Sistem Layanan Informasi Lowongan Kerja Berbasis RSS Melalui Media SMS dengan Menggunakan Protokol SMPP

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Teknik

Disusun oleh :

PUTRI ELFA MAS`UDIA

NIM. 0510630081

DEPARTEMEN PENDIDIKAN NASIONAL

UNIVERSITAS BRAWIJAYA

FAKULTAS TEKNIK

MALANG

2010

Sistem Layanan Informasi Lowongan Kerja Berbasis RSS Melalui Media SMS dengan Menggunakan Protokol SMPP

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan

memperoleh gelar Sarjana Teknik

Disusun oleh:

PUTRI ELFA MAS`UDIA

NIM. 0510630081

Telah diperiksa dan disetujui oleh

Dosen Pembimbing:

Suprapto, ST., MT.
NIP. 19710727 199603 1 001

Himawat Aryadita, ST., MT., M.Sc.
NIP. 19801018 200801 1 003

Sistem Layanan Informasi Lowongan Kerja Berbasis RSS Melalui Media SMS dengan Menggunakan Protokol SMPP

Disusun oleh :

PUTRI ELFA MAS`UDIA
NIM. 0510630081-63

Skripsi ini telah diuji dan dinyatakan lulus pada
tanggal **26 Januari 2010**

Majelis Penguji :

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

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

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 syukur ke hadirat Allah SWT karena berkat rahmat dan hidayahNya-lah penulis dapat menyelesaikan Tugas Akhir yang berjudul: "**Sistem Layanan Informasi Lowongan Kerja Berbasis RSS Melalui Media SMS Menggunakan protokol SMPP".** Hanya kepada-Nya kita menyembah dan memohon. Teriring doa keselamatan untuk Rasulullah Muhammad SAW, keluarga, sahabat serta seluruh umatnya. Tugas Akhir ini disusun untuk memenuhi sebagian persyaratan memperoleh gelar Sarjana Teknik di Jurusan Teknik Elektro Konsentrasi Teknik Informatika dan Komputer Fakultas Teknik Universitas Brawijaya Malang.

Melalui kesempatan ini, penulis ingin menyampaikan rasa hormat dan terima kasih penulis yang sebesar besarnya kepada semua pihak yang telah memberikan bantuan baik material maupun non-material sehingga Tugas Akhir ini dapat terselesaikan dengan baik dan tepat pada waktunya. Oleh karena itu, pada kesempatan ini penulis ingin menyampaikan rasa hormat dan terima kasih penulis kepada:

1. Bapak Rudy Yuwono, ST., M.Sc. dan Bapak Muhammad Aziz Muslim, ST., MT., Ph.D. selaku Ketua dan Sekretaris Jurusan Teknik Elektro serta segenap Bapak/Ibu Dosen, Staff Administrasi dan Perpustakaan Jurusan Teknik Elektro Fakultas Teknik Universitas Brawijaya
2. Bapak Suprapto, ST., MT. dan Bapak Himawat Aryadita, ST., MT., M.Sc selaku pembimbing penulis dalam menyelesaikan tugas akhir ini.
3. Kedua Orang Tua, Mertua, Suami, dan seluruh keluarga yang senantiasa tiada henti hentinya memberikan do'a dan dukungan demi terselesainya tugas akhir ini.
4. Seluruh Dosen Teknik Elektro Unibraw atas kesediaan membagi ilmunya kepada penulis.
5. Semua Asisten, Ka. Lab, serta Laboran dari Laboratorium Komputasi dan Jaringan yang telah memberikan banyak bantuan dan dukungan dalam menyelesaikan tugas akhir ini.
6. Teman-temanku Teknik Elektro Unibraw khususnya angkatan 2005, terima kasih atas segala bantuannya selama menjadi mahasiswa.

7. Semua pihak yang tidak dapat penulis sebutkan satu per satu yang terlibat baik secara langsung maupun tidak langsung demi terselesaikannya tugas akhir ini.

Hanya doa yang bisa penulis berikan semoga Allah SWT memberikan pahala serta balasan kebaikan yang berlipat.

Penulis menyadari bahwa tugas akhir ini masih banyak kekurangan dan masih jauh dari sempurna. Untuk itu, saran dan kritik yang membangun sangat penulis harapkan. Semoga tugas akhir ini membawa manfaat bagi penyusun maupun pihak lain yang menggunakannya

Malang, Januari 2010

Penulis

ABSTRAK

PUTRI ELFA MAS`UDIA. 2010. : Sistem Layanan Informasi Lowongan Kerja Berbasis RSS Melalui Media SMS Menggunakan Protokol SMPP. Skripsi Jurusan Teknik Elektro, Fakultas Teknik, Universitas Brawijaya. Dosen Pembimbing: Suprapto, ST., MT. dan Himawat Aryadita, ST., MT.

Baik *internet* maupun *handphone* saat ini bukan lagi merupakan suatu yang langka, bahkan telah menjadi kebutuhan bagi masyarakat. *handphone* bukan hanya sebagai media komunikasi saja, tetapi telah berkembang dengan berbagai macam fungsi dan kegunaan. Perusahaan dapat memanfaatkan *internet* untuk menyebarkan informasi lowongan pekerjaan. Dan pihak pencari kerja dengan mudah dapat mendapatkan informasi kerja tersebut dengan melakukan akses ke *internet.*

Pada skripsi ini akan dibuat sebuah rancangan sistem yang memanfaatkan dua teknologi sekaligus yaitu *handphone* dan *internet* untuk membuat aplikasi layanan *broadcast* sms berisi informasi lowongan kerja. Pada sistem ini, akan dibangun sebuah web untuk menampung informasi lowongan kerja yang didapatkan dari RSS web-web yang menyediakan informasi lowongan kerja atau dengan memasukkan secara manual informasi lowongan kerja yang didapat selain dari RSS web lain tersebut. Informasi tersebut akan dikirimkan ke semua nomor *handphone* member dengan menggunakan Protokol SMPP.

Sistem aplikasi ini dikembangkan berbasis web dengan menggunakan arsitektur MVC *(Model View Controller).* Aplikasi ini dikembangkan dengan teknologi Java sebagai bahasa pemrograman, Struts sebagai *framework* MVC dan Hibernate sebagai *framework* ORM *(Object Relaional Mapping).* Dalam proses pengembangan dan pengujian, digunakan SMSC Simulator sebagai pengganti sistem SMSC yang disediakan oleh *provider* seluler.

Pengujian aplikasi dilakukan dengan pengujian *E-R Diagram*, pengujian koneksi *database*, pengujian unit, pengujian integrasi, dan pengujian validasi. Hasil pengujian menunjukkan bahwa aplikasi yang dikembangkan ini valid sesuai kebutuhan yang diperoleh dari hasil analisis kebutuhan

Kata Kunci : Web, SMS, RSS, MVC*(Model View Controller)*, SMPP.

DAFTAR ISI

Halaman

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR ALGORITMA

BAB I
PENDAHULUAN

1.1. Latar Belakang

Perkembangan teknologi komunikasi dan informasi pada era globalisasi ini berjalan sangat pesat. Termasuk dalam kategori piranti teknologi komunikasi dan informasi tersebut adalah *internet* dan *handphone*. Baik *internet* maupun *handphone* saat ini bukan lagi merupakan suatu yang langka, bahkan telah menjadi kebutuhan bagi masyarakat. *handphone* bukan hanya sebagai media komunikasi saja, tetapi telah berkembang dengan berbagai macam fungsi dan kegunaan.

