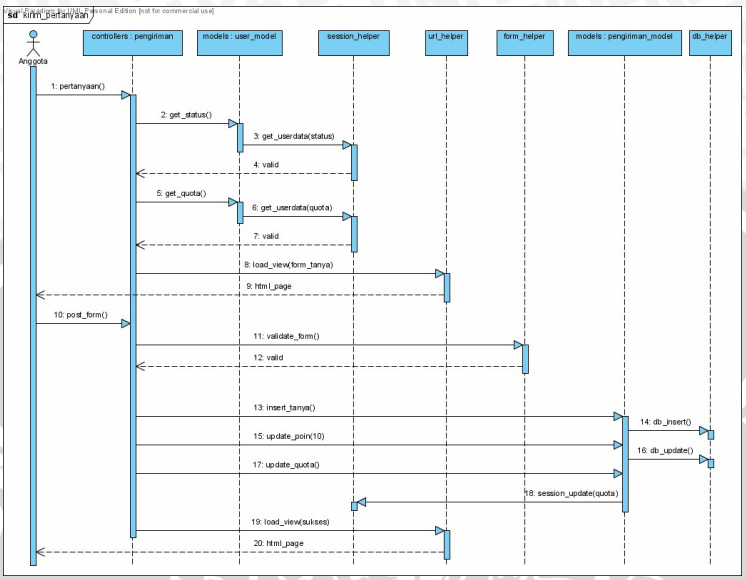


Gambar 4.11 Diagram Sekuensial untuk *Use Case* Ubah Sandi

Sumber: Perancangan

4.2.4.7 Diagram Sekuensial Kirim Pertanyaan

Diagram sekuensial ini menggambarkan interaksi yang terjadi ketika seorang user melakukan aksi kirim pertanyaan. Diagram sekuensial kirim pertanyaan ditunjukkan pada gambar 4.12.

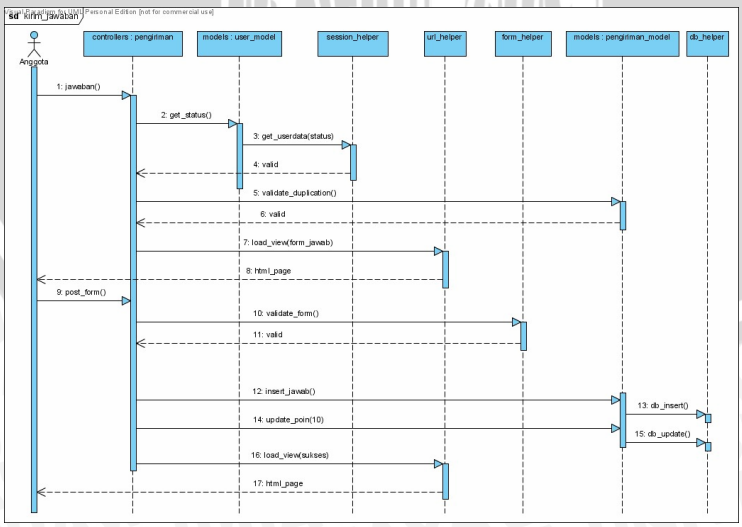


Gambar 4.12 Diagram Sekuensial untuk *Use Case* Kirim Pertanyaan

Sumber: Perancangan

4.2.4.8 Diagram Sekuensial Kirim Jawaban

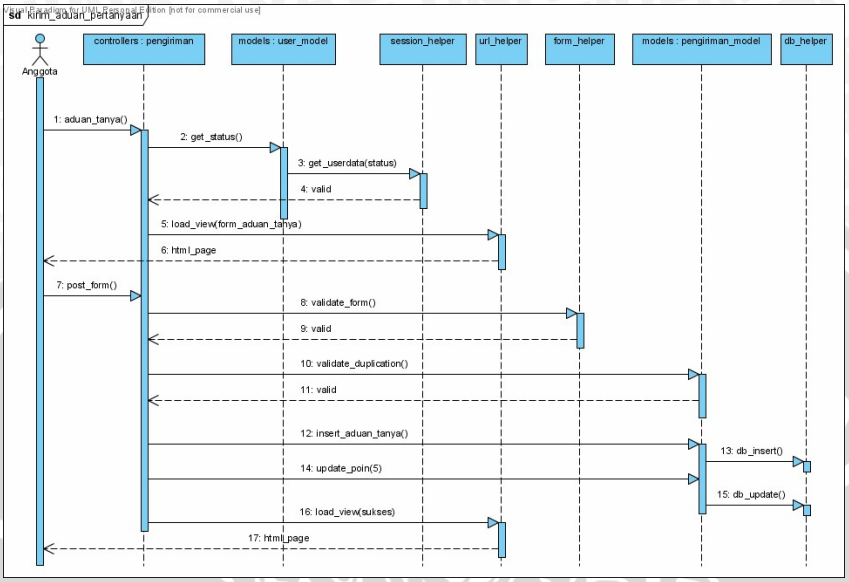
Diagram sekuensial ini menggambarkan interaksi yang terjadi ketika seorang user melakukan aksi kirim jawaban. Diagram sekuensial kirim jawaban ditunjukkan pada gambar 4.13.



Gambar 4.13 Diagram Sekuensial untuk *Use Case* Kirim Jawaban. Sumber: Perancangan

4.2.4.9 Diagram Sekuensial Kirim Aduan Pertanyaan

Diagram sekuensial ini menggambarkan interaksi yang terjadi ketika seorang user melakukan aksi kirim aduan penyalahgunaan di bagian pertanyaan. Diagram sekuensial kirim aduan pertanyaan ditunjukkan pada gambar 4.14.

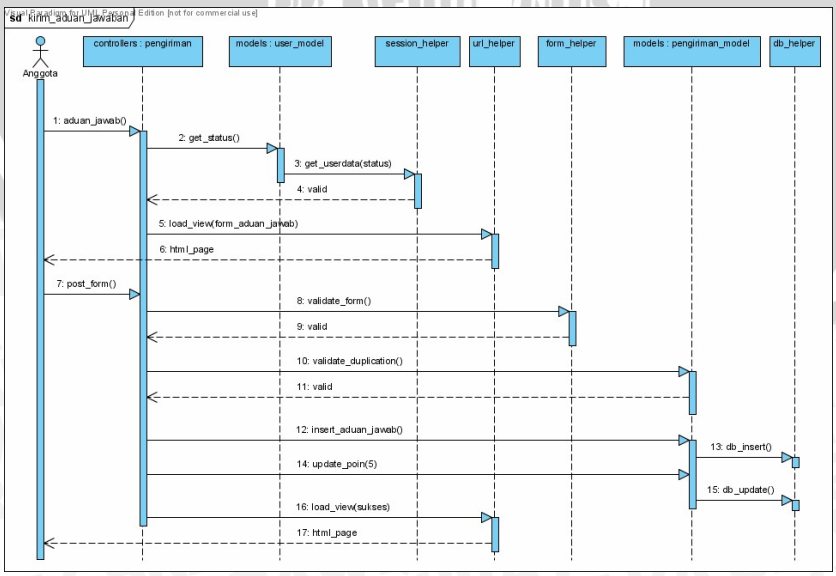


Gambar 4.14 Diagram Sekuensial untuk *Use Case* Kirim Aduan Pertanyaan

Sumber: Perancangan

4.2.4.10 Diagram Sekuensial Kirim Aduan Jawaban

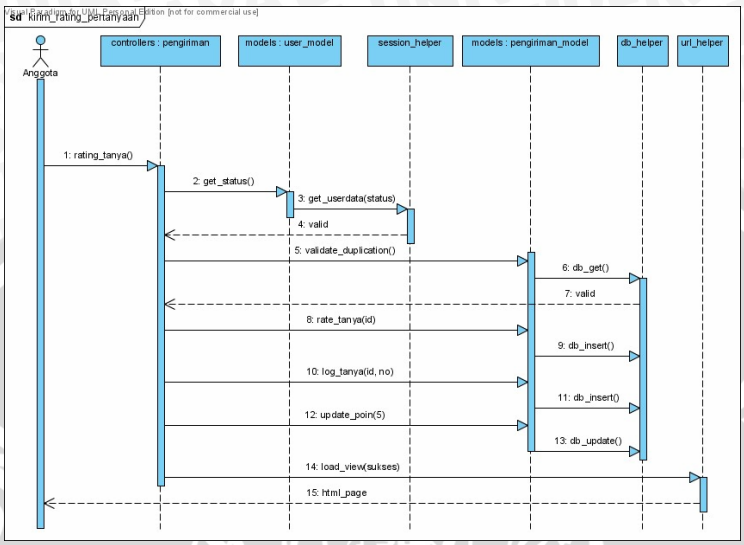
Diagram sekuensial ini menggambarkan interaksi yang terjadi ketika seorang user melakukan aksi kirim aduan penyalahgunaan di bagian jawaban. Diagram sekuensial kirim aduan jawaban ditunjukkan pada gambar 4.15.



Gambar 4.15 Diagram Sekuensial untuk *Use Case* Kirim Aduan Jawaban. Sumber: Perancangan

4.2.4.11 Diagram Sekuensial Kirim Rating Pertanyaan

Diagram sekuensial ini menggambarkan interaksi yang terjadi ketika seorang user melakukan aksi kirim rating pertanyaan. Diagram sekuensial kirim rating pertanyaan ditunjukkan pada gambar 4.16.

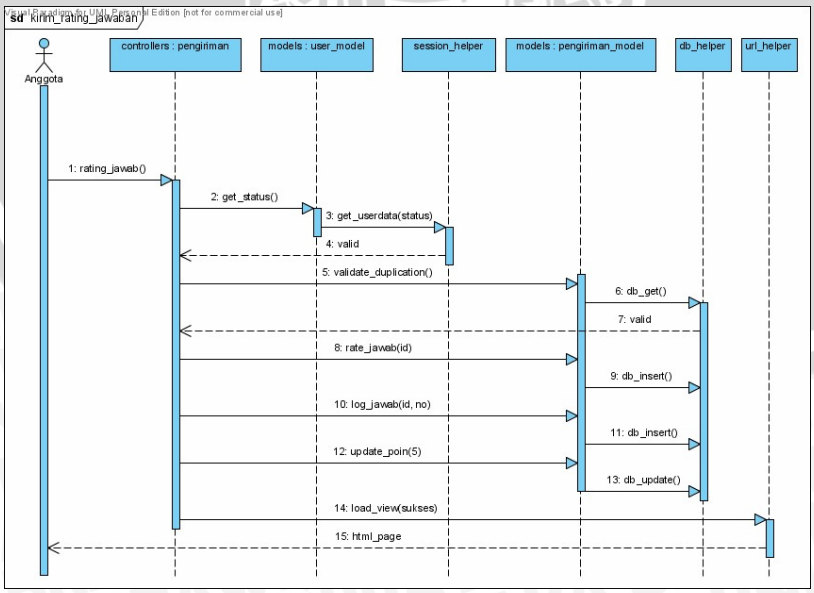


Gambar 4.16 Diagram Sekuensial untuk *Use Case* Kirim rating Pertanyaan

Sumber: Perancangan

4.2.4.12 Diagram Sekuensial Kirim Rating Jawaban

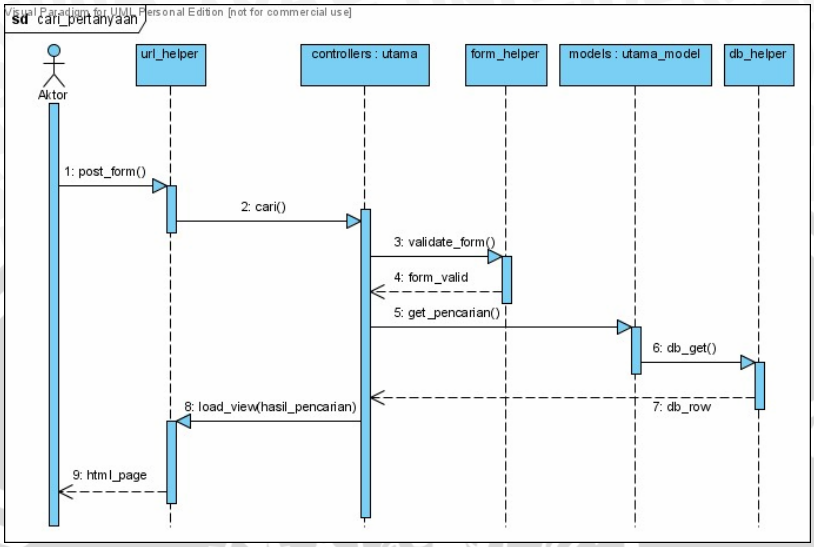
Diagram sekuensial ini menggambarkan interaksi yang terjadi ketika seorang user melakukan aksi kirim rating jawaban. Diagram sekuensial kirim rating jawaban ditunjukkan pada gambar 4.17.



Gambar 4.17 Diagram Sekuensial untuk *Use Case* Kirm Rating Jawaban. Sumber: Perancangan.

4.2.4.12 Diagram Sekuensial Cari Pertanyaan

Diagram sekuensial ini menggambarkan interaksi yang terjadi ketika seorang user melakukan aksi cari pertanyaan. Diagram sekuensial cari pertanyaan ditunjukkan pada gambar 4.18.

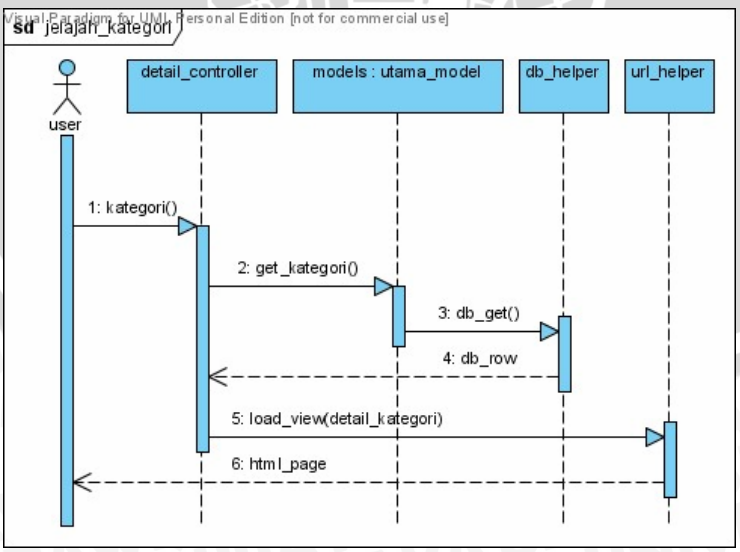


Gambar 4.18 Diagram Sekuensial untuk *Use Case* Cari Pertanyaan

Sumber: Perancangan

4.2.4.13 Diagram Sekuensial Jelajah Kategori

Diagram sekuensial ini menggambarkan interaksi yang terjadi ketika seorang user melakukan aksi jelajah kategori. Diagram sekuensial jelajah kategori ditunjukkan pada gambar 4.19.

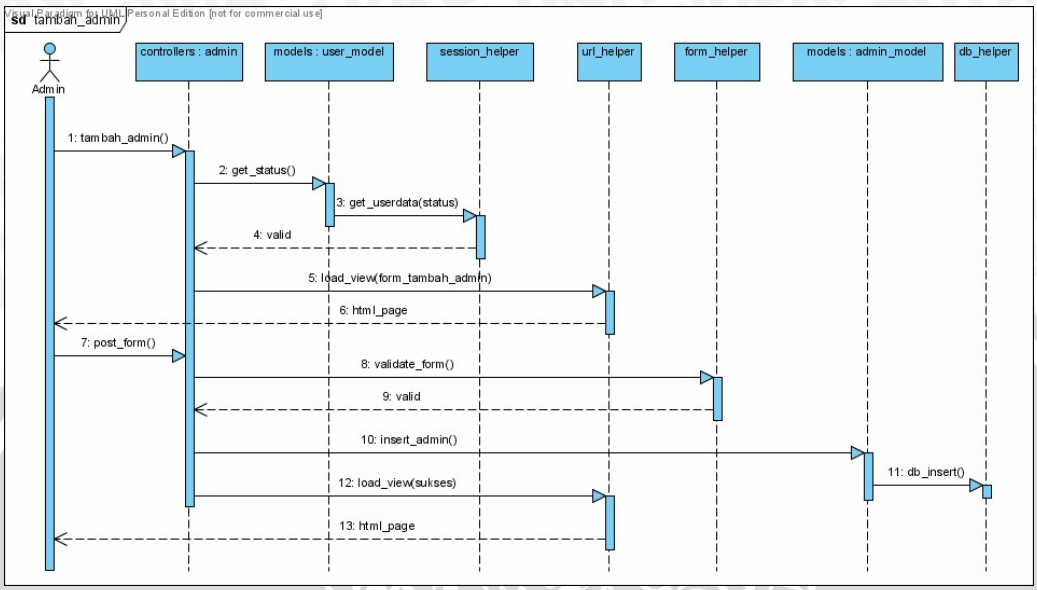


Gambar 4.19 Diagram Sekuensial untuk *Use Case* Jelajah Kategori

Sumber: Perancangan

4.2.4.14 Diagram Sekuensial Tambah Admin

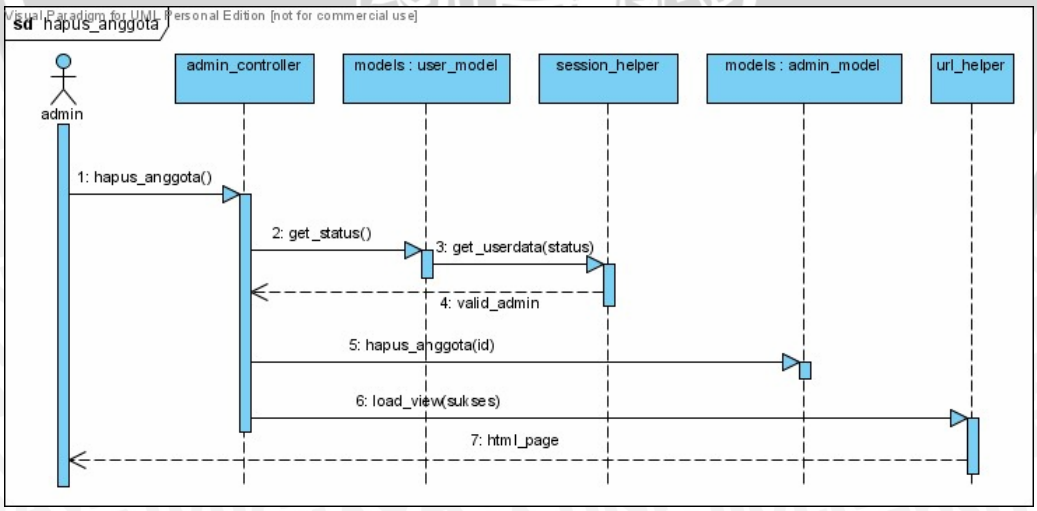
Diagram sekuensial ini menggambarkan interaksi yang terjadi ketika seorang user melakukan aksi tambah admin. Diagram sekuensial tambah admin ditunjukkan pada gambar 4.20.



Gambar 4.20 Diagram Sekuensial untuk *Use Case* Tambah Admin. Sumber: Perancangan

4.2.4.15 Diagram Sekuensial Hapus Anggota

Diagram sekuensial ini menggambarkan interaksi yang terjadi ketika seorang user melakukan aksi hapus anggota. Diagram sekuensial hapus anggota ditunjukkan pada gambar 4.21.

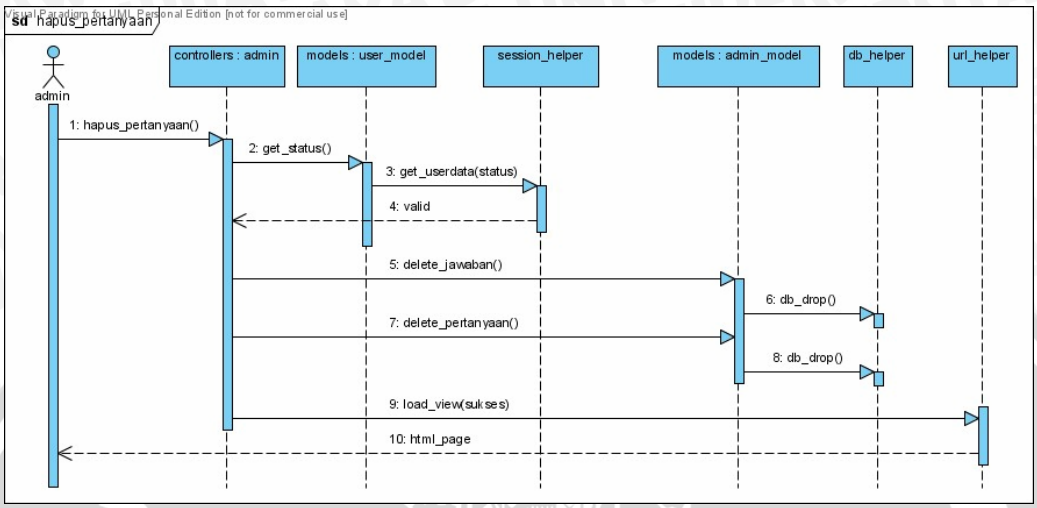


Gambar 4.21 Diagram Sekuensial untuk *Use Case* Hapus Anggota

Sumber: Perancangan

4.2.4.16 Diagram Sekuensial Hapus Pertanyaan

Diagram sekuensial ini menggambarkan interaksi yang terjadi ketika seorang user melakukan aksi hapus pertanyaan. Diagram sekuensial hapus pertanyaan ditunjukkan pada gambar 4.22.