Perusahaan dan para lulusan akademik merupakan sebagian dari segmen masyarakat yang juga memperoleh dampak akibat perkembangan *internet*. Perusahaan dan para alumni dari institusi pendidikan merupakan dua pihak yang saling memiliki relasi ketergantungan. Perusahaan berperan sebagai pihak penyedia lapangan kerja, dan alumni institusi pendidikan sebagai pihak pencari kerja. Perusahaan dapat memanfaatkan *internet* untuk menyebarkan informasi lowongan pekerjaan. Dan pihak pencari kerja dengan mudah dapat mendapatkan informasi kerja tersebut dengan melakukan akses ke *internet*.

Akan tetapi perlu diakui bahwa masih terdapat keterbatasan dalam hal akses ke *internet*. Terutama bagi masyarakat pedesaan dan area-area yang belum masuk jaringan *internet*. Pertimbangan jarak dan biaya kerap kali menjadi hambatan sebelum memutuskan untuk mengakses *internet*. Sedangkan *handphone* telah menawarkan kemudahan dalam hal akses dan biaya yang relatif lebih murah [SMI-04:52].

Dari latar belakang tersebut, maka akan dibuat sebuah rancangan sistem yang memanfaatkan dua teknologi sekaligus yaitu *handphone* dan *internet* untuk membuat aplikasi layanan *broadcast* sms berisi informasi lowongan kerja. Pada sistem ini, akan dibangun sebuah web sederhana untuk menampung informasi lowongan kerja yang didapatkan dari RSS web-web yang menyediakan informasi lowongan kerja atau dengan memasukkan secara manual informasi lowongan kerja yang didapat selain dari RSS web lain tersebut. Informasi tersebut akan

dikirimkan ke semua nomor *handphone* member dengan menggunakan Protokol SMPP.

Short Message Peer-to-peer Protocol, adalah merupakan protokol standar dalam industri telekomunikasi yang dikhususkan untuk pertukaran pesan singkat atau SMS antar SMSC. Dengan menggunakan koneksi SMPP (*Short Message Peer to Peer*, waktu pengiriman mencapai 30 SMS per detik. [SSS-04]

1.2. Rumusan Masalah

Berdasarkan pada permasalahan yang telah dijelaskan pada bagian latar belakang, maka rumusan masalah dapat disusun sebagai berikut :

1. Analisis terhadap kebutuhan-kebutuhan yang diperlukan untuk membangun sebuah sistem.
2. Membangun sebuah web untuk mengumpulkan dan menyimpan informasi RSS tentang lowongan pekerjaan dari web-web lain.
3. Merancang sistem *database* untuk menyimpan data RSS dan administrasi member.
4. Membuat software implementasi untuk aplikasi sistem layanan informasi lowongan kerja.
5. Menguji software implementasi yang telah dibuat.

1.3. Batasan Masalah

Pembatasan masalah yang diajukan dalam penyusunan skripsi ini adalah sebagai berikut:

1. Informasi lowongan kerja didapatkan dari membaca RSS situs lowongan pekerjaan lain dan dengan memasukkan secara manual info lowongan yang didapatkan selain dari informasi RSS.
2. Pembahasan difokuskan pada perancangan dan pembuatan sistem informasi untuk mengumpulkan informasi lowongan kerja dan mengirimkan ke *handphone* member.
3. Aplikasi yang digunakan untuk mengirimkan SMS menggunakan Protokol SMPP
4. Bahasa pemrograman yang digunakan adalah JAVA dan DBMS yang digunakan adalah MySQL.

5. Simulasi yang dibuat hanya pengiriman data sampai ke SMSC Simulator bukan ke *handphone* member.

1.4. Tujuan

Tujuan tugas akhir ini adalah mengembangkan aplikasi yang bisa diimplementasikan pada institusi pendidikan, sebagai infrastruktur yang memberikan fasilitas kemudahan dalam memberikan informasi *update* lowongan kerja bagi para alumninya

1.5. Manfaat

Manfaat yang diharapkan dapat diperoleh dalam pengerjaan tugas akhir ini adalah dapat membantu dan memudahkan para pencari kerja untuk mendapatkan informasi lowongan kerja yang *uptodate*.

1.6. Sistematika Pembahasan

Sistematika pembahasan yang digunakan dalam menyusun laporan skripsi ini adalah:

BAB I Pendahuluan

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

BAB II Dasar Teori

Menjelaskan dasar teori yang digunakan dalam mengembangkan sistem, diantaranya Aplikasi Web Berbasis Java, basis data MySQL, RSS, Protokol SMPP, pemodelan dengan UML, teknik pengujian dan strategi pengujian.

BAB III Metodologi

Pada bab ini dibahas metodologi penelitian yang digunakan untuk mengerjakan tugas akhir ini. Diantaranya studi literatur terhadap dasar teori, pengumpuan data, metode untuk analisis kebutuhan (*requirement analysis*), perancangan sistem, implementasi, pengujian sistem, serta pengambilan kesimpulan dan saran.

BAB IV Perancangan

Pada bab ini dibahas proses perancangan sistem aplikasi. Dimulai dari analisis kebutuhan sistem dan pemodelan perancangan tersebut.

BAB V Implementasi

Pada bab ini dibahas proses implementasi sistem dari hasil perancangan yang telah dilakukan pada bab sebelumnya

BAB VI Pengujian dan Analisis Sistem

Membahas strategi pengujian, teknik pengujian, kasus uji, serta hasil pengujian.

BAB VII Penutup

Membuat kesimpulan dan saran.

BAB II
DASAR TEORI

Untuk menunjang dalam pengembangan aplikasi, dilakukan kajian terhadap dasar teori yang relefan. Dasar teori tersebut diantaranya adalah teori tentang aplikasi berbasis web (yang meliputi Java Servlet, JSP, Struts Framework, dan MySQL), RSS, komunikasi seluler dan protokol SMPP.

1.1. Aplikasi Berbasis Web

Aplikasi dimana pengguna mengakses atau berinteraksi dengan aplikasi tersebut melalui *web browser* dikenal sebagai *web based application* (aplikasi berbasis web). *Web browser* mengambil data yang akan ditampilkan dari suatu *web server*. Data tersebut dikirimkan dalam suatu standar *hypertext* (HTML) melalui protokol HTTP. Aplikasi berbasis web bisa melibatkan penggunaan database untuk menyimpan data yang terlibat dalam aplikasi tersebut.

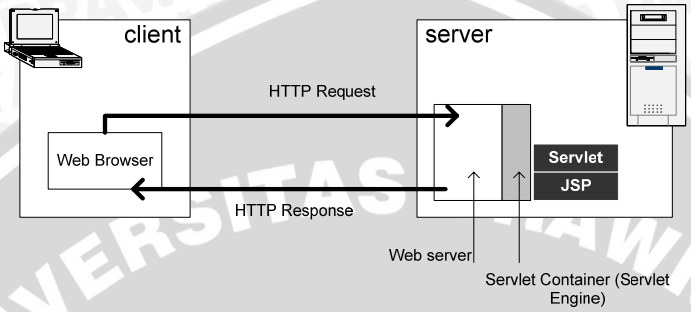
1.1.1. Java Servlet

Sebuah servlet berfungsi untuk memperluas fungsionalitas sebuah server (server web, server aplikasi, server HTTP). Servlet adalah program Java yang diintegrasikan di dalam *Web-server* untuk melakukan fungsi-fungsi *server-side*. Fungsi *server-side* ini dijalankan untuk menanggapi permintaan dari *client* (berupa *web-browser*). Server yang dapat menjalankan servlet disebut dengan *servlet-container* atau *servlet-engine*. Servlet dapat menggantikan fungsi CGI. Servlet memiliki beberapa keunggulan dibanding CGI, yaitu pemakaian memori lebih efisien, multi-platform dan *protocol –independent* [WIJ-07:18]. Untuk meminta layanan dari servlet dapat memakai sembarang bahasa pemrograman. Yang penting adalah server mengerti bahasa dan permintaan tersebut. Dengan demikian sebuah layanan servlet dapat diminta dengan mengetikkan URL di dalam web-browser atau bisa juga memakai sembarang bahasa pemrograman. Layanan servlet dapat juga diminta dari dalam sebuah halaman HTML.