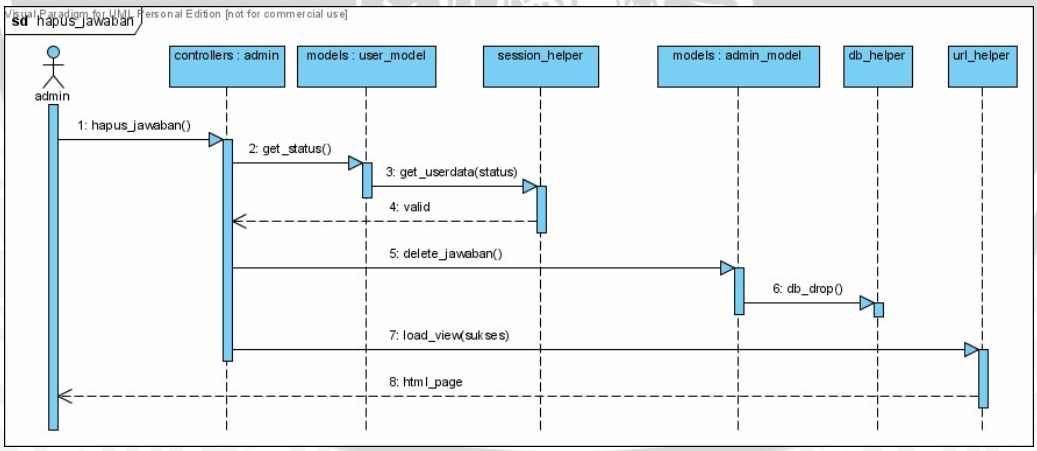


Gambar 4.22 Diagram Sekuensial untuk *Use Case* Hapus Pertanyaan

Sumber: Perancangan

4.2.4.17 Diagram Sekuensial Hapus Jawaban

Diagram sekuensial ini menggambarkan interaksi yang terjadi ketika seorang user melakukan aksi hapus jawaban. Diagram sekuensial hapus jawaban ditunjukkan pada gambar 4.23.

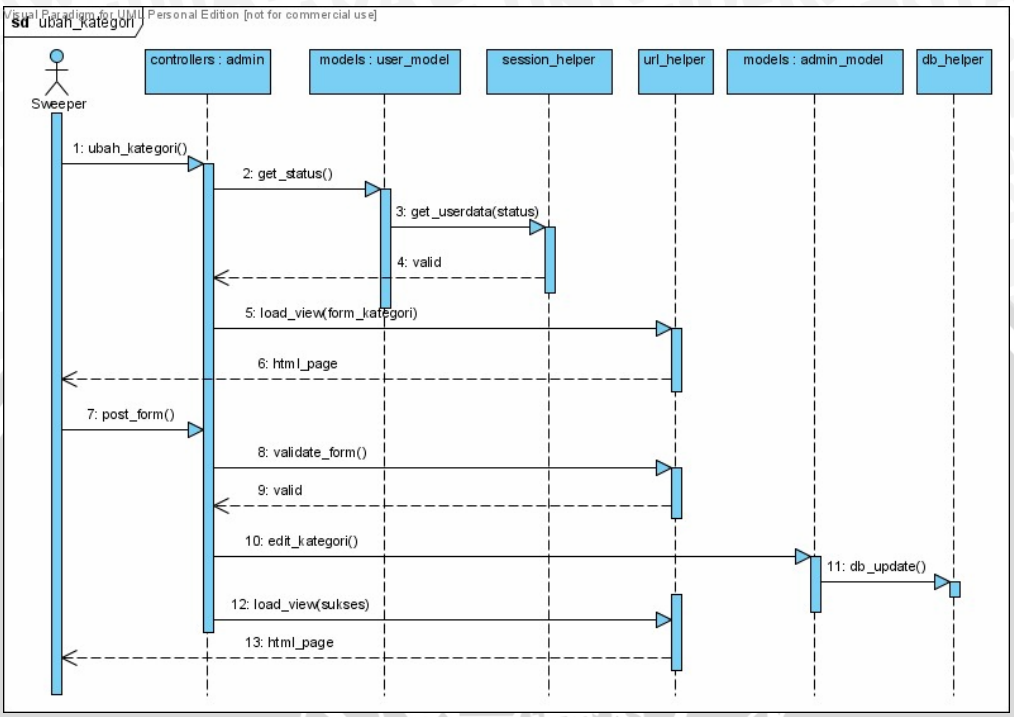


Gambar 4.23 Diagram Sekuensial untuk *Use Case* Hapus Jawaban

Sumber: Perancangan

4.2.4.18 Diagram Sekuensial Ubah Kategori

Diagram sekuensial ini menggambarkan interaksi yang terjadi ketika seorang user melakukan aksi ubah kategori. Diagram sekuensial ubah kategori ditunjukkan pada gambar 4.24.



Gambar 4.24 Diagram Sekuensial untuk *Use Case* Ubah Kategori. Sumber: Perancangan

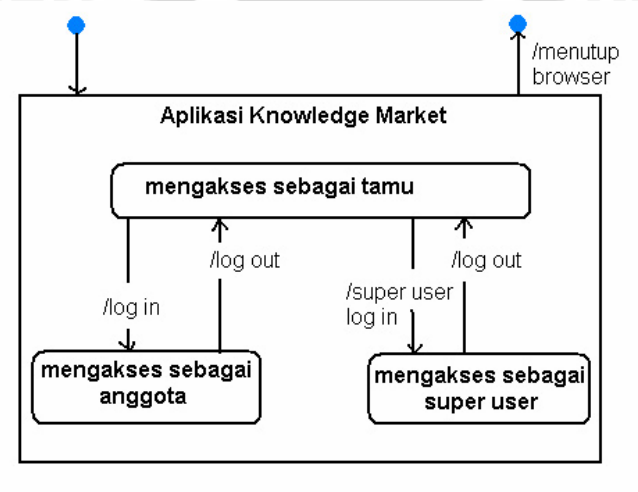
4.2.5 Diagram State (*Statechart Diagram*)

Diagram state menggambarkan perubahan kondisi atau keadaan (*state*) dalam suatu sistem. Setiap *inner-state* dapat menuju ke *inner-state* yang lainnya. Kondisi pada aplikasi *knowledge market* ini direpresentasikan dengan perubahan data status pada *session*.

Pertama kali pengguna mengakses aplikasi, maka pengguna diberikan status sebagai tamu. Dengan status ssebagai tamu, pengguna hanya dapat melihat-lihat fasilitas dan data yang terdapat dalam aplikasi, tetapi tidak dapat melakukan pengiriman atau penyimpanan data baru.

Untuk dapat melakukan proses-proses yang lebih lanjut, pengguna harus berubah kondisi menjadi anggota. Untuk mendapatkan status anggota, pengguna harus melakukan proses *log in* terlebih dahulu. Dengan status sebagai anggota, pengguna dapat melakukan proses-proses pengiriman dan penyimpanan data, tetapi tidak dapat melakukan proses-proses manajemen data dan manajemen keanggotaan.

Untuk dapat melakukan proses-proses manajemen, pengguna harus berubah kondisi menjadi *admin* atau *sweeper*. Untuk mendapatkan status tersebut, pengguna harus melakukan proses *log in* sebagai *super user* terlebih dahulu. Dengan status sebagai *super user*, pengguna dapat melakukan proses-proses pengiriman dan penyimpanan data dan juga proses-proses manajemen data dan manajemen keanggotaan. Diagram state aplikasi *knowledge market* ditunjukkan pada gambar 4.25.



Gambar 4.25 Diagram State Aplikasi Secara Umum

Sumber: Perancangan

BAB V
IMPLEMENTASI

Bab ini membahas implementasi perangkat lunak berdasarkan hasil yang telah didapatkan dari analisis kebutuhan dan proses perancangan perangkat lunak. Pembahasan terdiri atas penjelasan mengenai spesifikasi lingkungan (spesifikasi perangkat keras dan perangkat lunak) di mana sistem diimplementasikan, implementasi basis data (*database*), implementasi klas pada file program, implementasi algoritma, implementasi antarmuka aplikasi, implementasi *hosting*, dan implementasi modul keamanan (*security*).

5.1 Spesifikasi Sistem

Hasil analisis kebutuhan dan perancangan pada Bab 4, diimplementasikan menjadi sebuah sistem yang nyata agar dapat berfungsi sesuai kebutuhan perangkat lunak.

5.1.1 Spesifikasi Perangkat Keras

Spesifikasi perangkat keras yang digunakan untuk implementasi aplikasi ini ditunjukkan pada Tabel 5.1.

Tabel 5.1 Spesifikasi Perangkat Keras Komputer untuk Implementasi

Spesifikasi Perangkat Keras	
Prosessor	Intel(R) Pentium(R) 4 CPU 1.70 GHz
Memori (RAM)	256 MB DDR
Hard Disk	Seagate ST380011A (80 GB, kapasitas 80 GB, 7200 RPM
Motherboard	Asus P4B533

Sumber: Implementasi

5.1.2 Spesifikasi Perangkat Lunak

Spesifikasi perangkat lunak yang digunakan untuk implementasi aplikasi ini ditunjukkan pada Tabel 5.2.

Tabel 5.2 Spesifikasi Perangkat Lunak Komputer untuk Implementasi

Spesifikasi Perangkat Lunak	
Sistem Operasi	Windows XP SP2 versi 2002
Bahasa Pemrograman	PHP 5.2.8
Basis Data (*Database*)	MySQL Version 5.1.30

Spesifikasi Perangkat Lunak	
Web Server	Apache Server 2.2.11
Browsers	Opera 10.10, Mozilla Firefox 3.5, Internet Explorer 6
Tools Pemrograman	PHP Designer 7 (PHP Editor) phpMyAdmin 3.1.1 (MySQL Editor)
Framework	Code Igniter 1.7.2

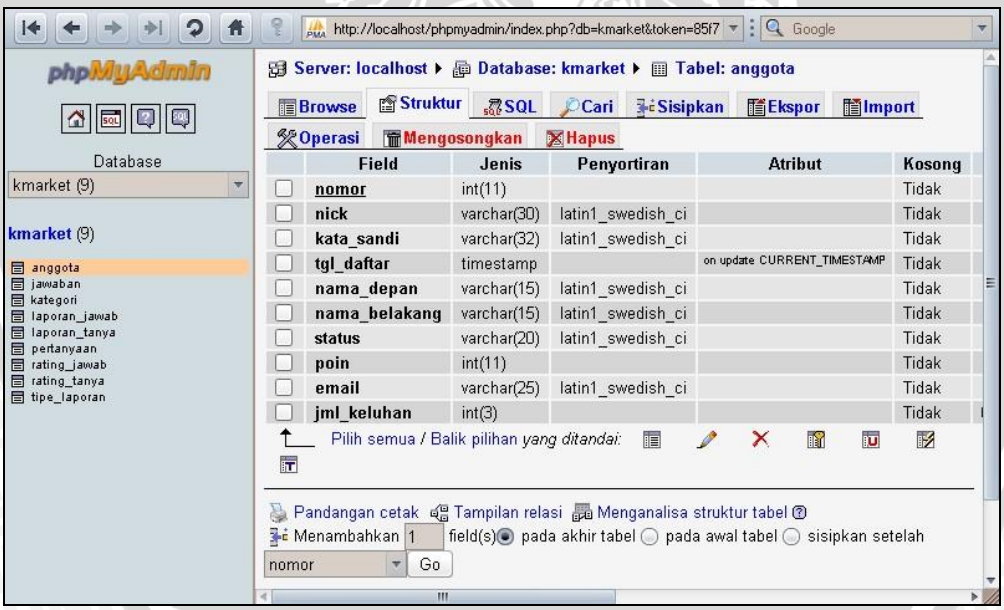
Sumber: Implementasi

5.2 Implementasi Basis Data (*Database*)

Hasil perancangan basis data menggunakan ERD pada Bab 4 diimplementasikan menjadi sebuah basis data yang nyata menggunakan sistem basis data MySQL dengan *tools* phpMyAdmin.

5.2.1 Implementasi Tabel Anggota

Tabel anggota terdiri dari sepuluh kolom (*field*) yaitu nomor, nick, kata_sandi, tgl_daftar, nama_depan, nama_belakang, status, poin, email, dan jml_keluha n. Struktur tabel anggota hasil implementasi ditunjukkan pada gambar 5.1.

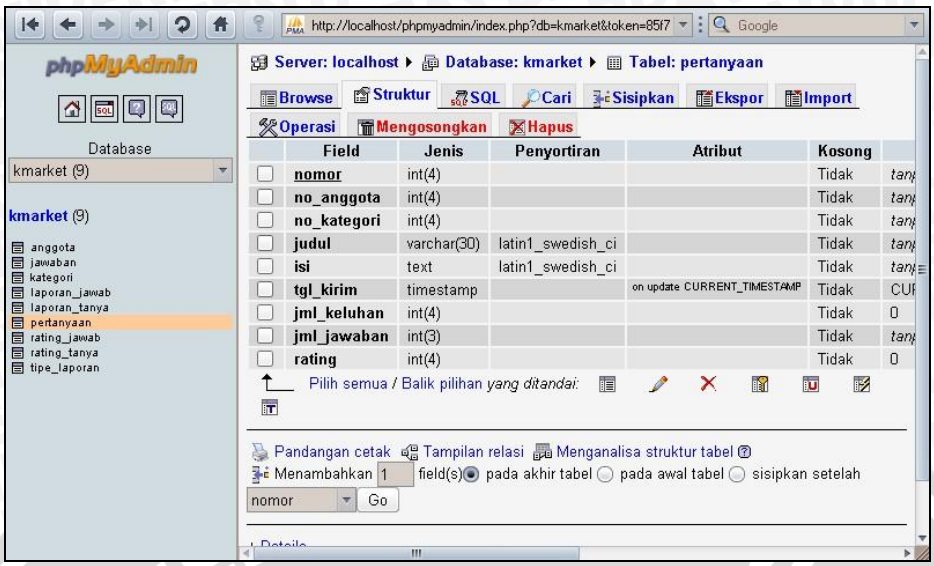


Gambar 5.1 Implementasi Tabel Anggota

Sumber: Implementasi

5.2.2 Implementasi Tabel Pertanyaan

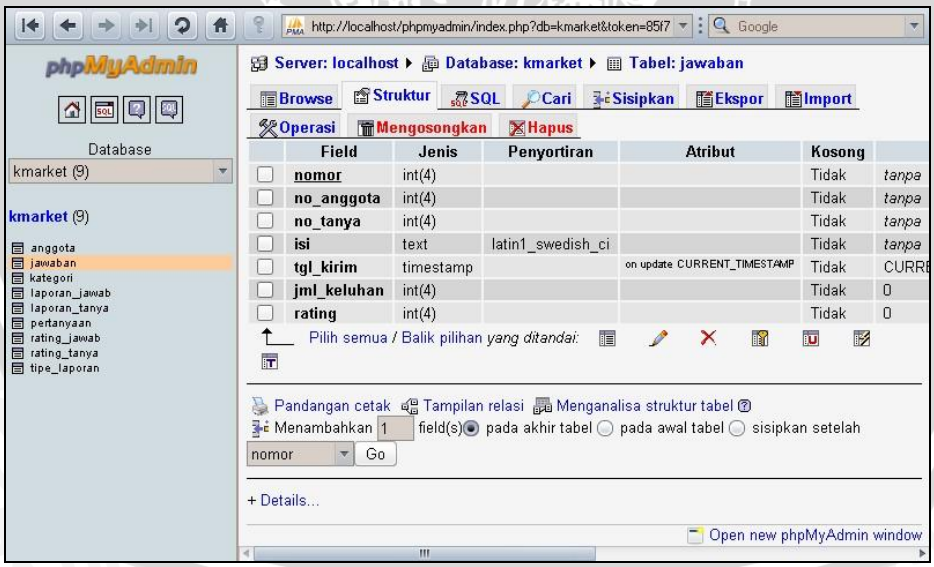
Tabel pertanyaan terdiri dari sembilan kolom yaitu nomor, no_anggota, no_kategori, judul, isi, tgl_kirim, jml_keluhan, jml_jawaban, dan rating. Struktur tabel pertanyaan hasil implementasi ditunjukkan pada gambar 5.2.



Gambar 5.2 Implementasi Tabel Pertanyaan
Sumber: Implementasi

5.2.3 Implementasi Tabel Jawaban

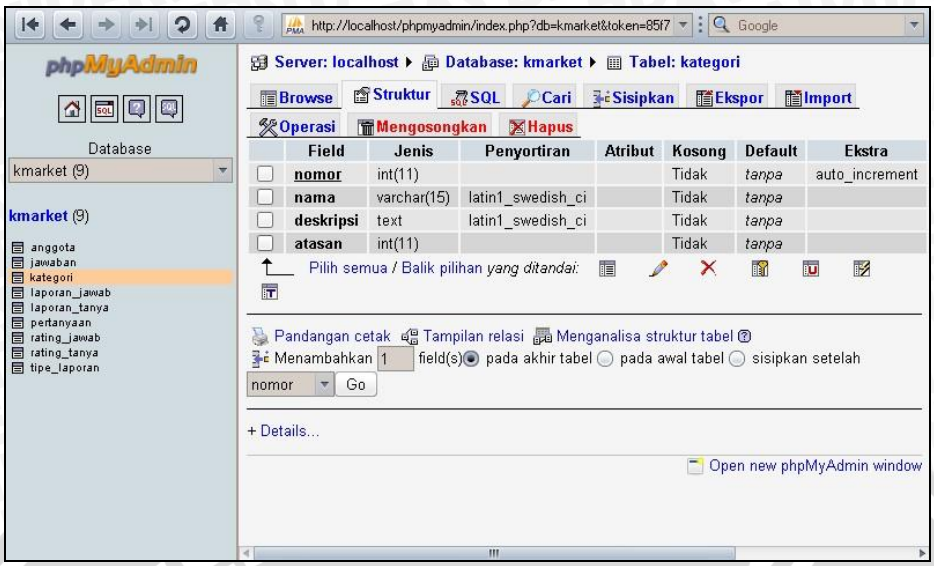
Tabel jawaban terdiri dari tujuh kolom yaitu nomor, no_anggota, no_tanya, isi, tgl_kirim, jml_keluhan, dan rating. Struktur tabel jawaban hasil implementasi ditunjukkan pada gambar 5.3.



Gambar 5.3 Implementasi Tabel Jawaban
Sumber: Implementasi

5.2.4 Implementasi Tabel Kategori

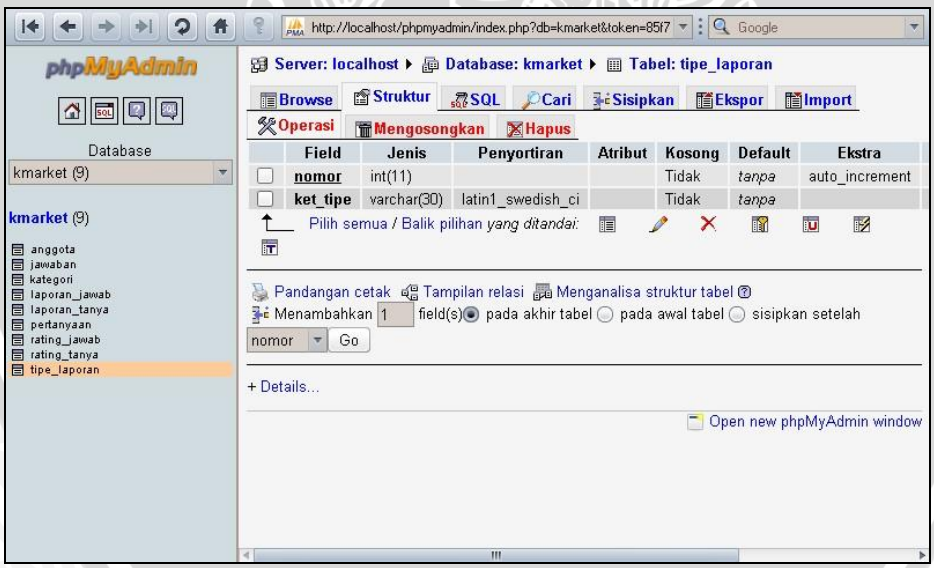
Tabel kategori terdiri dari empat kolom yaitu nomor, nama, deskripsi, dan atasan. Struktur tabel kategori hasil implementasi ditunjukkan pada gambar 5.4.



Gambar 5.4 Implementasi Tabel Kategori
Sumber: Implementasi

5.2.5 Implementasi Tabel Tipe_Laporan

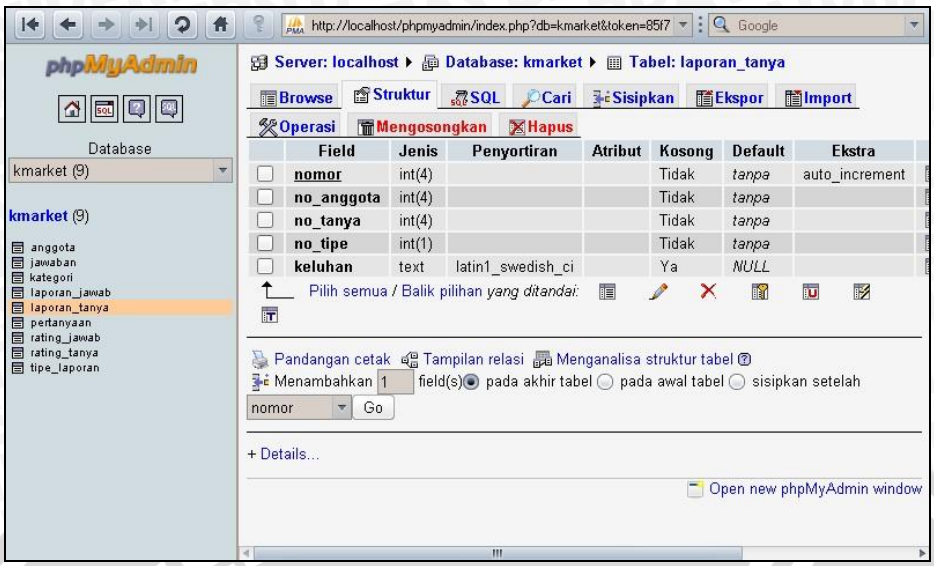
Tabel tipe_laporan terdiri dari dua kolom yaitu nomor dan ket_tipe. Struktur tabel tipe_laporan hasil implementasi ditunjukkan pada gambar 5.5.



Gambar 5.5 Implementasi Tabel Tipe_Laporan
Sumber: Implementasi

5.2.6 Implementasi Tabel Laporan_Tanya

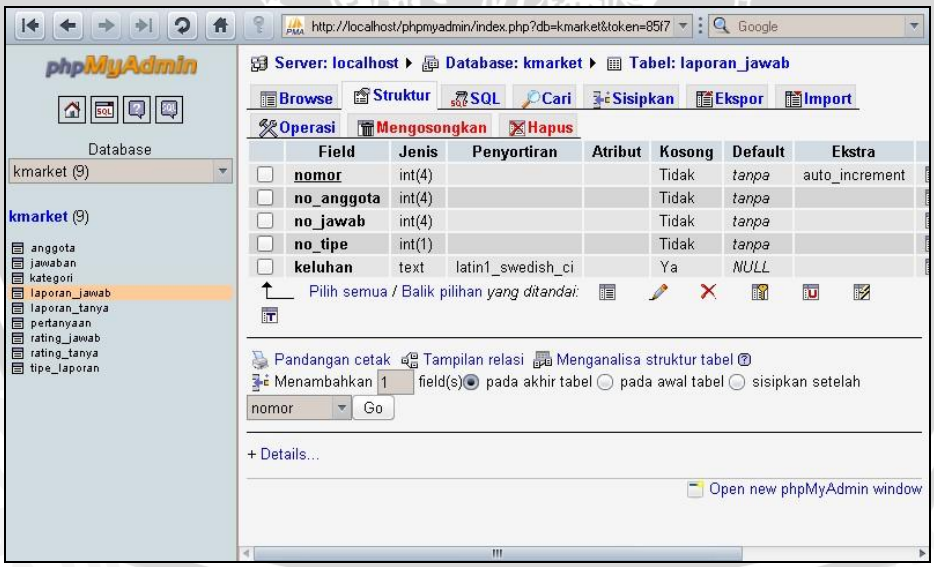
Tabel laporan_tanya terdiri dari lima kolom yaitu nomor, no_anggota, no_tanya, no_tipe, dan keluhan. Struktur tabel laporan_tanya hasil implementasi ditunjukkan pada gambar 5.6.



Gambar 5.6 Implementasi Tabel Laporan_Tanya
Sumber: Implementasi

5.2.7 Implementasi Tabel Laporan_Jawab

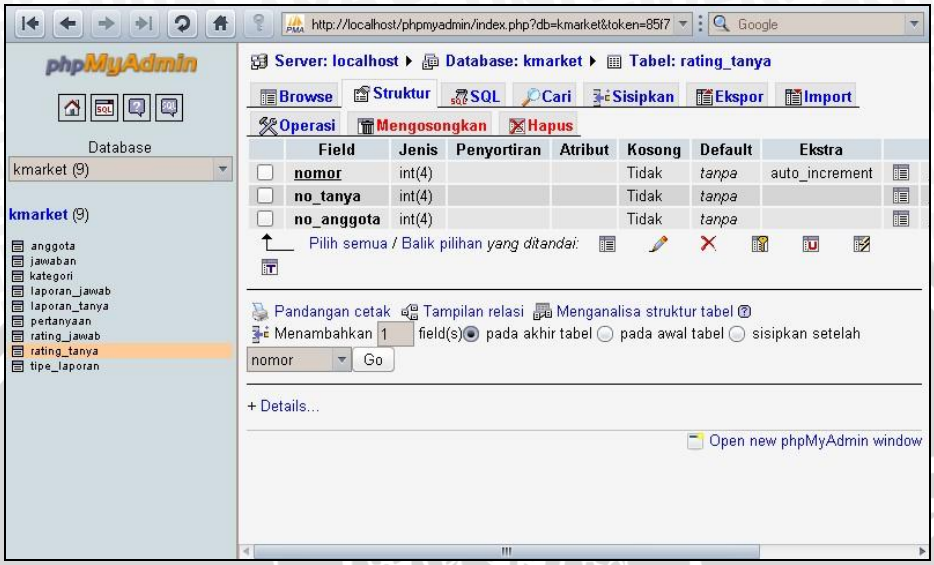
Tabel laporan_jawab terdiri dari lima kolom yaitu nomor, no_anggota, no_jawab, no_tipe, dan keluhan. Struktur tabel laporan_jawab hasil implementasi ditunjukkan pada gambar 5.7.



Gambar 5.7 Implementasi Tabel Laporan_Jawab
Sumber: Implementasi

5.2.8 Implementasi Tabel Rating_Tanya

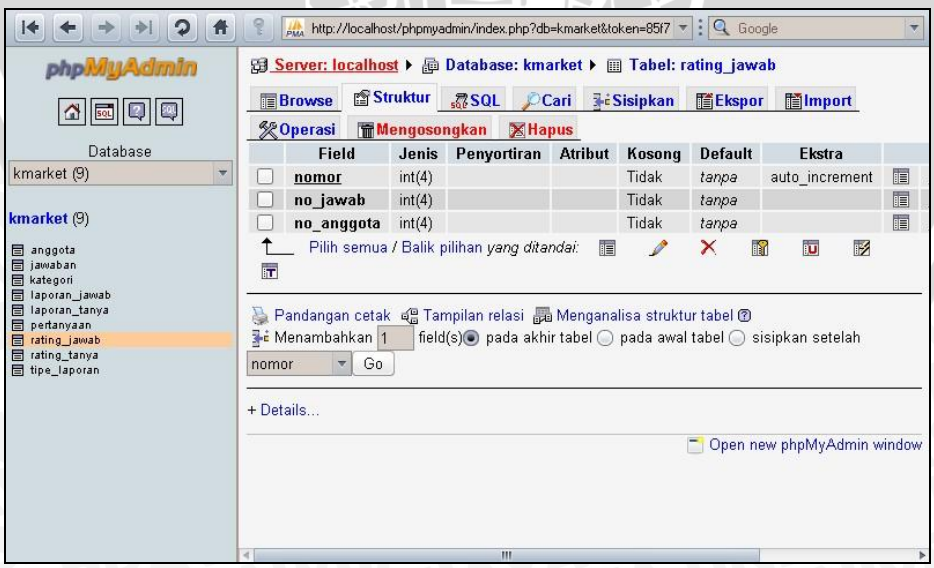
Tabel rating_tanya terdiri dari tiga kolom yaitu nomor, no_tanya, dan no_anggota. Struktur tabel rating_tanya hasil implementasi ditunjukkan pada gambar 5.8.



Gambar 5.8 Implementasi Tabel Rating_Tanya
Sumber: Implementasi

5.2.9 Implementasi Tabel Rating_Jawab

Tabel rating_jawab terdiri dari tiga kolom yatiu nomor, no_jawab, dan no_anggota. Struktur tabel rating_jawab hasil implementasi ditunjukkan pada gambar 5.9.



Gambar 5.9 Implementasi Tabel Rating_Jawab
Sumber: Implementasi

5.3. Implementasi Klas pada File Program

Klas-klas yang telah dirancang pada proses perancangan masing-masing direalisasikan pada sebuah file program bertipe php. Pasangan klas dengan file program yang digunakan untuk mengimplentasikannya ditunjukkan pada tabel 5.3.

Tabel 5.3 Implementasi Klas pada File Program

No.	Nama Klas	Nama File Program
Klas Controller		
1.	Utama	utama.php
2.	Admin	admin.php
3.	Detail	detail.php
4.	Pengiriman	pengiriman.php
Klas Model		
7.	Utama	utama_model.php
8.	Admin	admin_model.php
9.	User	user_model.php
10.	Pengiriman	pengiriman_model.php

Sumber: Implementasi

5.4 Implementasi Algoritma

Pada sistem ini, terdapat beberapa proses utama. Di antara proses utama tersebut adalah proses tambah_anggota, hapus_anggota, tambah_pertanyaan, hapus_pertanyaan, tambah_jawaban, dan hapus_jawaban. Algoritma merupakan salah satu bagian penting yang harus dirumuskan karena berguna untuk memudahkan proses selanjutnya yaitu proses pengujian.

5.4.1 Implementasi Algoritma Registrasi

Algoritma registrasi digunakan untuk memasukkan data user baru yang dilakukan sendiri oleh user yang bersangkutan. Algoritma ini diterapkan pada metode (*method*) yang terdapat pada klas admin_kontroler dan ditunjukkan pada algoritma 5.1.

```
Implementasi algoritma registrasi pada klas admin_kontroler
(Algoritma untuk memasukkan data user ke dalam basis data)

BEGIN

DESCRIPTION
1       show form registrasi
2       get information from form
3       validate form
4       IF form IS valid
5               call insert_anggota
6       ELSE
7               show error_message
8       END IF

END
```

Algoritma 5.1 Implementasi Algoritma Registrasi pada Klas Admin_kontroller
Sumber: Implementasi

5.4.2 Implementasi Algoritma Tambah Admin

Algoritma tambah_admin digunakan untuk memasukkan data user admin atau sweeper baru yang prosesnya dilakukan oleh admin lain yang sudah terdaftar. Algoritma ini diterapkan pada metode (*method*) yang terdapat pada klas admin_kontroler dan ditunjukkan pada algoritma 5.2.

5.4.3 Implementasi Algoritma Hapus Anggota

Algoritma hapus_anggota digunakan untuk menghapus data user yang hanya dapat dilakukan oleh otritas tertinggi yaitu admin. Algoritma ini diterapkan pada metode (*method*) yang terdapat pada klas admin_kontroler dan ditunjukkan pada algoritma 5.3.

```
Implementasi algoritma tambah_admin pada klas admin_kontroler
(Algoritma untuk memasukkan data  admin ke dalam basis data)

BEGIN

DESCRIPTION
1      get member_status
2      IF member_status IS admin
3             show form tambah_admin
4             get information from form
5             validate form
6             IF form IS valid
7                    call insert_admin
8             ELSE
9                    show error_message
10            END IF
11     ELSE
12            show error_message
13     ENDIF

END
```

Algoritma 5.2 Implementasi Algoritma Tambah Admin pada Klas Admin_kontroller
Sumber: Implementasi

```
Implementasi algoritma hapus_anggota pada klas admin_kontroler
(Algoritma untuk menghapus  data user dari basis data)

BEGIN

DESCRIPTION
1      get member_status
2      IF member_status IS admin
3             get nomor_anggota
4             call delete_anggota(nomor_anggota)
5      ELSE
6             show error_message
7      END IF

END
```

Algoritma 5.3 Implementasi Algoritma Hapus Anggota pada Klas Admin_kontroller
Sumber: Implementasi

5.4.4 Implementasi Algoritma Tambah Pertanyaan

Algoritma tambah_pertanyaan digunakan untuk memasukkan data pertanyaan baru yang dilakukan oleh member. Algoritma ini diterapkan pada metode (*method*) yang terdapat pada klas pengiriman_kontroler dan ditunjukkan pada algoritma 5.4.

```
Implementasi algoritma tambah_pertanyaan pada klas pengiriman_kontroler
(Algoritma untuk memasukkan data user ke dalam basis data)

BEGIN

DESCRIPTION
1      GET logged_status
2      IF logged_status IS false
3             show error_message
4      ELSE
5             GET quota
6             IF quota IS 0
7                    show error_message
8             ELSE
9                    show form kirim_pertanyaan
10                   validate form
11                   IF form IS valid
12                          CALL insert_pertanyaan
13                   ELSE
14                          show error_message
15                   END IF
16            ENDIF
17     ENDIF

END

END
```

Algoritma 5.4 Implementasi Algoritma Tambah Pertanyaan pada Klas Pengiriman_kontroller
Sumber: Implementasi

5.4.5 Implementasi Algoritma Hapus Pertanyaan

Algoritma hapus_pertanyaan digunakan untuk menghapus data pertanyaan yang hanya dapat dilakukan oleh otoritas tinggi yaitu admin atau sweeper. Algoritma ini diterapkan pada metode (*method*) yang terdapat pada klas admin_kontroler dan ditunjukkan pada algoritma 5.5.

```
Implementasi algoritma hapus_pertanyaan pada klas admin_kontroler
(Algoritma untuk menghapus data pertanyaan dari basis data)

BEGIN

DESCRIPTION
1      get member_status
2      IF member_status IS admin
3             get nomor_pertanyaan
4             call delete_jawaban(nomor_pertanyaan)
5             call delete_pertanyaan(nomor_pertanyaan)
6      ELSE
7             show error_message
8      END IF

END
```

Algoritma 5.5 Implementasi Algoritma Hapus Pertanyaan pada Klas Admin_kontroller
Sumber: Implementasi

5.4.6 Implementasi Algoritma Tambah Jawaban

Algoritma tambah_jawaban digunakan untuk memasukkan data jawaban baru yang dilakukan oleh member. Algoritma ini diterapkan pada metode (*method*) yang terdapat pada klas pengiriman_kontroler dan ditunjukkan pada algoritma 5.6.

5.4.7 Implementasi Algoritma Hapus Jawaban

Algoritma hapus_jawaban digunakan untuk menghapus data jawaban yang hanya dapat dilakukan oleh otoritas tinggi yaitu admin atau sweeper. Algoritma ini diterapkan pada metode (*method*) yang terdapat pada klas admin_kontroler dan ditunjukkan pada algoritma 5.7.

```
Implementasi algoritma tambah_jawaban pada klas pengiriman_kontroler
(Algoritma untuk memasukkan data  jawaban ke dalam basis data)

BEGIN

DESCRIPTION
1      GET logged_status
2      IF logged_status IS false
3             show error_message
4      ELSE
5             get nomor_pertanyaan
6             show form kirim_jawaban
7             validate form
8             IF form IS valid
9                    CALL insert_jawaban(nomor_pertanyaan)
10            ELSE
11                   show error_message
12            ENDIF
13     ENDIF

END
```

Algoritma 5.6 Implementasi Algoritma Tambah Jawaban pada Klas Pengiriman_kontroller
Sumber: Implementasi

```
Implementasi algoritma hapus_jawaban pada klas admin_kontroler
(Algoritma untuk menghapus  data jawaban dari basis data)

BEGIN

DESCRIPTION
1      get member_status
2      IF member_status IS admin
3             get nomor_jawaban
4             call delete_jawaban(nomor_jawaban)
5      ELSE
6             show error_message
7      END IF

END
```

Algoritma 5.7 Implementasi Algoritma Hapus Jawaban pada Klas Admin_kontroller
Sumber: Implementasi

5.5 Implementasi Antarmuka

Antarmuka merupakan bagian penting yang sangat menunjang fungsionalitas suatu situs web. Pada situs Internet-Based Knowledge Market ini perancangan

antarmuka secara garis besar terdiri dari dua jenis, yaitu halaman berisi form dan halaman berisi hasil query basis data.

5.5.1 Implementasi Antarmuka Halaman Utama

Antarmuka halaman utama akan ditampilkan ketika pengguna masuk ke dalam sistem dengan mengetikkan alamat aplikasi pada browser. Halaman utama berisi menu navigasi, deskripsi singkat situs, daftar pertanyaan terbaru, daftar pertanyaan belum terjawab, dan daftar pengguna dengan poin tertinggi. Antarmuka halaman utama ditunjukkan pada Gambar 5.10.

Gambar 5.10 Antarmuka Halaman Utama
Sumber: Implementasi

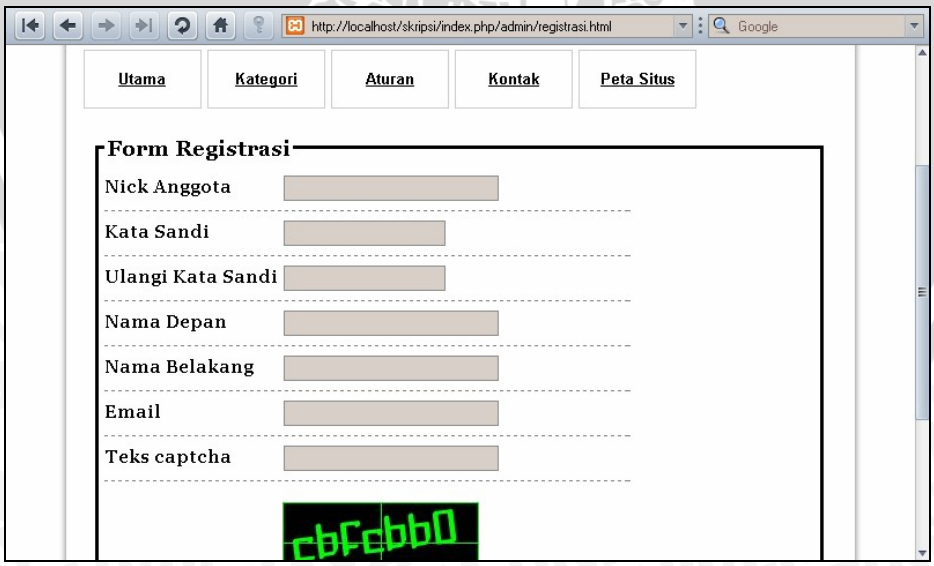
5.5.2 Implementasi Antarmuka Form Log In

Antarmuka form log in ini akan ditampilkan ketika pengguna ingin masuk ke dalam sistem agar dapat melakukan proses-proses pengiriman data. Pengguna harus melakukan log in terlebih dahulu untuk dapat mengakses seluruh fasilitas aplikasi. Antarmuka form log in ditunjukkan pada Gambar 5.11.

Gambar 5.11 Antarmuka Form Log In
Sumber: Implementasi

5.5.3 Implementasi Antarmuka Form Registrasi

Antarmuka form registrasi ini akan ditampilkan ketika pengguna ingin mendaftar menjadi anggota situs agar dapat melakukan proses -proses pengiriman data. Proses registrasi akan menyimpan nickname dan kata sandi yang diperlukan untuk melakukan proses log in. Antarmuka form registrasi ditunjukkan pada Gambar 5. 12.



Gambar 5.12 Antarmuka Registrasi
Sumber: Implementasi

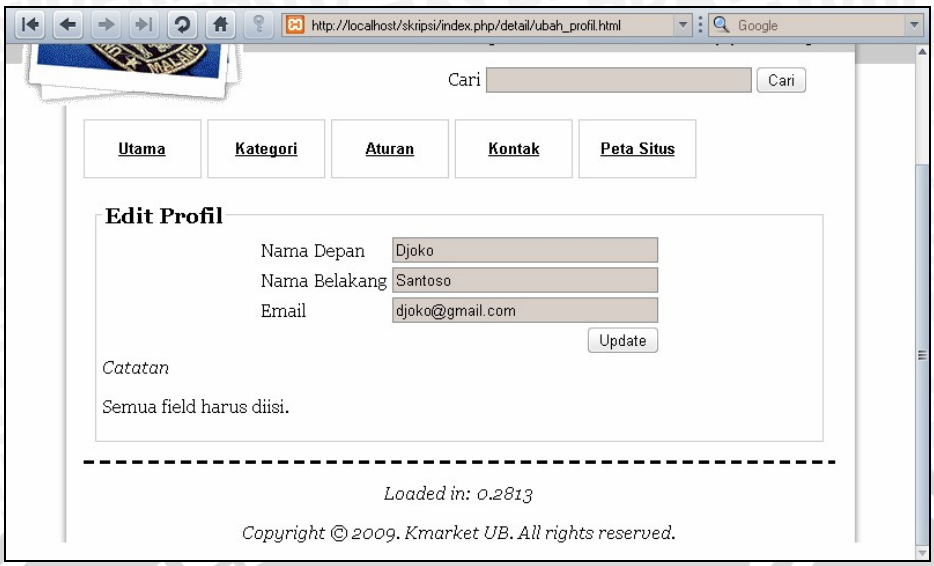
5.5.4 Implementasi Antarmuka Detail Profil

Antarmuka detail profil ini akan ditampilkan ketika pengguna menekan link nama anggota. Halaman detail profil menampilkan nama lengkap dan total poin dari seorang anggota. Antarmuka detail profil ditunjukkan pada Gambar 5. 13.

5.5.5 Implementasi Antarmuka Form Ubah Profil

Antarmuka form ubah profil ini akan ditampilkan ketika pengguna ingin mengubah data diri pengguna tersebut yang telah disimpan sebelumnya dengan melakukan proses registrasi. Proses ubah profil dapat mengubah nama dan email pengguna. Antarmuka form ubah profil ditunjukkan pada Gambar 5. 14.