Dalam Java Servlet API yang tersedia sekarang kita bisa memakai servlet generic, atau umumnya dalam praktik kita memakai servlet yang berkomunikasi memakai protokol HTTP.

Setelah menerima permintaan, servlet akan mengolah permintaan tersebut pada *server-side*. Hasilnya akan dikirimkan kepada *client*, yaitu konten

yang biasanya berupa halaman web. Dengan demikian servlet mampu memperluas fungsi server yang outputnya statis menjadi server yang outputnya dapat berupa konten dinamis. Dinamis dalam hal ini berarti bahwa permintaan yang berbeda dari *client* yang berbeda akan menghasilkan konten yang berbeda pula.



Gambar 2.1 Ilustrasi *Request* dan *response* antara *client* dan servlet
Sumber : [WIJ-07:18]

Gambar 2.1 menunjukkan ilustrasi sebuah *web-browser* yang meminta layanan kepada servlet. *Request* dikirimkan melalui protokol HTTP. Setelah diterima oleh *web server*, *request* akan diserahkan kepada server yang menjalankan servlet *(servlet-container atau servlet-engine)*.

Servlet yang dituju akan dijalankan dan menghasilkan *response* berupa halaman HTML. Respon akan dikirim kepada *client* oleh web server. Oleh *web-browser*, halaman HTML ini akan ditampilkan supaya dapat dibaca oleh pemakai. Dari gambar telihat di belakang servlet *container* terdapat komponen web berupa servlet dan JSP. JSP ketika dijalankan akan dikompilasi menjadi servlet. Servlet akan dijalankan oleh *servlet-container*, sedangkan JSP akan dijalankan oleh JSP *container*, meskipun *JSP container* secara internal boleh juga dianggap sebagai *servlet container* karena JSP dikompilasi terlebih dahulu menjadi servlet [WIJ-07:19].

1.1.2. JSP (*Java Server Pages*)

JSP merupakan suatu teknologi web berbasis bahasa pemrograman Java dan berjalan di platform Java. JSP merupakan bagian dari platform J2EE *(Java 2 Enterprise Edition)*. JSP sangat sesuai dan tangguh untuk menangani presentasi di web. Sedangkan J2EE merupakan platform Java untuk pengembangan sistem aplikasi enterprise dengan dukungan API *(Application Programming Interface)*

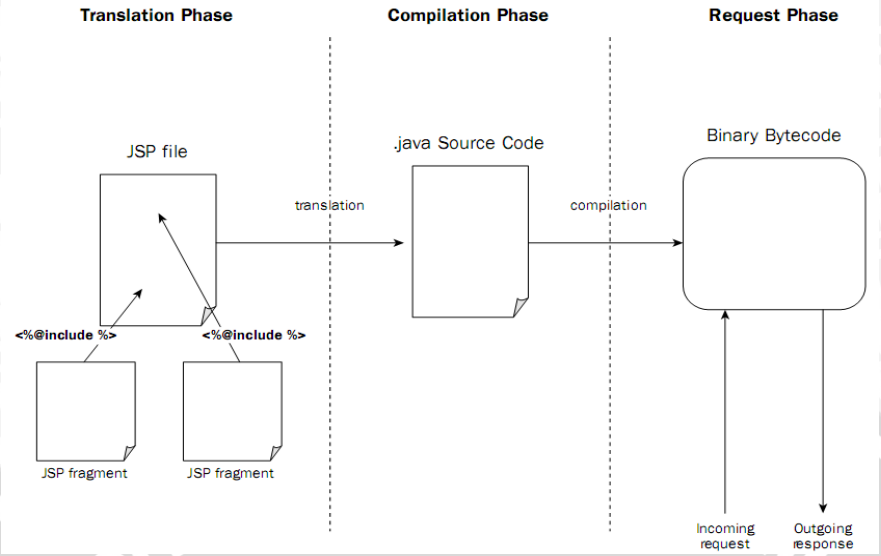
yang lengkap dan portabilitas serta memberikan sarana untuk membuat suatu aplikasi yang memisahkan antara logika bisnis, presentasi dan data. JSP memerlukan *Java Virtual Machine* (JVM) agar dapat berjalan, sehingga mengisyaratkan keharusan menginstal JVM di server tempat JSP dijalankan. Selain JVM, JSP juga memerlukan server yang disebut *Web Container* .

JSP merupakan perluasan dari teknologi servlet. Tujuan dari JSP adalah untuk lebih menyederhanakan penulisan servlet. JSP sendiri pada akhirnya sebelum dijalankan oleh server, akan dikompilasi terlebih dulu menjadi servlet, meskipun proses ini tidak terlihat.

JSP dan servlet dapat dipakai bersama-sama dalam sebuah aplikasi web. Perbedaan utama antara servlet dan JSP adalah: pada servlet layer aplikasi tidak sepenuhnya terpisah dari layer presentasi, dimana logika aplikasi atau logika bisnis berada dalam file program Java. Sedangkan presentasi diletakkan dalam output berupa konten yang dihasilkan juga oleh servlet.

JSP sendiri lebih menitikberatkan pada aspek presentasi ketimbang aspek aplikasi. Untuk JSP, kode Java dan HTML digabungkan dalam satu file yaitu file yang memiiki ekstensi *.jsp. Dalam JSP layer presentasi boleh dikatakan terpisah dari logika aplikasi atau logika bisnis.

Bahkan dalam perkembangannya sekarang JSP dapat tidak mengandung kode Java sama sekali. Beberapa logika pemrograman Java dapat digantikan oleh *tag-library*. Misalnya JSTL *(JavaServer Pages Standard Tag Library)* dapat mengenali beberapa logika pemrograman seperti perulangan dan kondisional [WIJ-07:20].



Gambar 2.2 Pemrosesan JSP
Sumber : [CHO-05:216]

Pada Gambar 2.2 ditunjukkan tahapan file JSP dieksekusi. Pada tahap pertama, JSP pertama kali diubah ke dalam kode Java. Tahap ini disebut sebagai tahap translasi. Proses ini berjalan dalam kurun waktu yang disebut sebagai waktu translasi. Pada tahap kedua, kode Java yang dihasilkan dari translasi JSP, dikompilasi menjadi *executable byte code.* Tahap kedua ini disebut sebagai tahap kompilasi. *Executable byte code* yang dihasilkan dalam bentuk binary yang dapat dieksekusi oleh JVM. Waktu untuk proses kompilasi ini disebut sebagai waktu kompilasi. *Executable byte code* akan dieksekusi setiap ada proses *request* dan *response.* Sekali file JSP sudah ditranslasi dan kemudian dikompilasi menjadi *Executable byte code,* maka hasil ini akan bersifat *re-usable*. Artinya, proses yang sama tidak perlu dilakukan lagi setiap kali ada *request.* Proses tersebut akan dilakukan lagi ketika ada modifikasi pada file JSP [CHO-05:215-216]. Beberapa langkah dalam mengembangkan JSP diantaranya adalah [PER-04:3]:

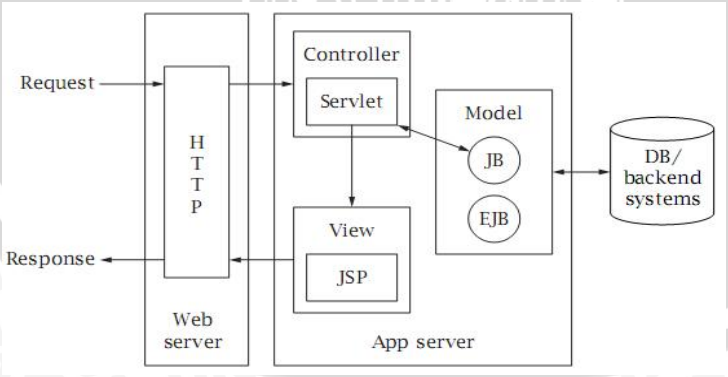
1. Buka editor teks, atau aplikasi IDE yang memberikan dukungan terhadap JSP.
2. Jika JSP dikembangkan untuk menangani HTTP *request,* maka dimasukkan kode HTML.
3. Gunakan JSP *directive* yang diperlukan. Hal ini ditandai dengan tag <% @

4. Masukkan aksi atau tag dimanapun diperlukan
5. Simpan dalam format file *.jsp

1.1.3. Struts Framework

Struts merupakan aplikasi framework open source yang di design untuk membantu para developer dalam membangun arsitektur aplikasi berbasis pada Model-View-Controller (MVC), dimana design MVC sudah merupakan standard umum dalam membangun aplikasi Web dengan penggunaaan servlet Java dan Java Server Pages (JSP) sebagai teknologinya.

Aplikasi apapun, bagian dalam kode sering mengalami perubahan terutama bagian user interfacenya. *Bussiness – logic* yang rumit pada user *interface* memungkinkan sekali untuk melakukan perubahan yang kompleks, yang lebih rumit sehingga mudah terjadi kesalahan, karena perubahaan pada satu bagian berpotensi mengakibatkan perubahaan pada keseluruhan aplikasi. Oleh sebab itu pola MVC menyediakan sebuah solusi untuk mengatasi permasalah tersebut, dengan membagi aplikasi menjadi bagian – bagian tersendiri yaitu *Model, View* dan *Controller*. MVC terbukti merupakan sebuah pola yang lebih efektif dalam membangun sebuah proyek. Pola pengembangan aplikasi dengan MVC hanya dapat diterapkan pada bahasa pemrograman yang berbasis objek (OOP), karena MVC sendiri merupakan objek dengan fungsi yang berbeda. Arsitektur MVC ditunjukkan dalam Gambar 2.3



Gambar 2.3 Arsitektur MVC *(Model-View-Controller)*
Sumber : [SPI-03: 6]

Model merupakan sesuatu yang merepresentasikan data, misalnya tabel database yang diolah (simpan, ubah, hapus) oleh *Controller* untuk ditampilkan

(*View*). *Model* ini mengatur respon terhadap permintaan, serta memberi hak akses untuk memanipulasi data. *Framework* berbasis MVC menekankan pada pentingnya desain database yang valid, menggunakan *Model* ini memungkinkan *developer* melakukan *query* antar *database* bila diperintahkan oleh *Controller* (*logic*). Beberapa kelebihan menggunakan *Model*, yaitu dalam proses *maintenance* aplikasi yang lebih menguntungkan karena detail dari data dan operasinya dapat ditempatkan pada area yang ditentukan oleh *Model*. Keuntungan lainnya komponen *Model* dapat digunakan kembali oleh aplikasi lain yang memiliki kegunaan / fungsi yang hampir sama, karena telah dipisahkan secara total antara data dengan *interface*-nya.

View merupakan informasi yang ditampilkan ke *user* sebagai media *interface* (menggunakan HTML, CSS, javasript), jadi tidak berisi proses bisnis yang berhubungan dengan data pada *database*. Komponen *grafis* menyediakan *representasi* proses internal aplikasi dan menuntun alur interaksi user terhadap aplikasi yang ada. Contoh *View* adalah template dari tampilan aplikasi atau pada website yang kita lihat, sehingga tidak ada layer lain yang berinteraksi dengan user kecuali pada layer *View* ini. Keuntungan *View* antara lain, memudahkan desainer sehingga bisa berkonsentrasi penuh pada desain tanpa harus memperhatikan hal – hal detail lainnya. Dan juga, memungkinkan ketersediaan multiple *interface* (Swing, Web, Console) dalam aplikasi namun mengeksekusi komponen *Model* sesuai fungsionalitas yang diharapkan.

Controller akan melakukan segala aktifitas proses bisnis dan aktifitas control lainnya seperti mengolah data dari *Model*, menyimpannya dalam variabel – variabel (manipulasi data) lalu menampilkannya pada *View*, benar atau tidaknya hasil olahan data akan sangat tergantung dari logika kerja aplikasi yang tersusun pada bagian *Controller* ini, sehingga *Controller* bisa disebut sebagai bagian yang paling signifikan dari aplikasi berbasis MVC. Dalam *Controller* ini menyediakan detail alur program dan bertanggung jawab menampung events dari user melalui *View* dan mengupdate komponen *Model* menggunakan data yang dimasukkan user. *Controller* juga bertugas melakukan packaging dan routing HTTP request ke obyek logik bisnis yang benar (bisa berupa halaman JSP atau class Action)

Arsitektur MVC secara sederhana dirancang dan diadaptasi dalam penggunaan Web – Application. Arsitektur yang dihasilkan disebut dengan Model 2 Architecture. Umumnya aplikasi Model 2 memiliki: *Servlet Controller* yang menyediakan akses tunggal terhadap keseluruhan aplikasi. *Controller* ini bertanggungjawab menyediakan manajemen terpusat terhadap alur aplikasi, penanganan security dan user management. *Controller Servlet* umumnya menggunakan konfigurasi XML untuk menentukan alur aplikasi dan pemrosesan perintah. Hal itu membuat helper components (berfungsi sebagai Command objects) terasosiasikan dengan user actions dan dibuat/dipanggil untuk menangani actions yang terjadi, memanggil komponen *Model* sebagaimana diperlukan untuk memisahkan antara *Controller* servlet dari *Model*.

Model merupakan obyek yang akan menyimpan request dari pengguna (user) selama proses berlangsung, biasanya implementasi berupa JavaBean. Obyek model biasanya bersifat *application spesific*, karena implementasi daripada logika bisnis akan sangat tergantung kepada kepentingan tujuan dari aplikasi yang dibangun. Struts menyediakan class ActionForm dan Action. Melalui kedua model ini sebuah validasi form berlangsung, ataupun pra proses terhadap data berlangsung (misalkan : memeriksa apakah nama anggota sudah diisi, alamat email, dll sbgnya.)

Struts-config.xml

Struts-config.xml adalah tempat yang digunakan untuk menempelkan semua implementasi class Action dan model menjadi satu. Semua aturan main, dan aliran aplikasi yang dibangun berbasis Struts ada disini. Jika web.xml memberitahukan container dimana sebuah request harus ditranfer setelah sampai ke container, maka struts-config.xml memberitahukan kepada container apa yang harus dilakukan setelah mencapai layar ini. Controller mengatur request yang datang dan menentukan harus di kirim ke mana. Mapping Action yang diset di dalam struts-config.xml. akan mengarahkan request ke Action yang sesuai.

1.1.4. MySQL

MySQL adalah software RDBMS *(Relational DataBase Management System)* yang bersifat *open source*. Pada mulanya, MySQL kadang-kadang

menemui beberapa pelawanan karena dukungannya yang kurang mengakomodasi untuk beberapa intiSQL, seperti *subselect* dan *foreign key*. Tetapi di sisi lain MySQL mendapatkan banyak ketertarikan karena bersifat *free*, performa kerja yang bagus dan kemudahan untuk dipakai. Selain itu MySQL juga mendukung banyak teknologi pemrograman seperti PHP, Java, Perl, Phyton [CON-04:4].