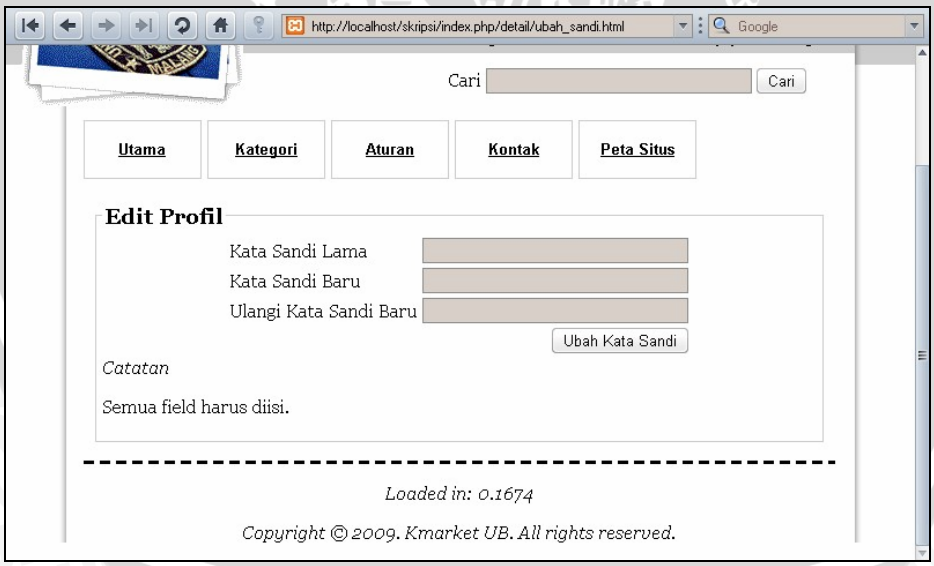
Gambar 5.13 Antarmuka Detail Profil
Sumber: Implementasi



Gambar 5.14 Antarmuka Form Ubah Profil
Sumber: Implementasi

5.5.6 Implementasi Antarmuka Form Ubah Sandi

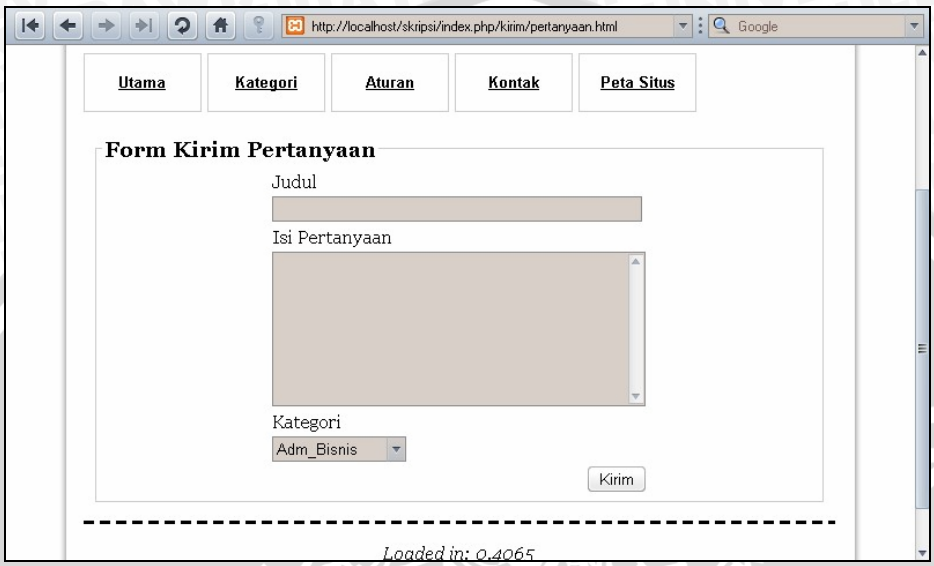
Antarmuka form ubah sandi ini akan ditampilkan ketika pengguna ingin mengubah data kata sandi yang digunakan untuk melakukan proses log in . Antarmuka form ubah sandi ditunjukkan pada Gambar 5. 15.



Gambar 5.15 Antarmuka Form Ubah Sandi
Sumber: Implementasi

5.5.7 Implementasi Antarmuka Form Kirim Pertanyaan

Antarmuka form kirim pertanyaan ini akan ditampilkan ketika pengguna ingin melakukan proses pengiriman data pertanyaan baru. Antarmuka form kirim pertanyaan ditunjukkan pada Gambar 5.16.



Gambar 5.16 Antarmuka Form Kirim Pertanyaan
Sumber: Implementasi

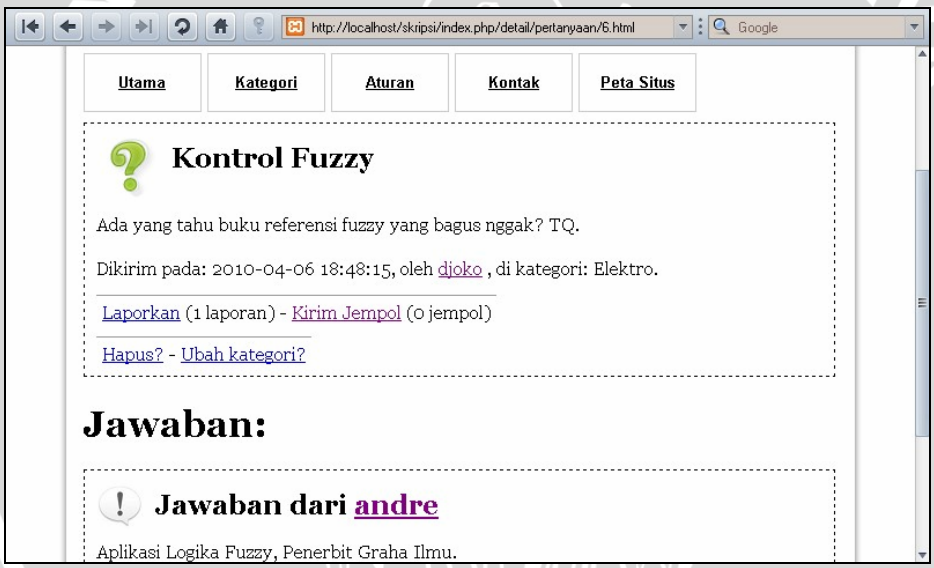
5.5.8 Implementasi Antarmuka Form Kirim Jawaban

Antarmuka form kirim jawaban ini akan ditampilkan ketika pengguna ingin melakukan proses pengiriman data jawaban. Antarmuka form kirim jawaban ditunjukkan pada Gambar 5.17.

5.5.9 Implementasi Antarmuka Detail Pertanyaan

Antarmuka detail pertanyaan akan ditampilkan ketika pengguna ingin melihat detail dari suatu pertanyaan. Antarmuka detail pertanyaan ditunjukkan pada Gambar 5.18.

Gambar 5.17 Antarmuka Form Kirim Jawaban
Sumber: Implementasi



Gambar 5.18 Antarmuka Detail Pertanyaan
Sumber: Implementasi

5.5.10 Implementasi Antarmuka Daftar Pertanyaan

Antarmuka daftar pertanyaan akan ditampilkan ketika pengguna ingin melihat daftar pertanyaan yang telah tersimpan di dalam basis data. Antarmuka daftar pertanyaan ditunjukkan pada Gambar 5. 19.

Gambar 5.19 Antarmuka Daftar Pertanyaan
Sumber: Implementasi

5.5.11 Implementasi Antarmuka Daftar Kategori

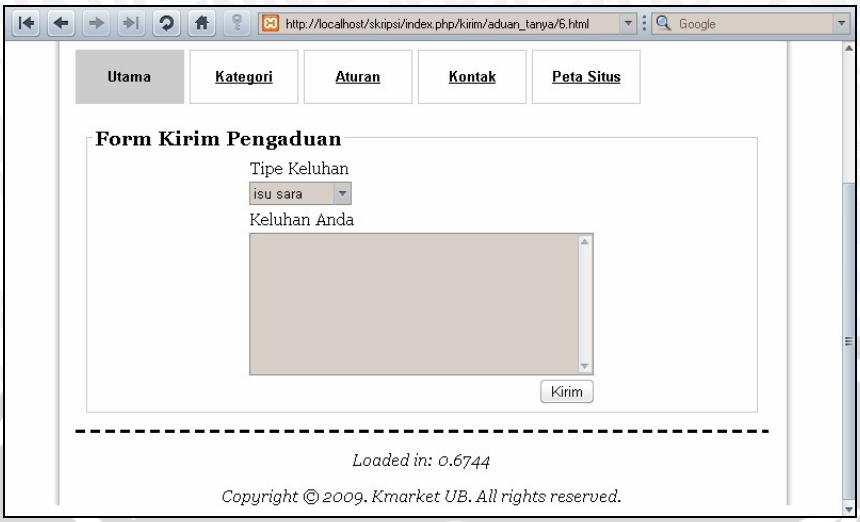
Antarmuka daftar kategori akan ditampilkan ketika pengguna ingin melihat daftar kategori yang disediakan oleh aplikasi. Antarmuka daftar kategori ditunjukkan pada Gambar 5.20.

Gambar 5.20 Antarmuka Daftar Kategori
Sumber: Implementasi

5.5.12 Implementasi Antarmuka Form Pengaduan Penyalahgunaan

Antarmuka form pengaduan penyalahgunaan ini akan ditampilkan ketika pengguna ingin melakukan proses pengiriman data aduan penyalahgunaan, baik untuk

data pertanyaan maupun untuk data jawaban. Antarmuka form pengaduan penyalahgunaan ditunjukkan pada Gambar 5. 21.



Gambar 5.21 Antarmuka Form Pengaduan Penyalahgunaan
Sumber: Implementasi

5.5.13 Implementasi Antarmuka Form Pencarian Tingkat Lanjut

Antarmuka form pecarian tingkat lanjut akan ditampilkan ketika pengguna ingin melakukan proses pencarian data dengan tingkat pencarian yang lebih detail. Antarmuka form pencarian tingkat lanjut ditunjukkan pada Gamb ar 5.22.

5.5.14 Implementasi Antarmuka Peta Situs

Antarmuka peta situs akan ditampilkan ketika pengguna ingin melihat daftar isi dan fasilitas bantuan yang ada di dalam aplikasi. Antarmuka peta situs ditunjukkan pada Gambar 5.23.

Gambar 5.22 Antarmuka Form Pencarian Tingkat Lanjut
Sumber: Implementasi

Gambar 5.23 Antarmuka Peta Situs
Sumber: Implementasi

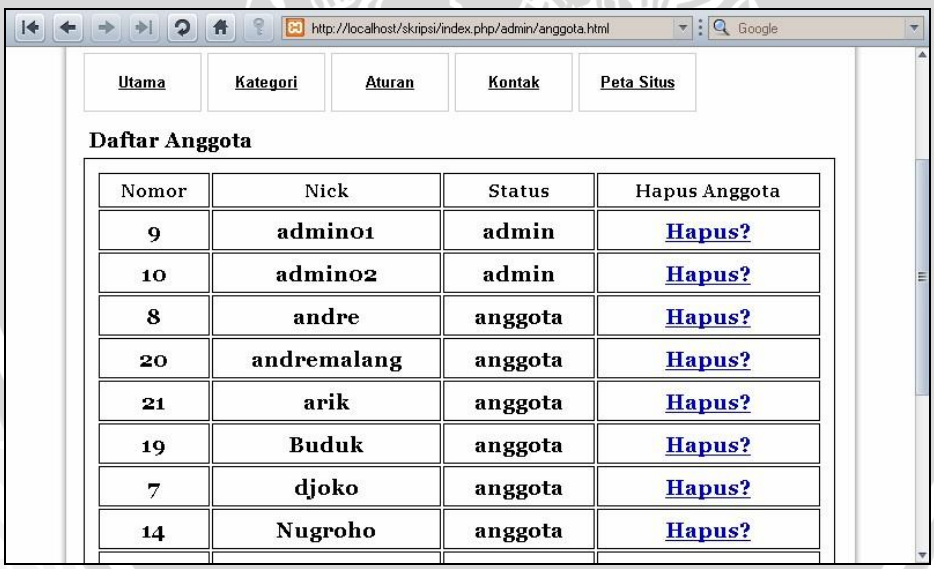
5.5.15 Implementasi Antarmuka Menu Anggota dan Admin

Antarmuka menu anggota dan admin akan ditampilkan ketika anggota atau admin ingin melihat proses apa saja yang dapat dilakukan. Antarmuka menu anggota dan admin ditunjukkan pada Gambar 5. 24.

Gambar 5.24 Antarmuka Menu Admin
Sumber: Implementasi

5.5.16 Implementasi Antarmuka Daftar Anggota

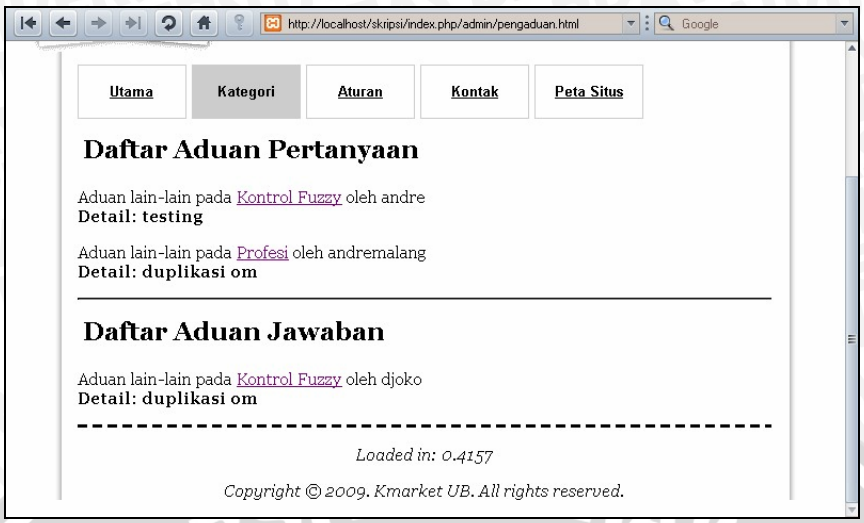
Antarmuka daftar anggota akan ditampilkan ketika admin ingin melihat daftar anggota. Antarmuka daftar anggota ditunjukkan pada Gambar 5. 25.



Gambar 5.25 Antarmuka Daftar Anggota
Sumber: Implementasi

5.5.17 Implementasi Antarmuka Daftar Pengaduan Penyalahgunaan

Antarmuka daftar pengaduan penyalahgunaan akan ditampilkan ketika admin ingin melihat daftar pengaduan penyalahgunaan. Antarmuka daftar pengaduan penyalahgunaan ditunjukkan pada Gambar 5. 26.



Gambar 5.26 Antarmuka Daftar Pengaduan Penyalahgunaan
Sumber: Implementasi

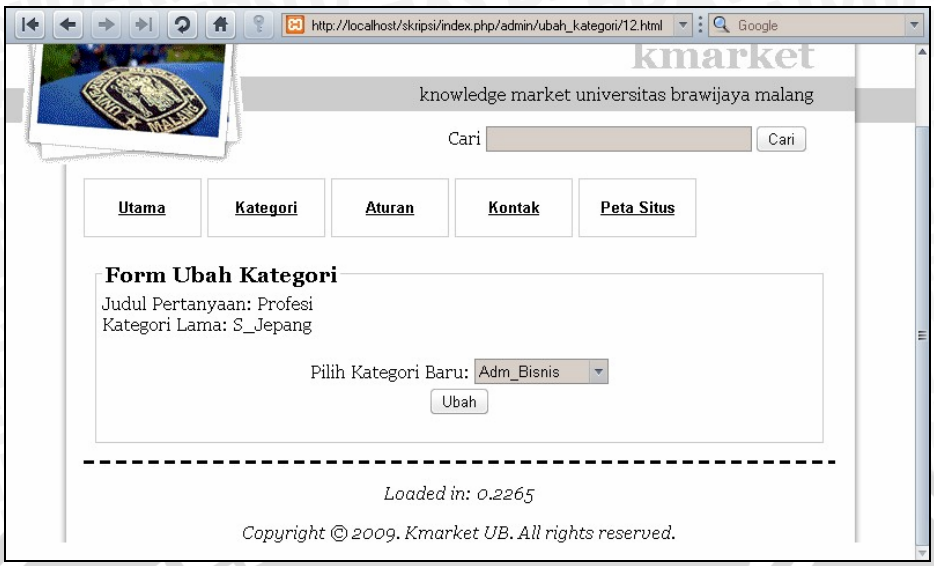
5.5.18 Implementasi Antarmuka Form Tambah Admin

Antarmuka form tambah admin akan ditampilkan ketika admin ingin melakukan proses penambahan data admin baru. Antarmuka form tambah admin ditunjukkan pada Gambar 5.27.

Gambar 5.27 Antarmuka Form Tambah Admin
Sumber: Implementasi

5.5.19 Implementasi Antarmuka Form Ubah Kategori

Antarmuka form ubah kategori akan ditampilkan ketika admin ingin melakukan proses pengubahan data kategori dari suatu pertanyaan. Antarmuka form ubah kategori ditunjukkan pada Gambar 5.28.



Gambar 5.28 Antarmuka Form Ubah Kategori
Sumber: Implementasi

5.6 Implementasi *Hosting* di *Remote Hosting*

Setelah aplikasi telah dibagun pada *local hosting* maka selanjutanya dilakukan proses *upload* data pada *remote hosting*. Proses ini sering disebut dengan istilah *deployment* yang mana bertujuan untuk menguji apakah sistem dapat berjalan pada lingkungan server dengan spesifikasi di lingkungan nyata yang mungkin berbeda dengan lingkungan pengembangan.

Proses dilakukan pertama-tama dengan mendaftar pada hosting gratis yang mendukung PHP dan MySQL. Hosting yang dipilih adalah http://www.000webhost.com. Setelah proses pendaftaran selesai, data dikirim melalui FTP (*File Transfer Protocol*) pada tempat yang telah ditentukan oleh pihak *hosting*.

5.7 Implementasi Modul Keamanan

Untuk meningkatkan keamanan aplikasi, khususnya pada bagian basis data, maka diperlukan implementasi modul keamanan ke dalam aplikasi. Sistem keamanan dasar yang diimplementasikan adalah memanfaatkan klas *form validation* yang dimiliki oleh *framework* CodeIgniter. Dengan memanfaatkan klas ini, masukan dari tiap *form* dapat diberi batasan-batasan terkait dengan panjang masukan minimal, panjang masukan maksimal, dan juga tipe masukan (email, alfabet, atau angka). Dengan batasan ini, masukan yang panjang seperti perintah *SQL injection* tidak dapat dieksekusi karena telah digunakan sistem validasi ini.

Gambar 5.29 Contoh Hasil Gambar dari Modul Captcha
Sumber: Implementasi

Sistem keamanan yang lebih lanjut diimplementasikan dengan membuat sebuah modul *captcha*, yaitu modul yang akan membuat suatu teks berformat gambar yang selanjutnya harus dimasukkan oleh pengguna ke dalam *form* yang disajikan. Modul *captcha* ini dibuat dengan memanfaatkan modul php-gd yang merupakan modul bawaan dari bahasa pemrograman php. Modul php-gd berfungsi untuk membuat suatu file gambar dengan masukan berupa teks tertentu. Adapun untuk fungsi keamanan, digunakan masukan acak yang kemudian dikonversi menggunakan enkripsi md-5. Contoh gambar yang dihasilkan oleh modul *captcha* ditunjukkan pada gambar 5.29.

BAB VI
PENGUJIAN DAN ANALISIS

Pada bab ini dilakukan proses pengujian dan analisis terhadap sistem yang telah dibangun. Proses pengujian terdiri atas tiga tahapan pengujian yaitu pengujian unit, pengujian integrasi, dan pengujian validasi. Pada pengujian unit dan pengujian integrasi akan digunakan teknik pengujian kotak putih (*white box testing*), sedangkan pada pengujian validasi akan digunakan teknik pengujian kotak hitam (*black box testing*). Proses analisis dilakukan pula untuk mengetahui unjuk kerja sistem perangkat lunak yang dibangun.

6.1 Pengujian

Proses pengujian dilakukan melalui tiga tahapan (strategi) pengujian yaitu pengujian unit, pengujian integrasi, dan pengujian validasi.

6.1.1 Pengujian Unit

Pengujian unit dilakukan untuk menguji unit terkecil dari sebuah perangkat lunak yaitu operasi dari sebuah klas. Penggunaan framework codeigniter menyebabkan terpisahnya kontroler dengan akses basis data, sehingga unit terkecil dari sistem yang perlu diuji adalah fungsi-fungsi yang terdapat dalam klas-klas model.

6.1.1.1 Pengujian Fungsi-Fungsi Dalam Klas Admin_Model

Klas admin_model berisi fungsi-fungsi yang menghubungkan kontroler dengan basis data atau *session* dalam kaitannya dengan aksi-aksi yang dapat dilakukan oleh admin. Fungsi-fungsi yang terdapat dalam klas admin_model antara lain: insert_admin, insert_anggota, delete_pertanyaan, delete_jawaban, dan ubah_kategori.

Penentuan kasus uji (*test case*) untuk setiap fungsi dan hasil eksekusi setiap fungsi dalam klas admin_model ditunjukkan pada Tabel 6.1.