MySQL sebenarnya merupakan produk yang berjalan pada platform Linux. Karena sifatnya yang *open-source*, MySQL dapat dijalankan pada semua platform baik Windows maupun Linux. Selain itu MySQL juga merupakan program pengakses database yang bersifat jaringan sehingga dapat digunakan untuk aplikasi *multiuser* [NUG-04:29].

1.2. RSS

RSS adalah sebuah file berformat XML untuk sindikasi yang telah digunakan (diantaranya dan kebanyakan) situs web berita dan weblog. Teknologi yang dibangun dengan RSS mengijinkan kita untuk berlangganan kepada situs web yang menyediakan umpan (feed) biasanya situs web yang isinya selalu diganti secara regular. Kita membutuhkan layanan pengumpul dalam pemanfaatan teknologi ini. Pengumpul bisa dibayangkan sebagai kotak surat pribadi. Kemudian kita dapat mendaftar ke situs yang kita ingin tahu perubahannya.[MON-05] .Tapi, kita hanya bisa mendapatkan satu baris atau sebuah pengantar dari isi situs berikut alamat terkait untuk membaca isi lengkap artikelnya.

RSS digunakan secara luas oleh komunitas *weblog* untuk menyebar ringkasan tulisan terbaru di jurnal, kadang-kadang juga menyertakan artikel lengkap dan bahkan gambar dan suara. Pada tahun 2000, penggunaan RSS meluas di berbagai penerbitan berita, termasuk Reuters, CNN, dan BBC. RSS digunakan pada hampir semua situs berita atau *weblog*, dengan berbagai tujuan termasuk: pemasaran, press release, laporan regular produk, atau aktivitas lain yang membutuhkan pemberitahuan periodic dan tentunya publikasi.

Sebuah program komputer yang dikenal sebagai pembaca umpan (feed reader) bertindak sebagai pengumpul. Program ini mengecek situs yang menyediakan RSS dan menampilkan berbagai artikel baru yang ditemukan. Tenggang waktu dan siklus pengumpulan RSS biasanya dapat diatur oleh

penggunanya. Program pengumpul dapat berupa program computer atau sebuah layanan *web* yang tersedia *online.*

Program pengumpul RSS di computer biasanya berupa aplikasi (*software)* sendiri yang harus dipasang di computer sebelum dapat digunakan. Program ini tersedia untuk berbagai jenis sistem operasi dengan harga bervariasi. Ada juga program pengumpul RSS yang gratis.

Program pengumpul di *web* tidak memerlukan pemasangan dan pengaturan, kita dapat melihat dan mengecek RSS kita dimana saja, asal ada browser dan koneksi *internet*. Beberapa layanan pengumpul RSS juga menyediakan penggabungan dan juga pencarian.

Syndication feeds telah menjadi sebuah tool standar di dunia web. Suatu tanda ketika sebuah web menggunakan *syndication feeds* adalah adanya sebuah tombol kecil berwarna orange dan bertuliskan XML berwarna putih, atau mungkin juga tombol dengan label Atom, RSS 2.0, RSS 1.0 atau juga Feed, seperti ditunjukkan pada Gambar 2.4. Ini semua merupakan contoh *syndication feeds* yang merupakan file dengan konten dalam format XML yang berisi item-item yang tersusun dari item yang paling terakhir diposting [POW-05:1].

Gambar 2.4 Tombol RSS dan XML
Sumber : [FIN-05:16]

1.2.1. Versi dan Modul RSS

Terdapat beberapa versi yang berbeda dari format RSS yang digunakan sekarang ini. Versi yang sering digunakan adalah RSS 1.0 dan RSS 2.0. Masing-masing versi memiliki keunggulan dan efek balik sendiri-sendiri. RSS 2.0 dikenal dengan tingkat kesederhanaan dan kemudahan dalam penggunaannya, sedangkan RSS 1.0 dikenal karena kemudahan dalam pengembangannya dan memiliki spesifikasi yang lengkap [MON-05]. Kedua format ini berbasis XML dan memiliki struktur dasar yang sama.

A. RSS 2.0

RSS 2.0 dikenalkan pertama kali oleh UserLand`s Dave winer. Dalam versi ini RSS merupakan akronim dari *Really Simple Syndication* dan memfokuskan pada tingkat kesederhanaan.

```
<?xml version=”1.0”?>
<rss version=”2.0”>
    <channel>
     <title></title>
     <link></link>
     <description></description>
          <item>
          <title></title>
          <link></link>
          <description></description>
          </item>
    </channel>
</rss>
```

Gambar 2.5 Struktur RSS 2.0
Sumber : [RON-08]

B. RSS 1.0

RSS 1.0 merupakan akronim dari RDF *Site Summary* dimana hal ini tergabung dalam RDF itu sendiri yang merupakan standarisasi web untuk metadata.

```
<?xml version=”1.0”?>
<rdf : RDF>
    <channel>
     <title></title>
     <link></link>
     <description></description>
     <items>
       <rsd: Seq>
         <rdf:lresource=”http://example.com/2002/09/01”>
         <rdf:lresource=”http://example.com/2002/09/02”>
       </rdf:Seq>
     </items>
   </channel>
</rdf:RDF>
```

Gambar 2.6 Struktur RSS 1.0
Sumber : [RON-08]

C. Atom

Kelemahan dari RSS 2.0 dan RSS 1.0 adalah dalam kedua versi RSS ini tidak memiliki standarisasi yang resmi. Atom memiliki standarisasi yang resmi, mengadaptasi fungsi RSS dari kedua versi ini dan tetap mempertahankan format berbasis XML.

Atom memiliki sebuah *tag feed* yang berisikan metadata dari kedua versi RSS sebelumnya. Penggunaan *tag-tag* lainnya seperti *tittle, link, id* dapat digunakan pada RSS versi ini. Standarisasi ini dimaksudkan untuk memudahkan para *web developer* dalam mengembangkan teknologi ini.

```
<?xml version=”1.0” encoding=”utf-8”?>
<feed xmlns=”http://www.w3.org/2005/Atom”>
       <title></title>
       <link></link>
       <update></update>
       <author></author>
     <entry>
          <title></title>
                <link></link>
          <update></update>
               <author></author>
     </entry>
</feed>
```

Gambar 2.7 Struktur Atom
Sumber : [RON-08]

1.2.2. RSS *Syntax*

Untuk mengaplikasikan sebuah RSS dalam sebuah situs, terlebih dahulu dibuat sebuah dokumen RSS dan menyimpanya pada sebuah file berekstensi .xml. Kemudian mendaftarkan pada sebuah RSS *agregrator*. Setiap hari *aggregrator* mencari dokumen RSS dari website yang telah terdaftar dan menampilkan informasi tentang *feed.*

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<rss version="2.0">
<channel>
  <title>W3Schools Home Page</title>
  <link>http://www.w3schools.com</link>
  <description>Free web building
tutorials</description>
  <item>
    <title>RSS Tutorial</title>
    <link>http://www.w3schools.com/rss</link>
    <description>New RSS tutorial on
W3Schools</description>
  </item>
  <item>
    <title>XML Tutorial</title>
    <link>http://www.w3schools.com/xml</link>
    <description>New XML tutorial on
W3Schools</description>
  </item>
</channel>
</rss>
```

Gambar 2.8 Penulisan sintak sebuah dokumen RSS
Sumber : [RON-08]

Baris pertama menunjukkan deklarasi XML, baris berikutnya merupakan deklarasi RSS yang menunjukkan bahwa RSS yang digunakan adalah versi RSS 2.0. Pada baris selanjutnya terdapat elemen <*channel*>. Elemen ini digunakan untuk menggambarkan RSS *feed*. Elemen <*channel*> memiliki anak elemen :

1. <*tittle*> yang mendefinisikan judul dari *channel*.
2. <*link*> yang mendefinisikan *hyperlink* ke *channel*.
3. <*description*> yang mendefinisikan *channel.*

Tiap elemen <*channel*> dapat memiliki satu atau lebih <*item*> elemen. Masing-masing memiliki elemen <*item*> mendefinisikan sebuah artikel atau "cerita" pada RSS *feed*. Elemen <*item*> memiliki tiga anak elemen :

1. <*tittle*> yang mendefinisikan judul dari *item*
2. <*link*> yang mendefinisikan *hyperlink* ke *item*
3. <*description*> yang mendefinisikan *item.*

1.2.3. RSS Reader

RSS *reader* digunakan untuk membaca RSS *feed*. Sudah banyak RSS *reader* yang tersedia. Beberapa bekerja sebagai *web service*, dan beberapa terbatas pada sistem operasi (contoh: Windows, Mac, Linux, dll). Secara garis besar RSS dibaca dengan cara *parsing* file RSS kemudian mengembalikan nilai entri dengan *field* yang terdapat di dalam file RSS.[WOR-09]

```
class FeedReader {
SyndFeed feed;
String url,description,title;
int numEntries;
FeedEntry entry[];

public FeedReader(String _url) {
url=_url;
try {

feed=new SyndFeedInput().build(new XmlReader(new U
RL(url)));
description=feed.getDescription();
title=feed.getTitle();

java.util.List entrl=feed.getEntries();
Object [] o=entrl.toArray();
```

```
  numEntries=o.length;

entry=new FeedEntry[numEntries];
        for(int i=0; i< numEntries; i++) {

        entry[i]=new FeedEntry((SyndEntryImpl)o[i]);

          println(i+": "+entry[i]);
         }
       }
       catch(Exception e) {

   println("Exception in Feedreader: "+e.toString());

         e.printStackTrace();
       }
     }

  }
```

Gambar 2.9 Contoh Kode Javascript Untuk Membaca RSS *feed*
Sumber : [WOR-09]

Beberapa langkah ketika diinginkan untuk menyediakan *syndication feeds* pada suatu website diantaranya [FIN-05:15]:

1. Mengidentifikasi konten dari website yang di-*update* secara teratur, yang diinginkan untuk bisa dilihat oleh user.
2. Membuat dokumen RSS yang menjelaskan dan mengarahkan ke konten yang selengkapnya. Mayoritas dokumen RSS terdiri dari beberapa item, dan masing-masing item mengarahkan ke lokasi konten selengkapnya yang berbeda. Dokumen RSS merupakan dokumen dengan format XML.
3. *Reviewer* yang akan berlangganan *update* berita RSS akan menambahkan URL dari dokumen RSS ke RSS *reader* mereka, seperti ditunjukkan pada Gambar 2.10.

Gambar 2.10 Alur pembacaan dokumen RSS
Sumber : [FIN-05:16]

1.3. XML

XML adalah sebuah *markup language*. XML menggunakan tag-tag untuk memberi label, mengkategorikan, dan mengorganisir informasi dengan cara yang spesifik. *Markup* menggambarkan dokumen atau struktur data dan organisasi. konten seperti teks, gambar, dan data, merupakan bagian yang dikandung di dalam tag-tag *markup*. XML tidak terbatas pada satu set *markup*. Kita dapat membuat *markup* buatan sendiri yang cocok dengan data dan dokumen kita. Fleksibilitas XML menyebabkan penggunaannya semakin luas untuk pertukaran data dengan bentuk yang berbeda-beda [DYK-05:11].

Salah satu fungsi utama dari XML adalah untuk mengklasifikasikan informasi. Sebagai permisalan, informasi mengenai buku bisa diklasifikasikan menjadi [DYK-05:14]:

- Judul
- Pengarang
- Penerbit
- Harga
- Isi
- Format
- ISBN

Dengan menggunakan XML, informasi tersebut bisa diklasifikasikan dengan menggunakan tag XML. Contohnya seperti