Tabel 6.1 Test Case untuk Pengujian Fungsi-Fungsi Dalam Klas Admin_Model

No.	Nama Fungsi	Tujuan	Hasil yang diharapkan	Hasil yang didapatkan
1	insert_admin	Menambahkan data admin baru ke dalam tabel anggota	Data admin baru tersimpan dalam tabel anggota	Data admin baru tersimpan dalam tabel anggota
2	delete_anggota	Menghapus satu data anggota dari tabel anggota	Data anggota yang dihapus hilang dari basis data	Data anggota yang dihapus hilang dari basis data
3	delete_pertanyaan	Menghapus satu data pertanyaan dari tabel pertanyaan	Data pertanyaan yang dihapus hilang dari basis data	Data pertanyaan yang dihapus hilang dari basis data
4	delete_jawaban	Menghapus satu data jawaban dari tabel jawaban	Data jawaban yang dihapus hilang dari basis data	Data jawaban yang dihapus hilang dari basis data
5	ubah_kategori	Mengubah nilai data kategori dari suatu pertanyaan dalam tabel pertanyaan	Nilai data kategori dari pertanyaan yang dimaksud berubah	Nilai data kategori dari pertanyaan yang dimaksud berubah

Sumber: Pengujian

6.1.1.2 Pengujian Fungsi-Fungsi Dalam Klas Pengiriman_Model

Klas pengiriman_model berisi fungsi-fungsi yang menghubungkan kontroler dengan basis data atau *session* dalam kaitannya dengan aksi-aksi yang menyangkut fungsi pengiriman data oleh anggota. Fungsi-fungsi yang terdapat dalam klas pengiriman_model antara lain: insert_pertanyaan, insert_jawaban, update_pertanyaan, insert_aduan_tanya, insert_aduan_jawab, update_poin, dan update_quota.

Penentuan kasus uji (*test case*) untuk setiap fungsi dan hasil eksekusi setiap fungsi dalam klas pengiriman_model ditunjukkan pada Tabel 6.2.

Tabel 6.2 Test Case untuk Pengujian Fungsi-Fungsi Dalam Klas Pengiriman_Model

No.	Nama Fungsi	Tujuan	Hasil yang diharapkan	Hasil yang didapatkan
1	insert_pertanyaan	Menambahkan data pertanyaan baru ke dalam tabel pertanyaan	Data pertanyaan baru tersimpan dalam tabel pertanyaan	Data pertanyaan baru tersimpan dalam tabel pertanyaan
2	insert_jawaban	Menambahkan data jawaban baru ke dalam tabel jawaban	Data jawaban baru tersimpan dalam tabel jawaban	Data jawaban baru tersimpan dalam tabel jawaban
3	update_pertanyaan	Mengubah nilai data jml_jawaban pada tabel pertanyaan dengan nilai yang baru	Nilai data jml_jawaban pada tabel pertanyaan berubah	Nilai data jml_jawaban pada tabel pertanyaan berubah
4	insert_aduan_tanya	Menambahkan data aduan baru ke dalam tabel aduan_tanya	Data aduan_tanya baru tersimpan dalam tabel aduan_tanya	Data aduan_tanya baru tersimpan dalam tabel aduan_tanya
5	insert_aduan_jawab	Menambahkan data aduan baru ke dalam tabel aduan_jawab	Data aduan_jawab baru tersimpan dalam tabel aduan_ jawab	Data aduan_jawab baru tersimpan dalam tabel aduan_ jawab

No.	Nama Fungsi	Tujuan	Hasil yang diharapkan	Hasil yang didapatkan
6	update_poin	Mengubah nilai data poin pada tabel anggota dengan nilai yang baru	Nilai data poin pada tabel anggota berubah	Nilai data poin pada tabel anggota berubah
7	update_quota	Mengubah nilai data *quota* pada *session* dengan nilai yang baru	Nilai data *quota* pada *session* berubah	Nilai data *quota* pada *session* berubah

Sumber: Pengujian

6.1.1.3 Analisis Pengujian Unit

Dari hasil pengujian unit terhadap fungsi-fungsi yang terdapat dalam klas admin_model dan pengiriman_model dapat disimpulkan bahwa semua fungsi telah dapat menghasilkan keluaran yang diharapkan seperti dapat dilihat pada Tabel 6.1 dan 6.2. Dari hasil pengujian unit dapat dipastikan bahwa sistem telah memenuhi fungsi sesuai dengan kebutuhan perangkat lunak.

6.1.2 Pengujian Integrasi

Pengujian integrasi diterapkan pada proses yang mengintegrasikan fungsionalitas dari beberapa klas untuk melakukan sebuah operasi. Pada pengujian integrasi yang dijadikan sebagai objek uji adalah klas-klas yang menggabungkan kinerja dari klas-klas yang lain. Pengujian integrasi terhadap sistem perangkat lunak ini menggunakan strategi *bottom-up*, dimana klas-klas yang diintegrasikan masing-masing diuji terlebih dahulu dalam pengujian unit dan kemudian bergerak menuju ke pengujian klas-klas kontrol yang mengintegrasikannya.

Pengujian integrasi untuk sistem yang dibangun menggunakan *MVC framework* lebih ditekankan pada klas-klas kontroler karena pada umumnya di dalam klas kontroler dilakukan pemanggilan fungsi dari beberapa klas yang berbeda. Di dalam klas kontroler, sesuai dengan namanya, terdapat aliran kontrol yang dapat digambarkan menggunakan *pseudo-code*. *Pseudo-code* ini mewakili algoritma yang diperlukan untuk menentukan kasus uji dalam tahap pengujian.

Teknik selanjutnya yang digunakan dalam tahap pengujian integrasi adalah teknik *basis path testing*. Algoritma yang telah dibuat kemudian dimodelkan ke dalam bentuk *flow graph*, yang selanjutnya akan menentukan kasus uji yang akan diuji satu per satu.

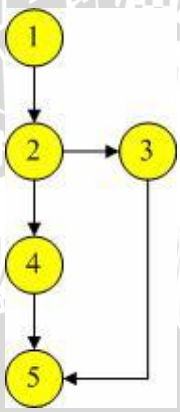
6.1.2.1 Pengujian Operasi Registrasi

Operasi registrasi digunakan untuk memasukkan data user baru yang dilakukan sendiri oleh user yang bersangkutan. Pemodelan operasi registrasi ditunjukkan pada gambar 6.1.

```
Pemodelan algoritma registrasi pada klas admin_kontroler
(Algoritma untuk memasukkan data  user ke dalam basis data)

BEGIN

DESCRIPTION
(1)    SHOW form_registrasi_view
       GET information FROM form
       CALL form_validation
(2)    IF form IS valid
(3)           CALL insert_anggota
       ELSE
(4)           SHOW error_message_view
(5)    END IF

END
```

Gambar 6.1 Pemodelan Operasi Registrasi pada Klas Admin_ Kontroler

Sumber: Pengujian



Gambar 6.2 Flow Graph Operasi Registrasi pada Klas Admin_ Kontroler

Sumber: Pengujian

Dari hasil pemodelan operasi registrasi pada klas admin_kontroler yang tertuang dalam *flow graph* pada Gambar 6.2, ditentukan jumlah kompleksitas siklomatis menggunakan persamaan V(G) = E – N + 2, dimana V(G) merupakan jumlah kompleksitas siklomatis, E merupakan jumlah sisi (garis penghubung antar *node*) dan N merupakan jumlah simpul (*node*).

V(G) = E – N + 2

= 5 – 5 + 2

= 2

Dari nilai kompleksitas siklomatis pada perhitungan di atas ditentukan 2 jalur independen yaitu:

Jalur 1: 1 – 2 – 3 – 5

Jalur 2: 1 – 2 – 4 – 5

Penentuan kasus uji (*test case*) untuk setiap jalur dan hasil eksekusi setiap jalur ditunjukkan pada Tabel 6.3.

Tabel 6.3 Kasus Uji untuk Pengujian Operasi Registrasi pada Klas Admin_ Kontroler

Jalur	Kasus Uji	Hasil yang diharapkan	Hasil yang didapatkan
1	Form diisi dengan data yang lengkap dan valid.	Data baru tersimpan dalam tabel anggota.	Data baru tersimpan dalam tabel anggota .
2	Form dibiarkan dalam keadaan kosong.	Sistem menampilkan pesan kesalahan.	Sistem menampilkan pesan kesalahan.

Sumber: Pengujian

6.1.2.2 Pengujian Operasi Tambah_Admin

Operasi tambah_admin digunakan untuk memasukkan data admin baru yang dilakukan oleh admin lain yang telah terdaftar pada s istem. Pemodelan operasi tambah_admin ditunjukkan pada gambar 6.3.

Dari hasil pemodelan operasi registrasi pada klas admin_kontroler yang tertuang dalam *flow graph* pada Gambar 6.4, ditentukan jumlah kompleksitas siklomatis sebagai berikut:

V(G) = E – N + 2

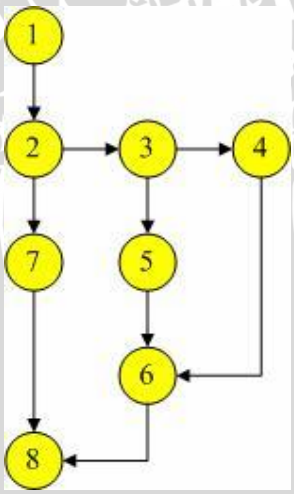
= 9 – 8 + 2

= 3

```
Pemodelan algoritma tambah_admin pada klas admin_kontroler
(Algoritma untuk memasukkan data  user ke dalam basis data)

BEGIN

DESCRIPTION
(1)    GET member_status
(2)    IF member_status IS admin
              SHOW form_tambah_admin_view
              GET information FROM form
              CALL form_validation
(3)           IF form IS valid
(4)                  CALL insert_admin
              ELSE
(5)                  SHOW error_message_view
(6)           END IF
       ELSE
(7)           SHOW error_message_view
(8)    ENDIF

END
```

Gambar 6.3 Pemodelan Operasi Tambah_Admin pada Klas Admin_ Kontroler

Sumber: Pengujian



Gambar 6.4 Flow Graph Operasi Tambah_Admin pada Klas Admin_Kontroler

Sumber: Pengujian

Dari nilai kompleksitas siklomatis pada perhitungan di atas ditentukan 3 jalur independen yaitu:

Jalur 1: 1 – 2 – 7 – 8

Jalur 2: 1 – 2 – 3 – 4 – 6 – 8

Jalur 3: 1 – 2 – 3 – 5 – 6 – 8

Penentuan kasus uji (*test case*) untuk setiap jalur dan hasil eksekusi setiap jalur ditunjukkan pada Tabel 6.4.

Tabel 6.4 Kasus Uji untuk Pengujian Operasi Tambah_Admin pada Klas Admin_Kontroler

Jalur	Kasus Uji	Hasil yang diharapkan	Hasil yang didapatkan
1	Tanpa *log in* masuk ke halaman tambah_admin	Sistem menampilkan pesan kesalahan.	Sistem menampilkan pesan kesalahan.
2	*Log in* sebagai admin kemudian mengisi form dengan data yang lengkap dan valid.	Data baru tersimpan dalam tabel anggota.	Data baru tersimpan dalam tabel anggota .
3	*Log in* sebagai admin kemudian membiarkan form dalam keadaan kosong.	Sistem menampilkan pesan kesalahan.	Sistem menampilkan pesan kesalahan.

Sumber: Pengujian

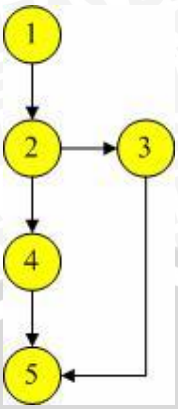
6.1.2.3 Pengujian Operasi Hapus Anggota

Operasi hapus_anggota digunakan untuk menghapus data anggota dari basis data. Pemodelan operasi hapus_anggota ditunjukkan pada gambar 6.5.

```
Operasi algoritma hapus_anggota pada klas admin_kontroler
(Algoritma untuk menghapus data user dari basis data)

BEGIN

DESCRIPTION
(1)    GET member_status
(2)    IF member_status IS admin
(3)           GET nomor_anggota
              CALL delete_anggota(nomor_anggota)
       ELSE
(4)           SHOW error_message_view
(5)    END IF

END
```

Gambar 6.5 Pemodelan Operasi Hapus_Anggota pada Klas Admin_Kontroler

Sumber: Pengujian



Gambar 6.6 Flow Graph Operasi Hapus_Anggota pada Klas Admin_Kontroler

Sumber: Pengujian

Dari hasil pemodelan operasi hapus_anggota pada klas admin_kontroler yang tertuang dalam *flow graph* pada Gambar 6.6, ditentukan jumlah kompleksitas siklomatis sebagai berikut:

V(G) = E – N + 2

= 5 – 5 + 2

= 2

Dari nilai kompleksitas siklomatis pada perhitungan di atas ditentukan 2 jalur independen yaitu:

Jalur 1: 1 – 2 – 3 – 5

Jalur 2: 1 – 2 – 4 – 5

Penentuan kasus uji (*test case*) untuk setiap jalur dan hasil eksekusi setiap jalur ditunjukkan pada Tabel 6.5.

Tabel 6.5 Kasus Uji untuk Pengujian Operasi Hapus_Anggota pada Klas Admin_Kontroler

Jalur	Kasus Uji	Hasil yang diharapkan	Hasil yang didapatkan
1	Log in sebagai admin kemudian menekan link hapus anggota.	Data hilang dari tabel anggota.	Data hilang dari tabel anggota.
2	Tanpa log in sebagai admin masuk ke halaman hapus anggota.	Sistem menampilkan pesan kesalahan.	Sistem menampilkan pesan kesalahan.

Sumber: Pengujian

6.1.2.4 Pengujian Operasi Kirim Pertanyaan

Operasi kirim_pertanyaan digunakan untuk menambah data pertanyaan baru ke dalam basis data. Pemodelan operasi kirim_pertanyaan ditunjukkan pada gambar 6.7.

```
Pemodelan algoritma kirim_pertanyaan pada klas pengiriman_kontroler
(Algoritma untuk memasukkan data user ke dalam basis data)

BEGIN

DESCRIPTION
(1)    GET logged_status
(2)    IF logged_status IS false
(3)           SHOW error_message_view
       ELSE
(4)           GET quota
(5)           IF quota IS 0
(6)                  SHOW error_message_view
              ELSE
                     SHOW form_kirim_pertanyaan_view
(7)                  CALL form_validation
(8)                  IF form IS valid
(9)                         CALL insert_pertanyaan
                     ELSE
(10)                        SHOW error_message_view
(11)                 END IF
(12)          END IF
(13)   END IF

END
```

Gambar 6.7 Pemodelan Operasi Kirim_Pertanyaan pada Klas Pengiriman_Kontroler

Sumber: Pengujian

Dari hasil pemodelan operasi kirim_pertanyaan pada klas pengiriman_kontroler yang tertuang dalam *flow graph* pada Gambar 6.8, ditentukan jumlah kompleksitas siklomatis sebagai berikut:

V(G) = E – N + 2

= 15 – 13 + 2

= 4

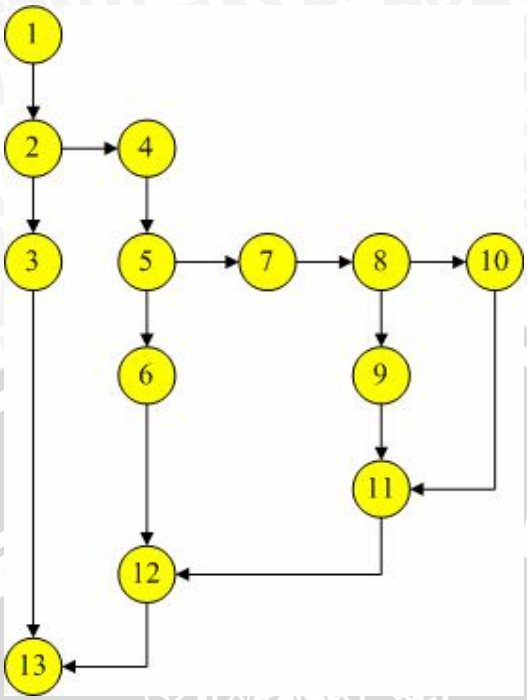
Dari nilai kompleksitas siklomatis pada perhitungan di atas ditentukan 4 jalur independen yaitu:

Jalur 1: 1 – 2 – 3 – 13

Jalur 2: 1 – 2 – 4 – 5 – 6 – 12 – 13

Jalur 3: 1 – 2 – 4 – 5 – 7 – 8 – 9 – 11 – 12 – 13

Jalur 4: 1 – 2 – 4 – 5 – 7 – 8 – 10 – 11 – 12 – 13



Gambar 6.8 Flow Graph Operasi Kirim_Pertanyaan pada Klas Pengiriman_ Kontroler

Sumber: Pengujian

Penentuan kasus uji (*test case*) untuk setiap jalur dan hasil eksekusi setiap jalur ditunjukkan pada Tabel 6.6.

Tabel 6.6 Kasus Uji untuk Pengujian Operasi Kirim_Pertanyaan pada Klas Pengiriman_Kontroler

Jalur	Kasus Uji	Hasil yang diharapkan	Hasil yang didapatkan
1	Tanpa log in masuk ke halaman kirim pertanyaan.	Sistem menampilkan pesan kesalahan.	Sistem menampilkan pesan kesalahan.
2	Log in kemudian masuk ke halaman kirim pertanyaan dengan status quota = 0.	Sistem menampilkan pesan kesalahan.	Sistem menampilkan pesan kesalahan.
3	Log in dengan status quota > 0 dan kemudian mengisi form dengan data yang lengkap dan valid.	Data baru tersimpan dalam tabel pertanyaan.	Data baru tersimpan dalam tabel pertanyaan.
4	Log in dengan status quota > 0 kemudian mengirim form dengan mengosongkan seluruh data isian yang diminta.	Sistem menampilkan pesan kesalahan.	Sistem menampilkan pesan kesalahan.

Sumber: Pengujian

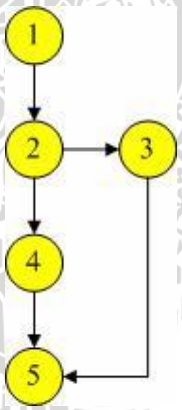
6.1.2.5 Pengujian Operasi Hapus Pertanyaan

Operasi hapus_pertanyaan digunakan untuk menghapus data pertanyaan dari basis data. Pemodelan operasi hapus_pertanyaan ditunjukkan pada gambar 6.9.

```
Pemodelan algoritma hapus_pertanyaan pada klas admin_kontroler
(Algoritma untuk menghapus data pertanyaan dari basis data)

BEGIN

DESCRIPTION
(1)    GET member_status
(2)    IF member_status IS admin
              GET nomor_pertanyaan
              CALL delete_jawaban(nomor_pertanyaan)
(3)           CALL delete_pertanyaan(nomor_pertanyaan)
       ELSE
(4)           SHOW error_message_view
(5)    END IF

END
```

Gambar 6.9 Pemodelan Operasi Hapus_Pertanyaan pada Klas Admin_Kontroler

Sumber: Pengujian



Gambar 6.10. Flow Graph Operasi Hapus_Pertanyaan pada Klas Admin_Kon troler

Sumber: Pengujian

Dari hasil pemodelan operasi hapus_pertanyaan pada klas admin_kontroler yang tertuang dalam *flow graph* pada Gambar 6.6, ditentukan jumlah kompleksitas siklomatis sebagai berikut:

V(G) = E – N + 2

= 5 – 5 + 2

= 2

Dari nilai kompleksitas siklomatis pada perhitungan di atas ditentukan 2 jalur independen yaitu:

Jalur 1: 1 – 2 – 3 – 5

Jalur 2: 1 – 2 – 4 – 5

Penentuan kasus uji (*test case*) untuk setiap jalur dan hasil eksekusi setiap jalur ditunjukkan pada Tabel 6.7.

Tabel 6.7 Kasus Uji untuk Pengujian Operasi Hapus_Pertanyaan pada Klas Admin_Kontroler

Jalur	Kasus Uji	Hasil yang diharapkan	Hasil yang didapatkan
1	Log in sebagai admin kemudian menekan link hapus pertanyaan.	Data hilang dari tabel pertanyaan.	Data hilang dari tabel pertanyaan.
2	Tanpa log in sebagai admin masuk ke halaman hapus pertanyaan.	Sistem menampilkan pesan kesalahan.	Sistem menampilkan pesan kesalahan.

Sumber: Pengujian

6.1.2.6 Pengujian Operasi Kirim Jawaban

Operasi kirim_jawaban digunakan untuk menambah data jawaban baru ke dalam basis data. Pemodelan operasi kirim_ jawaban ditunjukkan pada gambar 6.11.

```
Implementasi algoritma kirim_jawaban pada klas pengiriman_kontroler
(Algoritma untuk memasukkan data  jawaban ke dalam basis data)

BEGIN

DESCRIPTION
(1)    GET logged_status
(2)    IF logged_status IS false
(3)           SHOW error_message_view
       ELSE
              CALL duplicate_validation
(4)           IF status IS valid
                     GET nomor_pertanyaan
                     SHOW form_kirim_jawaban_view
                     CALL form_validation
(5)                  IF form IS valid
(6)                         CALL insert_jawaban(nomor_pertanyaan)
                     ELSE
(7)                         SHOW error_message_view
(8)                  END IF
              ELSE
(9)                  SHOW error_message_view
(10)          END IF
(11)   ENDIF

END
```

Gambar 6.11. Pemodelan Operasi Kirim_Jawaban pada Klas Pengiriman_Kontroler

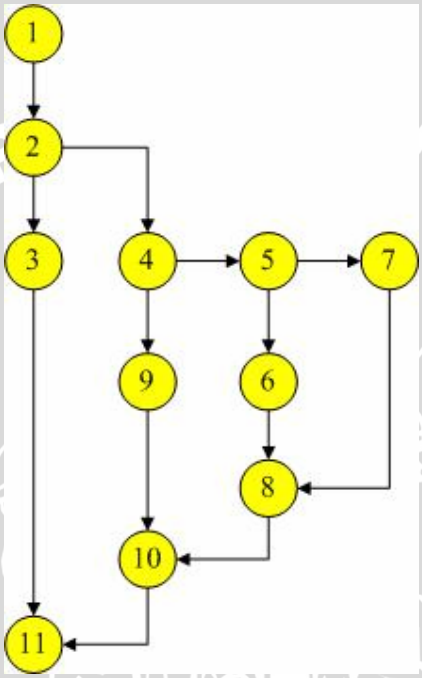
Sumber: Pengujian

Dari hasil pemodelan operasi kirim_jawban pada klas pengiriman_kontroler yang tertuang dalam *flow graph* pada Gambar 6.12, ditentukan jumlah kompleksitas siklomatis sebagai berikut:

V(G) = E – N + 2

= 13 – 11 + 2

= 4



Gambar 6.12. Flow Graph Operasi Kirim_Jawaban pada Klas Pengi riman_ Kontroler

Sumber: Pengujian

Dari nilai kompleksitas siklomatis pada perhitungan di atas ditentukan 4 jalur independen yaitu:

Jalur 1: 1 – 2 – 3 – 11

Jalur 2: 1 – 2 – 4 – 9 – 10 – 11

Jalur 3: 1 – 2 – 4 – 5 – 6 – 8 – 10 – 11

Jalur 4: 1 – 2 – 4 – 5 – 7 – 8 – 10 – 11

Penentuan kasus uji (*test case*) untuk setiap jalur dan hasil eksekusi setiap jalur ditunjukkan pada Tabel 6.8.