```
<book>
<title>Night Fall</title>
<author>Demille, Nelson</author>
<publisher>Warner</publisher>
<price>$26.95</price>
<contentType>Fiction</contentType>
<format>Hardback</format>
<isbn>0446576638</isbn>
</book>
```

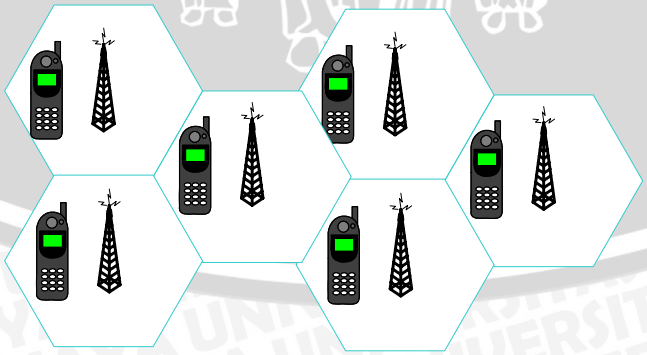
Data tersebut tersimpan dalam file *.xml dan ketika kita buka dengan menggunakan *web browser* akan nampak seperti pada Gambar 2.11

Gambar 2.11 Sebuah file XML dibuka dengan menggunakan Internet Explorer

Sumber : [DYK- 05:19]

1.4. Komunikasi Seluler

Istilah seluler menunjukkan bahwa jaringan komunikasi dibagi menjadi sejumlah sel, atau cakupan area geografis tertentu seperti ditunjukkan pada Gambar 2.12 Di dalam setiap sel terdapat sebuah *base station* yang terdiri dari transmisi sinyal radio *(radio transmission)* dan seperangkat peralatan penerima. *Base station* ini yang akan menyediakan layanan komunikasi menggunakan sinyal radio untuk perangkat ponsel yang ada di dalam area sel tersebut. Area cakupan dari masing-masing sel ditentukan oleh beberapa faktor seperti daya pancar dari *base station*, daya pancar perangkat *mobile*, tinggi antena, serta topologi permukaan tanah [SMI-04:11].



Gambar 2.12. Sistem Seluler

Sumber : [SMI-04:12]

1.4.1. SMS

Short Message Service (SMS) merupakan layanan dasar yang memungkinkan pertukaran pesan singkat dalam bentuk teks antar para pelanggan. Pesan teks singkat pertama kali dikirimkan pada tahun 1992 melalui jaringan GESM Eropa. Karena percobaan yang sukses ini, SMS kemudian menjadi hal yang sangat cepat perkembangannya. *Mobile Data Association* melaporkan bahwa total pengiriman pesan teks melalui jaringan GSM di UK pada tahun 2003 sejumlah 20.5 milyar. Standard SMS diinisiasi oleh ETSI. Karena pada awalnya layanan SMS ini dikenalkan pada jaringan GSM, SMS telah disesuaikan sehingga bisa diimplementasikan dengan teknologi lain seperti CDMA dan GPRS [BOD-05:47].

Beberapa layanan yang berbasis pada teknologi SMS yang bisa digunakan oleh pelanggan seperti [BOD-05:48-50]:

- *Person-to-Person Messaging*

  Layanan ini merupakan fungsi asli SMS didesain. Layanan ini memungkinkan pertukaran pesan singkat dalam bentuk teks yang terjadi antara dua orang pelanggan langsung.

- *Information Services*

  Aplikasi ini menggunakan skenario *machine-to-person.* Dengan aplikasi ini informasi semisal perubahan cuaca atau laporan keuangan bisa ditambahkan dalam *value added service* (VAS) yang bisa dikirimkan ke pelanggan melalui SMS.

- *Internet Email Alert*

  Dengan aplikasi ini, pelanggan akan mendapatkan SMS berupa pemberitahuan bahwa ada email yang masuk dan bisa untuk dilihat. Pesan biasanya berisi alamat pengirim email, judul email dan beberapa kata awal dari isi email.

- *Download Service*

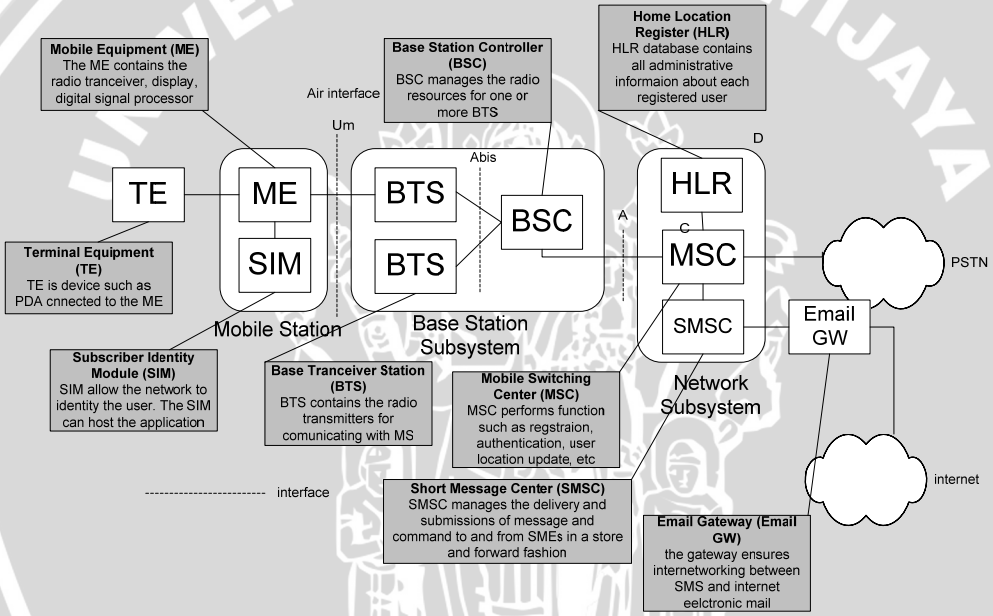
  Aplikasi ini biasanya digunakan oleh pelanggan untuk merubah tampilan ponsel. Seperti mengubah gambar animasi pada saat ponsel dinyalakan atau dimatikan, *ringtone*, *theme* dan sebagainya. Obyek yang digunakan

untuk melakukan kustomisasi ini bisa didapat dengan men-*download* sebagai satu atau lebih *short message*.

- *Chat Application*

  Beberapa aplikasi *chatting* menggunakan teknologi SMS ada proses pengiriman pesan singkat.

Gambar 2.13 menunjukkan arsitektur SMS pada jaringan GSM. Dua elemen tambahan pada jaringan tersebut adalah *SMS Center* dan *Email Gateway*. Selain itu juga terdapat istilah *Short Message Entity* (SME) yang biasanya merupakan software aplikasi yang ada dalam ponsel yang digunakan untuk menangani pesan seperti mengirimkan, menerima, menyimpan dan sebagainya.



Gambar 2.13. Arsitektur SMS pada jaringan GSM
Sumber : [BOD-05:52]

1. *Short Message Entity* (SME)

Elemen yang bisa mengirimkan atau menerima pesan singkat disebut sebagai *Short Message Entities* (SME). SME bisa berupa software aplikasi yang ada dalam ponsel, atau juga bisa berupa *facsimile*, *telex*, atau *remote Internet server*, dan sebagainya. Sebuah ponsel harus dikonfigurasi sedemikian rupa sehingga bisa bekerja dengan benar di dalam jaringan *mobile*. Biasanya ponsel telah dikonfigurasi selama proses produksi. Jika diinginkan diset secara manual seperti terlihat pada ilustrasi Gambar 2.14. SME bisa berupa sebuah server yang

terkoneksi langsung ke SMSC atau melalui gateway. Sehingga SME bisa juga disebut sebagai ESME *(External Short Message Entities)*. Biasanya ESME bisa berupa *WAP proxy/server*, e*mail gateway*, atau *voice mail server*. SME yang mengirimkan pesan disebut sebagai SME pengirim, sedangkan SME yang menerima disebut sebagai SME penerima [BOD-05:53].

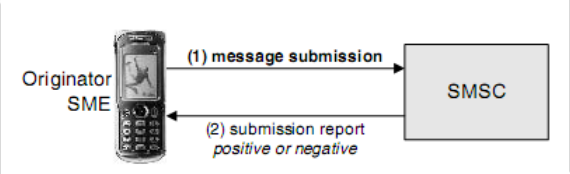
Gambar 2.14 Setting SMS
Sumber : [BOD-05:53]

2. *Service Center* (SC) atau *SMS Center* (SMSC)