Tabel 6.8 Kasus Uji untuk Pengujian Operasi Kirim_Jawaban pada Klas Pengiriman_Kontroler

Jalur	Kasus Uji	Hasil yang diharapkan	Hasil yang didapatkan
1	Tanpa log in masuk ke halaman kirim jawaban.	Sistem menampilkan pesan kesalahan.	Sistem menampilkan pesan kesalahan.
2	Log in kemudian mengirim jawaban pada pertanyaan yang sudah pernah dijawab sebelumnya	Sistem menampilkan pesan kesalahan.	Sistem menampilkan pesan kesalahan.
3	Log in kemudian mengisi form kirim jawaban dengan data yang lengkap dan valid.	Data baru tersimpan dalam tabel jawaban.	Data baru tersimpan dalam tabel jawaban.
4	Log in kemudian mengirim form dengan mengosongkan seluruh data isian yang diminta.	Sistem menampilkan pesan kesalahan.	Sistem menampilkan pesan kesalahan.

Sumber: Pengujian

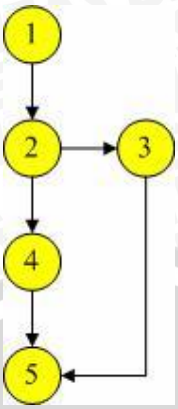
6.1.2.7 Pengujian Fungsi Hapus Jawaban

Operasi hapus_pertanyaan digunakan untuk menghapus data pertanyaan dari basis data. Pemodelan operasi hapus_pertanyaan ditunjukkan pada gambar 6.13.

```
Pemodelan algoritma hapus_jawaban pada klas admin_kontroler
(Algoritma untuk menghapus data jawaban dari basis data)

BEGIN

DESCRIPTION
(1)    GET member_status
(2)    IF member_status IS admin
              GET nomor_pertanyaan
(3)           CALL delete_jawaban(nomor_pertanyaan)
       ELSE
(4)           SHOW error_message_view
(5)    END IF

END
```

Gambar 6.13 Pemodelan Operasi Hapus_Jawaban pada Klas Admin_Kontroler

Sumber: Pengujian



Gambar 6.14 Flow Graph Operasi Hapus_Jawaban pada Klas Admin_Kontrole r

Sumber: Pengujian

Dari hasil pemodelan operasi hapus_jawaban pada klas admin_kontroler yang tertuang dalam *flow graph* pada Gambar 6.14, ditentukan jumlah kompleksitas siklomatis sebagai berikut:

V(G) = E – N + 2

= 5 – 5 + 2

= 2

Dari nilai kompleksitas siklomatis pada perhitungan di atas ditentukan 2 jalur independen yaitu:

Jalur 1: 1 – 2 – 3 – 5

Jalur 2: 1 – 2 – 4 – 5

Penentuan kasus uji (*test case*) untuk setiap jalur dan hasil eksekusi setiap jalur ditunjukkan pada Tabel 6.9.

Tabel 6.9 Kasus Uji untuk Pengujian Operasi Hapus_Pertanyaan pada Klas Admin_Kontroler

Jalur	Kasus Uji	Hasil yang diharapkan	Hasil yang didapatkan
1	Log in sebagai admin kemudian menekan link hapus jawaban.	Data hilang dari tabel jawaban.	Data hilang dari tabel jawaban.
2	Tanpa log in sebagai admin masuk ke halaman hapus jawaban.	Sistem menampilkan pesan kesalahan.	Sistem menampilkan pesan kesalahan.

Sumber: Pengujian

6.1.2.8 Analisis Pengujian Integrasi

Dari hasil pengujian integrasi terhadap operasi-operasi yang terdapat dalam klas pengiriman_kontroler dan admin_kontroler dapat diambil kesimpulan sistem perangkat

lunak telah berhasil mengintegrasikan klas-klas yang berhubungan untuk melakukan fungsi yang telah ditentukan sebelumnya.

6.1.3 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. Dikarenakan tiap kebutuhan fungsional dituangkan ke dalam suatu *use case*, maka proses ini dikenal pula dengan istilah pengujian *use case* (*use case testing*). Dalam melakukan pengujian validasi digunakan metode pengujian kotak hitam dan difokuskan untuk menemukan konformitas antara kinerja sistem dengan daftar kebutuhan. Objek dari pengujian validasi adalah item-item kebutuhan yang telah dirumuskan pada daftar kebutuhan pada Tabel 4.2. Nama kasus uji dan objek pengujian ditunjukkan pada Sub Bab 6.1.3.1 dan hasil pengujiannya ditunjukkan pada Sub Bab 6.1.3.2.

6.1.3.1 Pengujian Kebutuhan Fungsional

Untuk mengetahui kesesuaian antara kebutuhan dengan kinerja sistem, pada setiap kebutuhan fungsional dilakukan proses pengujian dengan masing-masing kasus uji.

1. Kasus Uji Registrasi

Nama Kasus Uji	:	Kasus Uji Registrasi
Objek Uji	:	Kebutuhan Fungsional 1 (F01)
Tujuan Pengujian	:	Pengujian dilakukan untuk memastikan bahwa aplikasi dapat memenuhi kebutuhan fungsional untuk menambahkan data anggota baru.
Prosedur Uji	:	Nilai-nilai data yang valid dimasukkan ke form kemudian menekan tombol untuk menyimpan data.
Hasil yang diharapkan	:	Sistem dapat melakukan validasi form, menyimpan data di basis data, dan menampilkan pesan proses berhasil dilakukan.

2. Kasus Uji Kegagalan Registrasi

Nama Kasus Uji	:	Kasus Uji Registrasi

Objek Uji	:	Aliran Alternatif Kebutuhan Fungsional 1 (F01)
Tujuan Pengujian	:	Pengujian dilakukan untuk memastikan bahwa aplikasi dapat menampilkan pesan kesalahan jika terdapat form isian yang masih kosong.
Prosedur Uji	:	Nilai-nilai data pada form dikosongkan kemudian menekan tombol untuk menyimpan data.
Hasil yang diharapkan	:	Sistem menampilkan pesan kesalahan pengisian form.

3. Kasus Uji Log In

Nama Kasus Uji	:	Kasus Uji Log In
Objek Uji	:	Kebutuhan Fungsional 2 (F02)
Tujuan Pengujian	:	Pengujian dilakukan untuk memastikan bahwa aplikasi dapat memenuhi kebutuhan fungsional untuk melakukan log in. Memasukkan nilai nick dan kata sandi.
Prosedur Uji	:	Memasukkan nilai nick: “djoko” dan kata sandi: “123456” kemudian menekan tombol Login.
Hasil yang diharapkan	:	Sistem melakukan validasi form, dan menampilkan halaman utama sebagai tanda bahwa proses berhasil dilakukan.

4. Kasus Uji Kegagalan Login

Nama Kasus Uji	:	Kasus Kegagalan Uji Login
Objek Uji	:	Aliran alternatif kebutuhan fungsional 2 (F02)
Tujuan Pengujian	:	Pengujian dilakukan untuk memastikan bahwa aplikasi dapat menampilkan pesan kesalahan jika user salah memasukkan nickname dan/atau kata sandi.
Prosedur Uji	:	Memasukkan nilai nickname: “xxx” dan kata sandi: “abc” kemudian menekan tombol Login.
Hasil yang diharapkan	:	Sistem melakukan validasi form, dan menampilkan pesan kesalahan.

5. Kasus Uji Log Out

Nama Kasus Uji	:	Kasus Uji Log Out
Objek Uji	:	Kebutuhan fungsional 3 (F03)

Tujuan Pengujian : Pengujian dilakukan untuk memastikan bahwa sistem telah memiliki fasilitas untuk menghentikan suatu sesi yang telah dibuat oleh user.

Prosedur Uji :
1. Log In sebagai member valid.
2. Menekan link Log Out untuk keluar dari situs.

Hasil yang diharapkan : Sistem menampilkan pesan pertanda proses berhasil dilakukan.

6. Kasus Uji Lihat Profil Member

Nama Kasus Uji : Kasus Uji Lihat Profil Member

Objek Uji : Kebutuhan fungsional 4 (F04)

Tujuan Pengujian : Pengujian dilakukan untuk memastikan bahwa aplikasi dapat menampilkan halaman detail user.

Prosedur Uji : Menekan link nama user 'djoko' di bagian pertanyaan.

Hasil yang diharapkan : Sistem menampilkan halaman detail tentang user dengan nickname djoko.

7. Kasus Uji Ubah Profil Diri

Nama Kasus Uji : Kasus Uji Ubah Profil Diri

Objek Uji : Kebutuhan fungsional 5 (F05)

Tujuan Pengujian : Pengujian dilakukan untuk memastikan bahwa aplikasi memiliki fasilitas untuk mengubah detail profil member dan berfungsi sesuai dengan yang diharapkan.

Prosedur Uji :
1. Log In sebagai member 'djoko'.
2. Menekan link menu member.
3. Menekan link ubah profil.
4. Mengubah email dengan nama djoko_tampan@ gmail.com dan diakhiri dengan menekan tombol kirim.
5. Menekan link nama member untuk melihat detail profil yang telah diubah.

Hasil yang diharapkan : Sistem menyimpan nilai data baru pada basis data, dan menampilkan pesan pertanda proses berhasil dilakukan.

8. Kasus Uji Kegagalan Ubah Profil Diri

Nama Kasus Uji	:	Kasus Uji Kegagalan Ubah Profil Diri
Objek Uji	:	Aliran Alternatif Kebutuhan Fungsional 5 (F05)
Tujuan Pengujian	:	Pengujian dilakukan untuk memastikan bahwa aplikasi dapat menampilkan pesan kesalahan jika terdapat form isian yang masih kosong.
Prosedur Uji	:	Nilai-nilai data pada form dikosongkan kemudian menekan tombol untuk menyimpan data.
Hasil yang diharapkan	:	Sistem menampilkan pesan kesalahan pengisian form.

9. Kasus Uji Ubah Kata Sandi

Nama Kasus Uji	:	Kasus Uji Ubah Kata Sandi
Objek Uji	:	Kebutuhan fungsional 8 (F08)
Tujuan Pengujian	:	Pengujian dilakukan untuk memastikan bahwa aplikasi memiliki fasilitas untuk mengubah kata sandi member dan berfungsi sesuai dengan yang diharapkan.
Prosedur Uji	:	1. Log in sebagai member djoko. 2. Masuk ke dalam halaman ubah kata sandi. 3. Mengubah kata sandi menjadi ‘sandibaru’. 4. Logout dari sistem. 5. Masuk ke dalam halaman log in. 6. Memasukkan nickname ‘djoko’ dan kata sandi baru yaitu ‘sandibaru’.
Hasil yang diharapkan	:	Sistem menampilkan notifikasi berhasilnya login.

10. Kasus Uji Kegagalan Ubah Kata Sandi

Nama Kasus Uji	:	Kasus Uji Kegagalan Ubah Kata Sandi
Objek Uji	:	Aliran Alternatif Kebutuhan Fungsional 8 (F08)
Tujuan Pengujian	:	Pengujian dilakukan untuk memastikan bahwa aplikasi dapat menampilkan pesan kesalahan jika terdapat form isian yang masih kosong.
Prosedur Uji	:	Nilai-nilai data pada form dikosongkan kemudian menekan tombol untuk menyimpan data.
Hasil yang diharapkan	:	Sistem menampilkan pesan kesalahan pengisian form.

11. Kasus Uji Kirim Pertanyaan

Nama Kasus Uji	:	Kasus Uji Kirim Pertanyaan
Objek Uji	:	Kebutuhan fungsional tujuh (F07)
Tujuan Pengujian	:	Pengujian dilakukan untuk memastikan bahwa aplikasi dapat menyimpan data pertanyaan baru ke dalam basis data.
Prosedur Uji	:	1. Log in sebagai ’djoko’. 2. Masuk halaman kirim petanyaan. 3. Memasukkan nilai-nilai data yang valid dan mengakhiri proses dengan menekan tombol Kirim. 4. Masuk ke halaman utama.
Hasil yang diharapkan	:	Sistem menampilkan halaman utama yang di dalamnya terdapat judul pertanyaan yang baru dikirimkan.

12. Kasus Uji Kegagalan Kirim Pertanyaan

Nama Kasus Uji	:	Kasus Uji Kegagalan Kirim Pertanyaan
Objek Uji	:	Aliran Alternatif Kebutuhan Fungsional Tujuh (F07)
Tujuan Pengujian	:	Pengujian dilakukan untuk memastikan bahwa aplikasi dapat menampilkan pesan kesalahan jika terdapat form isian yang masih kosong.
Prosedur Uji	:	Nilai-nilai data pada form dikosongkan kemudian menekan tombol untuk menyimpan data.
Hasil yang diharapkan	:	Sistem menampilkan pesan kesalahan pengisian form.

13. Kasus Uji Kirim Jawaban

Nama Kasus Uji	:	Kasus Uji Kirim Jawaban
Objek Uji	:	Kebutuhan fungsional delapan (F08)
Tujuan Pengujian	:	Pengujian dilakukan untuk memastikan bahwa aplikasi dapat menyimpan data jawaban baru ke dalam basis data.
Prosedur Uji	:	1. Log in sebagai ’djoko’. 2. Masuk halaman detail petanyaan terbaru. 3. Masuk halaman kirim jawaban.

4. Memasukkan nilai-nilai data yang valid dan mengakhiri proses dengan menekan tombol Kirim.
5. Masuk ke halaman utama.
6. Masuk ke halaman detail pertanyaan kembali.

Hasil yang diharapkan : Sistem dapat menampilkan halaman detail pertanyaan beserta dengan jawaban terbaru yang baru dikirimkan.

14. Kasus Uji Kegagalan Kirim Jawaban

Nama Kasus Uji : Kasus Uji Kegagalan Kirim Jawaban

Objek Uji : Aliran Alternatif Kebutuhan Fungsional Delapan (F08)

Tujuan Pengujian : Pengujian dilakukan untuk memastikan bahwa aplikasi dapat menampilkan pesan kesalahan jika terdapat form isian yang masih kosong.

Prosedur Uji : Nilai-nilai data pada form dikosongkan kemudian menekan tombol untuk menyimpan data.

Hasil yang diharapkan : Sistem menampilkan pesan kesalahan pengisian form.

15. Kasus Uji Kirim Pengaduan Penyalahgunaan

Nama Kasus Uji : Kasus Uji Kirim Pengaduan Penyalahgunaan

Objek Uji : Kebutuhan fungsional sembilan (F09)

Tujuan Pengujian : Pengujian dilakukan untuk memastikan bahwa aplikasi dapat menyimpan data pengaduan penyalahgunaan baru ke dalam basis data.

Prosedur Uji :
1. Log in sebagai ’admin01’.
2. Masuk halaman detail petanyaan terbaru.
3. Masuk halaman kirim pengaduan pertanyaan.
4. Memasukkan nilai-nilai data yang valid dan mengakhiri proses dengan menekan tombol Kirim.
5. Masuk ke halaman menu admin.
6. Masuk ke halaman detail pengaduan penyalahgunaan kembali.

Hasil yang diharapkan : Sistem dapat menampilkan halaman detail pengaduan penyalahgunaan beserta dengan data terbaru yang baru dikirimkan.

16. Kasus Uji Kegagalan Kirim Pengaduan Penyalahgunaan

Nama Kasus Uji	:	Kasus Uji Kegagalan Kirim Pengaduan Penyalahgunaan
Objek Uji	:	Aliran Alternatif Kebutuhan Fungsional Sembilan (F09)
Tujuan Pengujian	:	Pengujian dilakukan untuk memastikan bahwa aplikasi dapat menampilkan pesan kesalahan jika terdapat form isian yang masih kosong.
Prosedur Uji	:	Nilai-nilai data pada form dikosongkan kemudian menekan tombol untuk menyimpan data.
Hasil yang diharapkan	:	Sistem menampilkan pesan kesalahan pengisian form.

17. Kasus Uji Cari Pertanyaan

Nama Kasus Uji	:	Kasus Uji Cari Pertanyaan
Objek Uji	:	Kebutuhan fungsional sepuluh/10 (F10)
Tujuan Pengujian	:	Pengujian dilakukan untuk memastikan bahwa aplikasi dapat melakukan proses pencarian suatu topik berdasarkan kata kunci tertentu.
Prosedur Uji	:	1. Masuk ke halaman utama. 2. Memasukkan kata kunci ‘profesi’ pada form cari pertanyaan dan mengakhiri proses dengan menekan tombol Cari.
Hasil yang diharapkan	:	Sistem menampilkan halaman hasil pencarian dengan kata kunci terkait.

18. Kasus Uji Jelajah Kategori

Nama Kasus Uji	:	Kasus Uji Jelajah Kategori
Objek Uji	:	Kebutuhan fungsional sebelas (F11)
Tujuan Pengujian	:	Pengujian dilakukan untuk memastikan bahwa aplikasi dapat menampilkan halaman jelajah kategori beserta topik-topik di dalamnya.
Prosedur Uji	:	1. Memilih menu kategori 2. Memilih kategori ‘teknik’ 3. Memilih kategori ‘elektro’

Hasil yang diharapkan : Sistem menampilkan halaman detail kategori beserta judul-judul topik di dalamnya

19. Kasus Uji Kirim Rating

Nama Kasus Uji : Kasus Uji Kirim Rating

Objek Uji : Kebutuhan fungsional 12 (F12)

Tujuan Pengujian : Pengujian dilakukan untuk memastikan bahwa aplikasi dapat menyimpan data rating yang dikirimkan oleh member

Prosedur Uji : 1. Log in sebagai 'andre'
2. Masuk ke menu/halaman utama
3. Masuk ke detail pertanyaan terbaru
4. Menekan link kirim rating
5. Masuk kembali ke detail pertanyaan tersebut

Hasil yang diharapkan : Sistem dapat menampilkan halaman detail pertanyaan dengan jumlah rating terbaru.

20. Kasus Uji Kegagalan Kirim Rating

Nama Kasus Uji : Kasus Uji Kegagalan Kirim Rating

Objek Uji : Aliran Alternatif Kebutuhan Fungsional Dua Belas (F12)

Tujuan Pengujian : Pengujian dilakukan untuk memastikan bahwa aplikasi dapat menampilkan pesan kesalahan jika terjadi pengiriman rating oleh pengguna tamu.

Prosedur Uji : Menekan link kirim rating dengan status sebagai tamu.

Hasil yang diharapkan : Sistem menampilkan pesan kesalahan pengisian form.

21. Kasus Uji Tambah Super User

Nama Kasus Uji : Kasus Uji Tambah Super User

Objek Uji : Kebutuhan fungsional 14 (F14)

Tujuan Pengujian : Pengujian dilakukan untuk memastikan bahwa aplikasi dapat menyimpan data super user baru baik sweeper maupun admin.

Prosedur Uji : 1. Log in sebagai admin01

2. Masuk ke halaman tambah admin
3. Mengisi form tambah admin dengan data yang valid dan mengakhiri proses dengan menekan tombol Simpan.
4. Keluar (Log out) dari situs
5. Log in sebagai admin yang baru ditambahkan

Hasil yang diharapkan : Sistem menampilkan halaman notifikasi keberhasilan proses log in.

22. Kasus Uji Kegagalan Tambah Super User

Nama Kasus Uji : Kasus Uji Kegagalan Tambah Super User

Objek Uji : Aliran Alternatif Kebutuhan Fungsional Empat Belas (F14)

Tujuan Pengujian : Pengujian dilakukan untuk memastikan bahwa aplikasi dapat menampilkan pesan kesalahan jika terdapat form isian yang masih kosong.

Prosedur Uji : Nilai-nilai data pada form dikosongkan kemudian menekan tombol untuk menyimpan data.

Hasil yang diharapkan : Sistem menampilkan pesan kesalahan pengisian form.

23. Kasus Uji Hapus Member

Nama Kasus Uji : Kasus Uji Hapus Member

Objek Uji : Kebutuhan fungsional 14 (F14)

Tujuan Pengujian : Pengujian dilakukan untuk memastikan bahwa aplikasi dapat melakukan proses penghapusan data member dari basis data sistem.

Prosedur Uji :
1. Log in sebagai admin01
2. Masuk ke halaman daftar anggota
3. Menghapus satu anggota dengan poin kecil
4. Masuk kembali ke halaman daftar anggota

Hasil yang diharapkan : Sistem menampilkan halaman daftar anggota dan nama anggota yang telah dihapus sudah tidak berada di antara daftar tersebut.

24. Kasus Uji Hapus Pertanyaan

Nama Kasus Uji	:	Kasus Uji Hapus Pertanyaan
Objek Uji	:	Kebutuhan fungsional 15 (F15)
Tujuan Pengujian	:	Pengujian dilakukan untuk memastikan bahwa aplikasi dapat melakukan proses penghapusan data pertanyaan dari basis data sistem.
Prosedur Uji	:	1. Log in sebagai admin01 2. Masuk ke halaman daftar pertanyaan 3. Menghapus satu pertanyaan dengan poin pengaduan besar 4. Masuk kembali ke halaman daftar pertanyaan
Hasil yang diharapkan	:	Sistem menampilkan halaman daftar pertanyaan dan data pertanyaan yang baru dihapus tidak dapat lagi ditemukan dalam daftar tersebut.

25. Kasus Uji Hapus Jawaban

Nama Kasus Uji	:	Kasus Uji Hapus Jawaban
Objek Uji	:	Kebutuhan fungsional 16 (F16)
Tujuan Pengujian	:	Pengujian dilakukan untuk memastikan bahwa aplikasi dapat melakukan proses penghapusan data jawaban dari basis data sistem.
Prosedur Uji	:	1. Log in sebagai admin01 2. Masuk ke halaman daftar pertanyaan 3. Masuk ke salah satu pertanyaan 4. Menghapus satu jawaban dengan poin pengaduan besar 5. Masuk kembali ke halaman daftar pertanyaan
Hasil yang diharapkan	:	Sistem menampilkan halaman detail pertanyaan dan data jawaban yang baru dihapus tidak dapat lagi ditemukan dalam halaman tersebut.

26. Kasus Uji Penggunaan Sistem Poin

Nama Kasus Uji	:	Kasus Uji Penggunaan Sistem Poin
Objek Uji	:	Kebutuhan non-fungsional satu (N01)

Tujuan Pengujian	:	Pengujian dilakukan untuk memastikan bahwa aplikasi telah mengimplementasikan sistem penambahan poin sebagai penyemangat aktifitas penggunaan aplikasi.
Prosedur Uji	:	1. Log in sebagai djoko 2. Masuk ke halaman daftar pertanyaan 3. Masuk ke detail pertanyaan terbaru 4. Masuk ke form kirim jawaban 5. Mengisi form dengan data yang valid
Hasil yang diharapkan	:	Sistem menampilkan halaman notifikasi keberhasilan dengan nilai poin yang lebih besar pada bagian status.

27. Kasus Uji Penggunaan PHP dan MySQL

Nama Kasus Uji	:	Kasus Uji PHP dan MySQL
Objek Uji	:	Kebutuhan non-fungsional dua (N02)
Tujuan Pengujian	:	Pengujian dilakukan untuk memastikan bahwa aplikasi dikembangkan dengan menggunakan bahasa pemrograman PHP dan sistem basis data MySQL.
Prosedur Uji	:	1. Membuka halaman phpinfo() 2. Membuka halaman phpmyadmin
Hasil yang diharapkan	:	1. Sistem menampilkan halaman phpinfo() yang berisi status-status berkaitan dengan sistem PHP yang digunakan. 2. Sistem menampilkan halaman phpmyadmin dengan keterangan status MySQL di dalamnya.

28. Kasus Uji Koneksi Client-Server via Internet

Nama Kasus Uji	:	Kasus Uji Koneksi Client-Server via Internet
Objek Uji	:	Kebutuhan non-fungsional tiga (N03)
Tujuan Pengujian	:	Pengujian dilakukan untuk memastikan bahwa koneksi dengan server dapat dilakukan.
Prosedur Uji	:	1. Melihat alamat server pada halaman status web server 2. Melakukan ping menggunakan perintah-perintah koneksi berbasis windows.

Hasil yang diharapkan : Sistem menampilkan hasil ping yang menyatakan bahwa koneksi sedang berlangsung.

6.1.3.3. Hasil Pengujian Validasi

Dari kasus uji yang telah dilaksanakan sesuai dengan prosedur pengujian pada sub pokok bahasan 6.1.3.1, didapatkan hasil se perti ditunjukkan pada Tabel 6. 10.

Tabel 6.10 Hasil Pengujian Validasi

No.	Kasus Uji	Hasil yang diharapkan	Hasil yang didapatkan	Status
1	Kasus Uji Registrasi	Sistem melakukan validasi form, menyimpan data di basis data, dan menampilkan pesan pertanda proses berhasil dilakukan.	Sistem melakukan validasi form, menyimpan data di basis data, dan menampilkan pesan pertanda proses berhasil dilakukan.	Valid
2	Kasus Uji Kegagalan Registrasi	Sistem melakukan validasi form, dan menampilkan pesan kesalahan.	Sistem melakukan validasi form, dan menampilkan pesan kesalahan.	Valid
3	Kasus Uji Log In	Sistem melakukan validasi form, dan menampilkan halaman utama sebagai tanda bahwa proses berhasil dilakukan.	Sistem melakukan validasi form, dan menampilkan halaman utama sebagai tanda bahwa proses berhasil dilakukan.	Valid
4	Kasus Uji Kegagalan Log In	Sistem melakukan validasi form, dan menampilkan pesan kesalahan.	Sistem melakukan validasi form, dan menampilkan pesan kesalahan.	Valid
5	Kasus Uji Log Out	Sistem menampilkan pesan pertanda proses berhasil dilakukan.	Sistem menampilkan pesan pertanda proses berhasil dilakukan.	Valid
6	Kasus Uji Lihat Profil Member	Sistem menampilkan halaman detail profil member.	Sistem menampilkan halaman detail profil member.	Valid
7	Kasus Uji Ubah Profil Diri	Sistem melakukan validasi form, menyimpan nilai data baru pada basis data, dan menampilkan pesan pertanda proses berhasil dilakukan.	Sistem melakukan validasi form, menyimpan nilai data baru pada basis data, dan menampilkan pesan pertanda proses berhasil dilakukan.	Valid
8	Kasus Uji Kegagalan Ubah Profil Diri	Sistem melakukan validasi form, dan kemudian menampilkan pesan kesalahan.	Sistem melakukan validasi form, dan kemudian menampilkan pesan kesalahan.	Valid
9	Kasus Uji Ubah Kata Sandi	Sistem melakukan validasi form, menyimpan nilai data baru pada basis data, dan menampilkan pesan pertanda proses berhasil dilakukan.	Sistem melakukan validasi form, menyimpan nilai data baru pada basis data, dan menampilkan pesan pertanda proses berhasil dilakukan.	Valid
10	Kasus Uji Kegagalan Ubah Kata Sandi	Sistem melakukan validasi form, dan kemudian menampilkan pesan kesalahan.	Sistem melakukan validasi form, dan kemudian menampilkan pesan kesalahan.	Valid
11	Kasus Uji Kirim Pertanyaan	Sistem melakukan validasi form, menyimpan nilai data baru pada basis data, dan menampilkan pesan pertanda proses berhasil dilakukan.	Sistem melakukan validasi form, menyimpan nilai data baru pada basis data, dan menampilkan pesan pertanda proses berhasil dilakukan.	Valid

No.	Kasus Uji	Hasil yang diharapkan	Hasil yang didapatkan	Status
12	Kasus Uji Kegagalan Kirim Pertanyaan	Sistem melakukan validasi form, dan kemudian menampilkan pesan kesalahan.	Sistem melakukan validasi form, dan kemudian menampilkan pesan kesalahan.	Valid
13	Kasus Uji Kirim Jawaban	Sistem melakukan validasi form, menyimpan nilai data baru pada basis data, dan menampilkan pesan pertanda proses berhasil dilakukan.	Sistem melakukan validasi form, menyimpan nilai data baru pada basis data, dan menampilkan pesan pertanda proses berhasil dilakukan.	Valid
14	Kasus Uji Kegagalan Kirim Jawaban	Sistem melakukan validasi form, dan kemudian menampilkan pesan kesalahan.	Sistem melakukan validasi form, dan kemudian menampilkan pesan kesalahan.	Valid
15	Kasus Uji Kirim Pengaduan Penyalahgunaan	Sistem melakukan validasi form, menyimpan nilai data baru pada basis data, dan menampilkan pesan pertanda proses berhasil dilakukan.	Sistem melakukan validasi form, menyimpan nilai data baru pada basis data, dan menampilkan pesan pertanda proses berhasil dilakukan.	Valid
16	Kasus Uji Kegagalan Kirim Pengaduan Penyalahgunaan	Sistem melakukan validasi form, dan kemudian menampilkan pesan kesalahan.	Sistem melakukan validasi form, dan kemudian menampilkan pesan kesalahan.	Valid
17	Kasus Uji Cari Pertanyaan	Sistem menampilkan halaman hasil pencarian dengan kata kunci terkait.	Sistem menampilkan halaman hasil pencarian dengan kata kunci terkait.	Valid
18	Kasus Uji Jelajah Kategori	Sistem menampilkan halaman detail kategori beserta judul-judul topik di dalamnya.	Sistem menampilkan halaman detail kategori beserta judul-judul topik di dalamnya.	Valid
19	Kasus Uji Kirim Rating	Sistem melakukan validasi (member dan duplikasi), kemudian menampilkan halaman detail pertanyaan dengan jumlah rating terbaru.	Sistem melakukan validasi (member dan duplikasi), kemudian menampilkan halaman detail pertanyaan dengan jumlah rating terbaru.	Valid
20	Kasus Uji Kegagalan Kirim Rating	Sistem melakukan validasi (member dan duplikasi), dan kemudian menampilkan pesan kesalahan.	Sistem melakukan validasi (member dan duplikasi), dan kemudian menampilkan pesan kesalahan.	Valid
21	Kasus Uji Tambah *Super User*	Sistem melakukan validasi, menyimpan data baru pada basis data, dan menampilkan pesan pertanda proses berhasil dilakukan.	Sistem melakukan validasi, menyimpan data baru pada basis data, dan menampilkan pesan pertanda proses berhasil dilakukan.	Valid
22	Kasus Uji Kegagalan Tambah *Super User*	Sistem melakukan validasi, dan kemudian menampilkan pesan kesalahan.	Sistem melakukan validasi, dan kemudian menampilkan pesan kesalahan.	Valid
23	Kasus Uji Hapus Member	Sistem melakukan validasi, menghapus data dari basis data, dan kemudian menampilkan pesan keberhasilan.	Sistem melakukan validasi, menghapus data dari basis data, dan kemudian menampilkan pesan keberhasilan.	Valid
24	Kasus Uji Hapus Pertanyaan	Sistem melakukan validasi, menghapus data dari basis data, dan kemudian menampilkan pesan keberhasilan.	Sistem melakukan validasi, menghapus data dari basis data, dan kemudian menampilkan pesan keberhasilan.	Valid
25	Kasus Uji Hapus Jawaban	Sistem melakukan validasi, menghapus data dari basis data, dan kemudian menampilkan pesan keberhasilan.	Sistem melakukan validasi, menghapus data dari basis data, dan kemudian menampilkan pesan keberhasilan.	Valid

No.	Kasus Uji	Hasil yang diharapkan	Hasil yang didapatkan	Status
26	Kasus Uji Penggunaan Sistem Poin	Sistem menampilkan jumlah poin member di bagian status member.	Sistem menampilkan jumlah poin member di bagian status member.	Valid
27	Kasus Uji Penggunaan PHP dan MySQL	Sistem menampilkan halaman phpinfo dan phpmyadmin.	Sistem menampilkan halaman phpinfo dan phpmyadmin.	Valid
28	Kasus Uji Koneksi Client-Server	Sistem menampilkan hasil ping yang menyatakan bahwa koneksi sedang berlangsung.	Sistem menampilkan hasil ping yang menyatakan bahwa koneksi sedang berlangsung.	Valid

Sumber: Pengujian

6.1.3.4 Analisis Pengujian Validasi

Dari hasil pengujian untuk setiap kebutuhan yang telah ditetapkan pada Tabel 4.2, dapat disimpulkan bahwa sistem telah memenuhi konformitas antara sistem yang dibangun dengan daftar kebutuhan. Kebutuhan fungsional yang telah ditetapkan telah berhasil diimplementasikan dan diuji sesuai dengan skenario perangkat lunak yang telah ditetapkan pada tahap perancangan perangkat lunak.

6.1.4 Pengujian Performansi Koneksi

Pengujian performansi dilakukan untuk mengetahui performa baik performa *local hosting* maupun performa *remote hosting*. Pengujian performansi koneksi terdiri dari pengujian koneksi basis data dan *web server*, pengujian waktu akses *query*, dan pengujian performansi *web server*.

Pengujian performansi koneksi ini menggunakan spesifikasi dan konfigurasi komputer *client* dan sekaligus *local hosting* seperti yang dituliskan dalam tabel 6.11.

Tabel 6.11 Spesifikasi Komputer *Client/Local hosting* untuk Implementasi

Spesifikasi Komputer *Local hosting*	
Prosessor	Intel(R) Pentium(R) 4 CPU 1.70 GHz
Memori (RAM)	256 MB DDR
Hard Disk	Seagate ST380011A, kapasitas 80 GB, 7200 RPM
Motherboard	Asus P4B533
Sistem Operasi	Windows XP SP2 versi 2002
Bahasa Pemrograman	PHP 5.2.8
Basis Data (*Database*)	MySQL Version 5.1.30
Web Server	Apache Server 2.2.11
Browsers	Opera 10.10, Mozilla Firefox 3.5, Internet Explorer 6

Sumber: Implementasi

Spesifikasi komputer *remote hosting* yang digunakan pada pengujian implementasi aplikasi ditampilkan pada tabel 6.12.

Tabel 6.12 Spesifikasi Komputer *Remote hosting* untuk Implementasi

Spesifikasi Komputer *Remote hosting*	
Sistem Operasi	*Unix-like*
Bahasa Pemrograman	PHP 5.2.*
Basis Data (*Database*)	MySQL Version 5.0.81-community
Web Server	Apache Server 2.2.13 (Unix)
Kapasitas Penyimpanan	1500 MB
Alamat Web	http://km-ub.comyr.com
Alamat IP	216.108.235.104

Sumber: Implementasi

6.1.4.1 Pengujian Koneksi *Server* dan Basis Data

A. Tujuan

- Pengujian dilakukan untuk mengetahui keberhasilan koneksi baik pada *local hosting* maupun *remote hosting*.

B. Prosedur Pengujian

1. *Local hosting*

- Menjalankan program Apache berserta *service* MySQL di dalamnya.
- Mengakses alamat web *local hosting* menggunakan *browser* untuk menguji koneksi *server*.
- Mengakses phpmyadmin pada browser dan membuka basis data kmarket untuk menguji koneksi basis data.

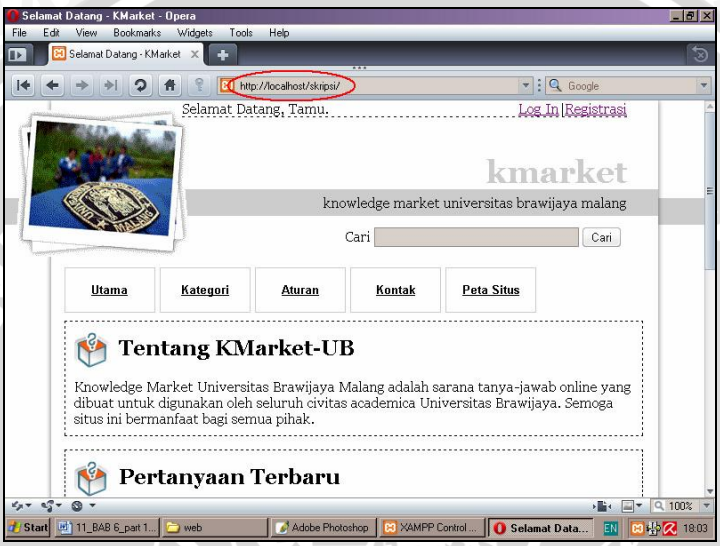
2. *Remote hosting*

- Mengakses alamat web *remote hosting* menggunakan *browser* untuk menguji koneksi *server*.
- Mengakses phpmyadmin pada browser dan membuka basis data kmarket untuk menguji koneksi basis data.

C. Proses Pengujian

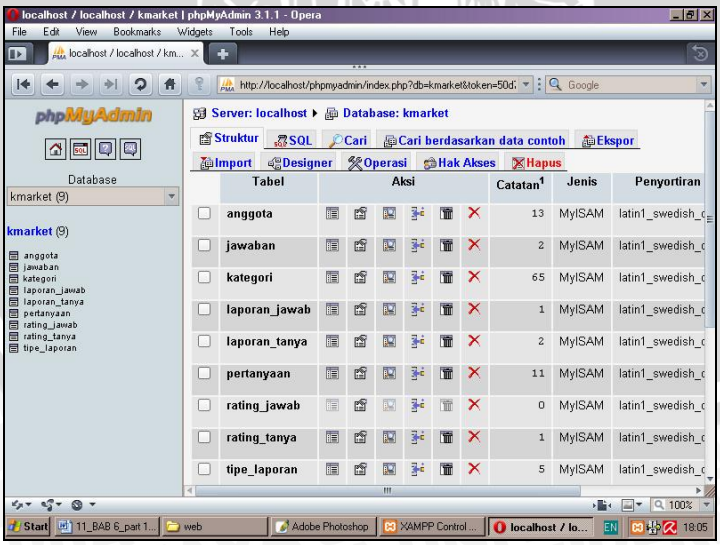
1. *Local hosting*

- Setelah alamat dimasukkan ke dalam *browser*, *browser* menampilkan halaman utama dari situs seperti ditampilkan pada gambar 6.14.
- Setelah phpmyadmin diakses menggunakan browser, browser menampilkan daftar tabel yang terdapat dalam basis data kmarket seperti ditampilkan pada gambar 6.15.



Gambar 6.14 Hasil Tampilan Pengujian Koneksi Local Hosting

Sumber: Pengujian

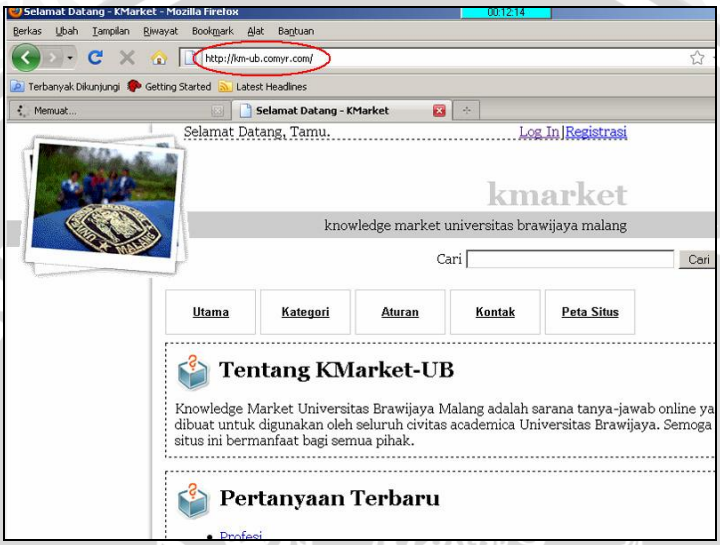


Gambar 6.15 Hasil Tampilan Pengujian Basis Data Local Hosting

Sumber: Pengujian

2. *Remote hosting*

- Setelah alamat dimasukkan ke dalam *browser*, *browser* menampilkan halaman utama dari situs seperti ditampilkan pada gambar 6.16.
- Setelah phpmyadmin diakses menggunakan *browser*, *browser* menampilkan daftar tabel yang terdapat dalam basis data kmarket yang serupa dengan tampilan di pengujian *local hosting*.



Gambar 6.16 Hasil Tampilan Pengujian Koneksi Remote Hosting

Sumber: Pengujian

D. Hasil Pengujian dan Analisis

- Aplikasi dapat dijalankan pada jaringan komputer yang menggunakan protokol TCP/IP.
- Koneksi *server* basis data MySQL pada kedua *server* berhasil dibangun.

6.1.4.2 Pengujian Waktu Akses *Query*

A. Tujuan

- Pengujian dilakukan untuk mengetahui rata-rata waktu yang dibutuhkan untuk melakukan *query* pada basis data baik pada *local hosting* maupun *remote hosting*.
- Pengujian dilakukan untuk mendapatkan perbandingan waktu *query* yang dilakukan terhadap kedua *server* yang digunakan.

B. Prosedur Pengujian

1. *Local hosting*

- Menjalankan program Apache berserta *service* MySQL di dalamnya.
- Mengakses alamat web *local hosting* menggunakan *browser*.
- Mengakses halaman utama untuk mendapatkan waktu akses *query* pertama.
- Mengakses halaman daftar pertanyaan untuk mendapatkan waktu akses *query* kedua.
- Mengakses kembali halaman utama untuk mendapatkan waktu akses *query* ketiga dan dilakukan seterusnya hingga mendapatkan dua puluh d ata.

2. *Remote hosting*

- Mengakses alamat web *remote hosting* menggunakan *browser*.
- Mengakses halaman utama untuk mendapatkan waktu akses *query* pertama.
- Mengakses halaman daftar pertanyaan untuk mendapatkan waktu akses *query* kedua.
- Mengakses kembali halaman utama untuk mendapatkan waktu akses *query* ketiga dan dilakukan seterusnya hingga mendapatkan dua puluh data.

C. Proses Pengujian

- Hasil pengujian waktu akses *query* terhadap kedua halaman menghasilkan data yang beragam yang mana ditunjukkan pada Tabel 6.13 dan Tabel 6.14.

Tabel 6.13 Hasil Pengujian Waktu Akses *Query* pada Halaman Utama

Langkah Pengujian	*Local hosting* (detik)	*Remote hosting* (detik)
1	0,1526	0,0273
2	0,1742	0,0303
3	0,1543	0,0307
4	0,1794	0,0272
5	0,1742	0,0282
6	0,1558	0,0256
7	0,1555	0,0278
8	0,1724	0,0267
9	0,1758	0,0264

Langkah Pengujian	*Local hosting* (detik)	*Remote hosting* (detik)
10	0,1561	0,0292
Data Rata-Rata	0,16503	0,02794

Sumber: Pengujian

Tabel 6.14 Hasil Pengujian Waktu Akses *Query* pada Halaman Pertanyaan

Langkah Pengujian	*Local hosting* (detik)	*Remote hosting* (detik)
1	0,1776	0,0273
2	0,1626	0,0295
3	0,1643	0,0269
4	0,1608	0,032
5	0,1622	0,0289
6	0,1613	0,0319
7	0,1583	0,027
8	0,1603	0,0274
9	0,1647	0,0272
10	0,1612	0,0276
Data Rata-Rata	0,16333	0,02857

Sumber: Pengujian

D. Hasil Pengujian dan Analisis

- Basis data kmarket pada komputer *server* dapat menangani permintaan *query* dari komputer *client* melalui jaringan komputer dengan waktu akses *query* rata-rata adalah 0,17 detik untuk *local hosting* dan 0,03 detik untuk *remote hosting*. Hasil pengujian *local hosting* lebih besar dibandingkan dengan remote hosting diperkirakan karena terdapat beban aplikasi yang besar pada *local hosting* dikarenakan sistem operasi menangani fungsi *client* dan fungsi *server* secara bersamaan.

6.1.4.3 Pengujian Performansi *Webserver*

A. Tujuan

- Pengujian performansi *web server* dilakukan untuk mengetahui kinerja dari kemampuan *web server* untuk menangani *request* dari *client*. Untuk mengetahui kinerja dari kemampuan *web server*, digunakan alat bantu *Apache Benchmark* (ab.exe). Aplikasi ini merupakan *tools* bawaan dari sistem *web server Apache*.

B. Prosedur Pengujian

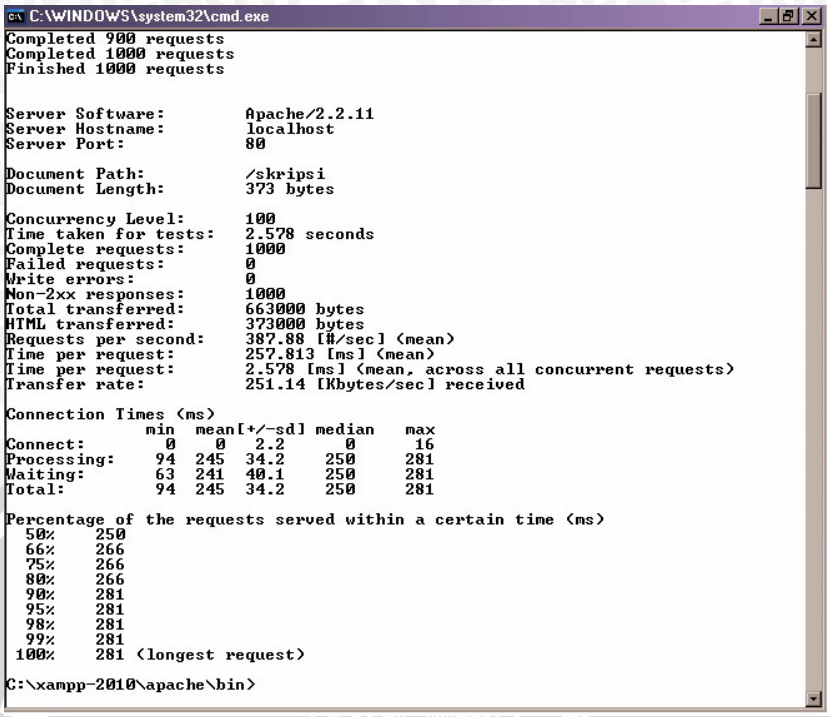
1. *Local hosting*
   - Menjalankan program Apache berserta *service* MySQL di dalamnya.
   - *Window* Command Prompt dijalankan dengan memberikan perintah sebagai berikut:
     Start | Run... | Open: cmd.exe
   - Aplikasi *Apache Benchmark* (ab.exe) dijalankan dengan memberikan perintah sebagai berikut:
     C:\Documents and Settings\ >ab -n 1000 -c 100 http://localhost/skripsi
2. *Remote hosting*

- *Window* Command Prompt dijalankan dengan memberikan perintah sebagai berikut:
  Start | Run... | Open: cmd.exe
- Aplikasi *Apache Benchmark* (ab.exe) dijalankan dengan memberikan perintah sebagai berikut:
  C:\Documents and Settings\ >ab -n 1000 -c 100 http://km-ub.comyr.com

C. Proses Pengujian

1. *Local hosting*

- Perintah “ab -n 1000 -c 100 http://localhost/skripsi” pada pengujian *local hosting* memberikan hasil yang ditunjukkan dalam Gambar 6.17. Maksud dari perintah ini adalah aplikasi *Apache Benchmark* melakukan pengujian pada server Apache dengan memberikan total 1000 permintaan (*request*) dengan 100 permintaan dilakukan secara bersamaan.

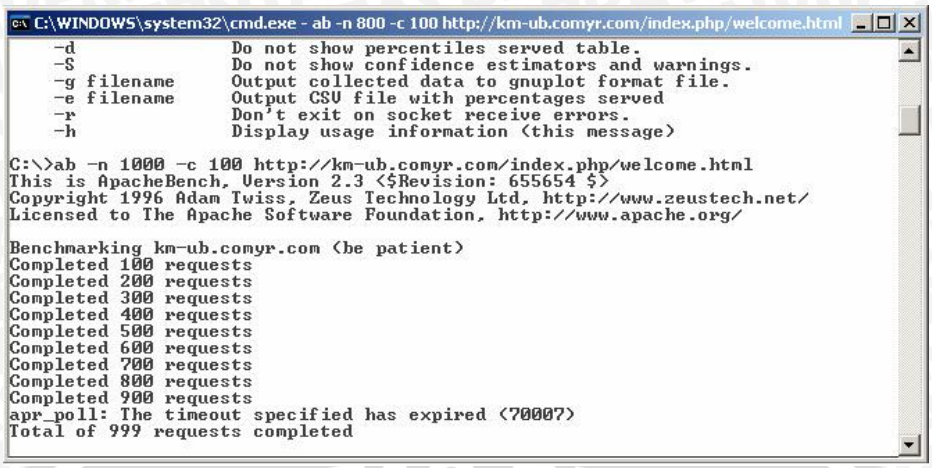


Gambar 6.17 Hasil Pengujian Performansi Server Apache Local Hosting

Sumber: Pengujian

2. *Remote hosting*

- Perintah “ab -n 1000 -c 100 http://km-ub.comyr.com” pada pengujian *remote hosting* memberikan hasil yang ditunjukkan dalam Gambar 6.18. Maksud dari perintah ini adalah aplikasi *Apache Benchmark* melakukan pengujian pada server Apache dengan memberikan total 1000 permintaan (*request*) dengan 100 permintaan dilakukan secara bersamaan.
- Dari hasil tersebut, ternyata proses pengujian gagal menghasilkan data seperti yang diharapkan, sehingga perlu dilakukan pengujian ulang dengan request yang lebih sedikit yaitu dengan menggunakan perintah: “ab -n 500 -c 100 http://km-ub.comyr.com”. Perintah ini memberikan hasil seperti ditunjukkan gambar 6.19.



Gambar 6.18 Hasil Pengujian Performansi Server Apache Remote Hosting 1000 Permintaan

Sumber: Pengujian

Gambar 6.19 Hasil Pengujian Performansi Server Apache Remote Hosting dengan 500 Permintaan

Sumber: Pengujian

D. Hasil Pengujian dan Analisis

- Hasil pengujian menunjukkan bahwa server Apache yang digunakan pada *local hosting* dapat bekerja dengan baik. Hal ini ditunjukkan oleh parameter waktu akses yang kecil yaitu rata-rata sebesar 245 ms.

- Hasil pengujian *remote hosting* menunjukkan bahwa server tidak dapat melayani 1000 permintaan dengan 100 permintaan yang dilakukan secara bersamaan. Namun dengan melakukan perubahan jumlah permintaan menjadi 500 permintaan, server dapat menangani permintaan namun dengan performa yang kurang baik yang ditunjukkan dengan data waktu akses yang besar yaitu rata - rata sebesar 30,845 s.

6.2 Analisis Kebutuhan Perangkat Lunak

Dari hasil pengujian dan analisis terhadap setiap kebutuhan perangkat lunak didapatkan hasil yang sesuai dengan keluaran yang diharapkan. Sistem perangkat lunak yang dibangun telah berhasil memenuhi seluruh kebutuhan perangkat lunak pada Tabel 4.2.

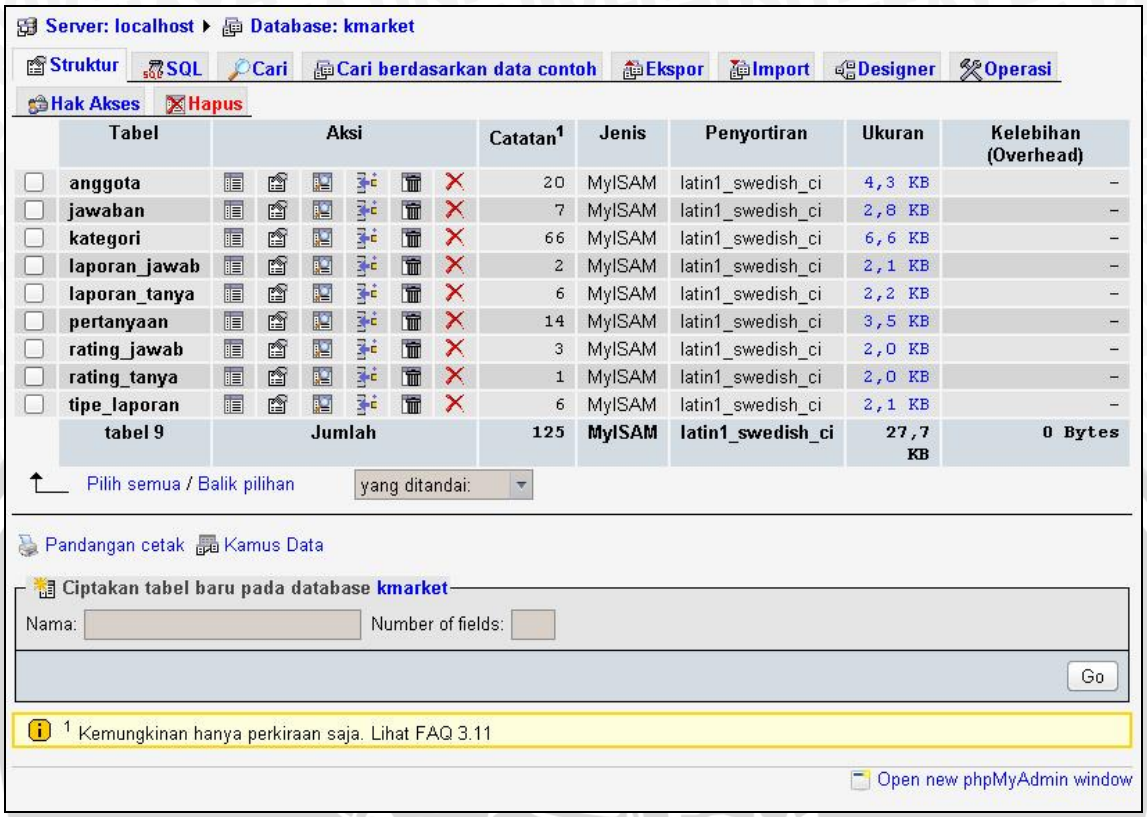
Berdasarkan pengujian unit didapatkan hasil pengujian yang sesuai dengan yang diharapkan. Pengujian unit dilakukan untuk menguji bagian terkecil dari sistem yang dalam aplikasi ini diwakili oleh operasi-operasi yang terdapat dalam klas model. Parameter kesuksesan ditentukan dengan membandingkan hasil yang diharapkan dengan keluaran yang dihasilkan.

Pengujian integrasi dengan menggabungkan operasi-operasi yang telah diuji pada tahap pengujian unit untuk dapat melakukan sebuah *use case* telah berhasil dilaksanakan. Pengujian integrasi menggunakan teknik *white box testing*. Objek pengujian integrasi dalam aplikasi ini diwakili oleh operasi-operasi yang terdapat dalam klas kontroler. Parameter kesuksesan ditentukan dengan membandingkan hasil yang diharapkan dengan keluaran yang dihasilkan.

Berdasarkan hasil pengujian validasi dapat dibuktikan bahwa antarmuka yang dibangun telah berhasil menjalankan seluruh fungsionalitas sistem yang telah ditetapkan pada Tabel 4.2. Parameter kesuksesan ditentukan dengan menguji setiap *use case* berdasarkan prosedur uji yang diturunkan dari *use case scenario*. Analisis kebutuhan fungsional dan non fungsional dilakukan dengan membandingkan keluaran yang dihasilkan dengan hasil yang diharapkan. Dari hasil pengujian validasi dapat disimpulkan bahwa sistem perangkat lunak yang dibangun telah berhasil memenuhi kebutuhan perangkat lunak yang telah ditetapkan pada tahap analisis kebutuhan.

6.3 Penanggulangan Bertambahnya Beban Data

Setelah aplikasi berhasil dibuat, perlu dilakukan analisis terhadap basis data untuk menanggulangi kondisi bertambahnya data di masa akan datang.



Gambar 6.20 Data-Data Perkiraan Penggunaan Ruang Penyimpanan Data

Dengan menggunakan bantuan aplikasi phpmyadmin, data -data perkiraan ruang penyimpanan yang digunakan aplikasi pada *hard disk* dapat diperhitungkan. Adapun data-data perkiraan utama diuraikan pada tabel 6.15.

Tabel 6.15 Data Acuan Penggunaan Ruang Penyimpanan

No.	Nama Tabel	Jumlah Data	Penggunaan Ruang Penyimpanan
1	anggota	20	4,3 kB
2	pertanyaan	14	3,5 kB
3	jawaban	7	2,8 kB

Dari data-data tersebut dapat diproyeksikan besar ruang penyimpanan dengan data yang berkembang. Adapun data-data hasil proyeksi diuraikan pada tabel 6.16.

Tabel 6.16 Data Perkiraan Penggunaan Ruang Penyimpanan

No.	Nama Tabel	Jumlah Data	Penggunaan Ruang Penyimpanan
1	anggota	500	107,5 kB
2	pertanyaan	1000	250 kB
3	jawaban	5000	2 MB

Dari data tersebut didapatkan jumlah total data utama sebesar 3 MB. Dengan demikian dapat diperkirakan bahwa penggunaan *hard disk* dengan ukuran yang secara umum digunakan untuk komputer personal (80 GB) sudah dapat menangani adanya perkembangan jumlah data yang akan terjadi, namun apabila perkembangan volume data sudah dirasa mengurangi unjuk kerja sistem, maka dapat dilakukan proses pengarsipan basis data (*database archiving*) dengan metode yang lebih mutakhir.

Beberapa metode pengarsipan basis data yang dapat diambil [OLS-09], antara lain:

- Menyimpan data pada *file* yang terpisah

  Dalam metode ini data yang dianggap kurang vital disimpan pada media penyimpanan lain dalam bentuk *file* teks (atau tipe data lainnya) yang dapat digunakan kembali di waktu yang akan datang, sedangkan data yang berada di dalam basis data dihapus untuk mengurangi beban pada sistem.

- Menyimpan data pada basis data cadangan (*parallel database*)

  Metode ini serupa dengan metode pertama, hanya berbeda dalam media penyimpanan yang berupa basis data dengan struktur sama yang berada pada media penyimpan lain.

BAB VII
PENUTUP

7.1 Kesimpulan

Berdasarkan pada hasil pengujian dan analisis penelitian yang dilakukan, dapat diambil kesimpulan sebagai berikut:

1. Berdasarkan hasil pengujian unit, pengujian integrasi, pengujian validasi, dan kemudian melakukan analisis untuk setiap kebutuhan (fungsional dan non fungsional), diperoleh hasil yang sesuai dengan spesifikasi dan fungsional sistem yang telah ditetapkan.
2. Perangkat lunak *Internet-Based Knowledge Market* telah berhasil dibuat yang mana berfungsi untuk menampung pertanyaan dan jawaban anggota situs.
3. Antarmuka dengan konsep *multiple views* telah berhasil diimplementasikan dengan menggunakan bantuan *framework CodeIgniter*.
4. Basis data relasional dibentuk berdasarkan kebutuhan yang digambarkan pada diagram *use case*. Terdapat 9 (sembilan) tabel di dalam basis data yang dibuat untuk menyimpan berbagai data dalam aplikasi.
5. Dari hasil pengujian waktu akses *query* database diperoleh hasil bahwa komputer *server* dapat menangani permintaan *query* dari komputer *client* melalui jaringan komputer, dengan waktu akses *query* rata-rata sebesar 0,17 detik untuk *local hosting* dan 0,03 detik untuk *remote hosting*.
6. Kemampuan *server* yang digunakan pada aplikasi ini dapat memenuhi 100 permintaan dari pengguna yang dilakukan secara bersamaan.

7.2 Saran

Saran yang dapat diberikan untuk pengembangan aplikasi Internet-Based Knowledge Market antara lain:

1. Perangkat lunak dapat digunakan untuk melengkapi fungsionalitas forum diskusi yang telah ada di lingkungan Universitas Brawijaya Malang.

2. Dengan menerapkan *framework* pada pengembangan perangkat lunak, tampilan aplikasi dapat dikembangkan untuk memperoleh tampilan yang lebih menarik . Penggunaan AJAX juga dapat dilakukan sehingga dapat mengurangi jumlah halaman serta membuat sistem tampak lebih dinamis.

DAFTAR PUSTAKA

[AND-06] Andersson, Eve & Greenspun, Philip. 2006. *Software Engineering for Internet Applications*. Cambridge: MIT Press.

[BOW-03] Bowen, R. & Coar, K. 2003. *Apache Cookbook.* Sebastopol: O'Reilly.

[COL-08] Coleman, David & Levine, Stewart. 2008. *Collaboration 2.0*. Cupertino: Happy About.

[CON-04] Converse, Tim & Park, Joyce & Morgan, Clark. 2004. *PHP5 and MySQL Bible*. Indianapolis: Wiley Publishing.

[CRO-07] Cross, Michael. 2007. *Developer's Guide to Web Application Security.* Rockland: Syngress Publishing.

[KUR-05a] Kurniawan, Tri A. 2005. *Handout Kuliah Rekayasa Perangkat Lunak : Analisis Kebutuhan Perangkat Lunak*. Malang: Universitas Brawijaya.

[KUR-05b] Kurniawan, Tri A. 2005. *Handout Kuliah Rekayasa Perangkat Lunak : Pengujian*. Malang: Universitas Bra-wijaya.

[LAV-06] Lavin, Peter. 2006. *Object-Oriented PHP: Concepts, Techniques, and Codes*. San Francisco: No Starch Press.

[MAI-07] Maier, Ronald. 2007. *Knowledge Management Systems: Informatio n and Communication Technologies for Knowledge Management, Third Edition* . New York: Springer-Verlag.

[NOR-04] Norman, Matthew. 2004. *Database Design Manual: using MySQL for Windows*. London: Springer-Verlag.

[OLI-06] Oliver, Dick & Morrison, Michael. 2006. *Teach Yourself HTML and CSS in 24 Hours.* Indianapolis: Sams Publishing.

[OLS-09] Olson, Jack. 2009. *Database Archiving*. Burlington: Morgan Kaufman.

[PRE-94] Pree, W. 1994. *Meta patterns - a means for capturing the essentials of reusable object-oriented design*. Bologna: Springer-Verlag.

[PRE-01] Pressman, Roger S. 2001. *Software engineering: a practitioner's approach 5th ed*. New York: McGraw-Hill.

[SKY-01] Skyrme, David J. 2001. *Online Knowledge Markets: How Do They Work?* Diakses dari: http://www.skyrme.com/ insights/28kmkt.htm . Tanggal akses: 23 September 2008.

[SPE-03] Speed, Tim & Ellis, Juanita. 2003. *Internet Security: A Jumpstart for Systems Administrators and IT Managers.* Burlington: Digital Press.

[SUL-06] Sullivan, Danny. 2006. *Look Out Wikipedia, Here Comes Yahoo Answers!* Alamat: http://searchenginewatch.com/showPage.html?page=3612046. Diakses tanggal: 10 September 2008.

[WEI-03] Weisfeld, Matt. 2003. *Object-Oriented Thought Process, The Second Edition*. Indianapolis: Sams Publishing.

[YOR-05] York, Richard. 2005. *Beginning CSS: Cascading Style Sheets for Web Design*. Indianapolis: Wiley Publishing, Inc.