Service Center (SC) atau *SMS Center* (SMSC) memainkan peran kunci di dalam arsitektur SMS. Fungsi utama dari SMSC adalah menyiarkan pesan singkat antar SME dan melakukan *store-and-forwarding* pesan singkat (pesan singkat akan disimpan jika SME penerima sedang tidak aktif). SMSC bisa jadi menjadi satu dengan *mobile network* (semisal menjadi satu dengan MSC) atau menjadi bagian dari jaringan yang terpisah/berbeda. SMSC bisa berada di luar jaringan dan dikelola secara mandiri oleh pihak ketiga. Secara praktis, banyak operator seluler yang memiliki satu atau lebih SMSC, karena layanan SMS sekarang ini merupakan layanan yang sangat banyak digunakan. Dari sisi teori, sebuah SMSC bisa digunakan oleh lebih dari satu operator seluler, tetapi mayoritas sekarang SMSC hanya digunakan khusus untuk mengelola layanan SMS pada sebuah operator saja. Biasanya antar operator seluler memiliki kesepakatan komersial untuk melakukan pertukaran pesan antar jaringan yang berbeda yang dipegang oleh masing-masing operator. Artinya sebuah SME yang ada dalam jaringan A bisa bertukar pesan dengan SME yang berada di luar jaringan A, misalnya

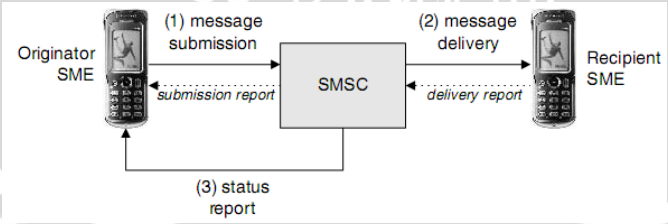
jaringan B. Sekarang ini, sebuah SMSC komesial bisa digunakan untuk menangani 1000 pesan tiap detik [BOD-05:53-54].

Dalam konteks sistem SMS, terdapat istilah *message submission* yang artinya merupakan proses pengiriman pesan dari *Short Message Entity (SME)* menuju ke SMSC. Interaksi ini ditunjukkan pada Gambar 2.15. Ketika pesan dari SME pengirim berhasil diterima oleh SMSC, maka SMSC akan mengembalikan laporan berupa *positive submission report,* dan jika gagal maka akan mengembalikan *negative submission report.* Jika SME pengirim tidak menerima laporan dari SMSC dalam jangka waktu yang telah ditentukan dalam konfigurasi, maka SME pengirim tersebut akan menyimpulkan bahwa proses *submission* gagal.



Gambar 2.15 Proses *message submission*
Sumber : [BOD-05:69]

Sekali pesan dari SME telah berhasil diterima oleh SMSC, maka SMSC akan melakukan *query* terhadap HLR dengan tujuan untuk mem-*forward* pesan tersebut menuju ke nomor ponsel penerima atau SME penerima [BOD-05:69]. Setelah SME penerima menerima pesan, maka akan dikirimkan *delivery report* ke SMSC. Dan *delivery report* ini akan dikirimkan kembali ke SME pengirim, seperti ditunjukkan pada Gambar 2.16 [BOD-05:59]



Gambar 2.16 Proses *message transfer*
Sumber : [BOD-05:59]

3. Email Gateway

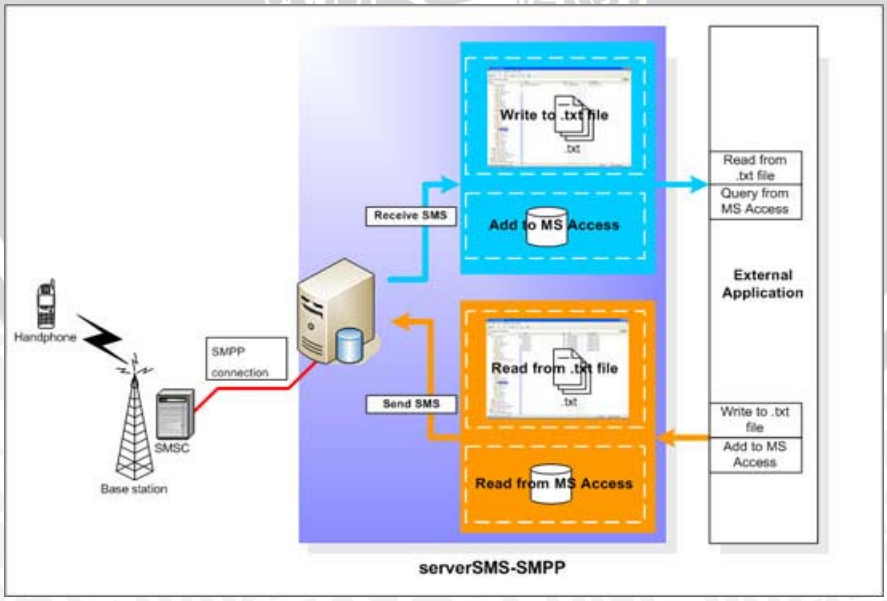
Email gateway memungkinkan interoperabilitas antara email dengan sms dengan menyediakan interkoneksi SMSC dengan internet. Dengan menggunakan *email*

gateway, pesan dapat dikirimkan oleh sebuah SME ke internet dan sebaliknya. Aturan *email gateway* adalah untuk mengubah format pesan (dari SMS ke email dan sebaliknya) dan untuk mengirimkan pesan antara SMS dan domain di internet [BOD-05:54]

1.5. Protokol SMPP

Short Message Peer-to-peer Protocol, adalah Merupakan protokol standar dalam industri telekomunikasi yang dikhususkan untuk pertukaran pesan singkat atau SMS antar SMSC. Protokol ini didesain berpasangan dan didasarkan pada konsep pertukaran data dengan cara pengiriman data *request/response* lewat PDU atau protokol data unit melalui lapisan keempat OSI (*Open System Interconnection*) menggunakan koneksi TCP atau X.25 SVC3. Untuk efisiensi, data tersebut berupa data biner yang dikodekan. Protokol SMPP yang digunahkan operator umumnya adalah versi 3.4. Pada versi ini, pada satu koneksi dapat dilakukan pengiriman maupun penerimaan pesan. Pertukaran data dilakukan secara sinkron maupun asinkron. [SSS-04].

Dengan menggunakan koneksi SMPP (*Short Message Peer to Peer*) kecepatan ke operator pengiriman mencapai 30 SMS per detik. Software berfungsi untuk menerima SMS kemudian disimpan dalam bentuk text file dan membaca text file untuk mengirim SMS.



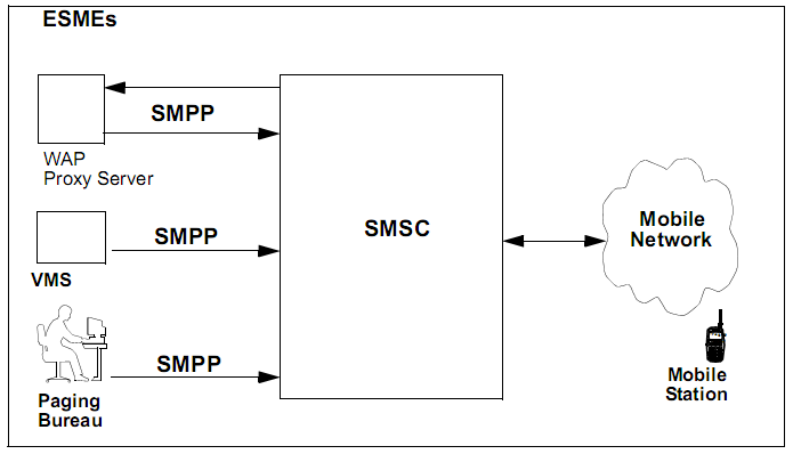
Gambar 2.17. serverSMS-SMPP

Sumber : [SSS-04]

SMPP mendukung penuh fungsi *messaging* dua arah seperti:

- Mengirimkan pesan dari sebuah ESME ke satu atau banyak tujuan melalui SMSC
- Sebuah ESME dapat menerima pesan melalui SMSC dari SME (*Short Message Entity*) seperti *Mobile Stations* yang lain
- Melakukan *query* status dari sebuah pesan yang tersimpan di SMSC
- Membatalkan atau mengubah sebuah pesan yang tersimpan di SMSC
- Manjadwalkan pengiriman pesan
- Mengatur prioritas pengiriman pesan

Gambar 2.18 menunjukkan arsitektur sistem dimana SMPP diimplementasikan



Gambar 2.18. Konteks SMPP di dalam *mobile network*
Sumber : [SMP-07]

SMPP didasarkan pada pertukaran *request* dan *response* protokol data unit (PDUs) antara ESME dan SMSC melalui koneksi jaringan TCP/IP or X.25. Protokol SMPP mendefinisikan:

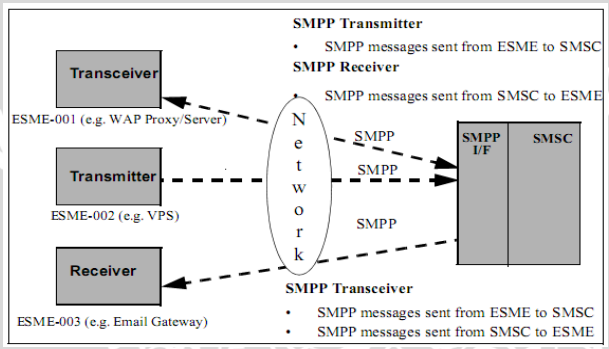
- Sebuah set operasi dan Protocol Data Units (PDUs) untuk pertukaran pesan singkat antara ESME dan SMSC
- Data yang bisa saling ditukarkan antara ESME dan SMSC selama operasi SMPP

Setiap operasi pada SMPP harus terdiri dari sebuah *PDU request* dan PDU *response*. Entitas yang diterima harus mengembalikan respon SMPP untuk

sebuah *request* PDU [SMP-07]. Pertukaran pesan antara ESME dan SMSC melalui protokol SMPP dikategorikan menjadi tiga macam:

- Pesan dikirimkan dari ESME (sebagai Transmitter) ke SMSC.
- Pesan dikirimkan dari SMSC ke ESME (sebagai *Receiver*).
- Pesan dikirimkan dari ESME ke SMSC dan pesan dikirimkan dari SMSC ke ESME. Dalam kategori ini ESME bertindak sebagai *Transceiver*.

Gambar 2.19 menunjukkan ilustrasi interface SMPP antara ESME dan SMSC



Gambar 2.19 SMPP *interface* antara ESME dan SMSC
Sumber : [SMP-07]

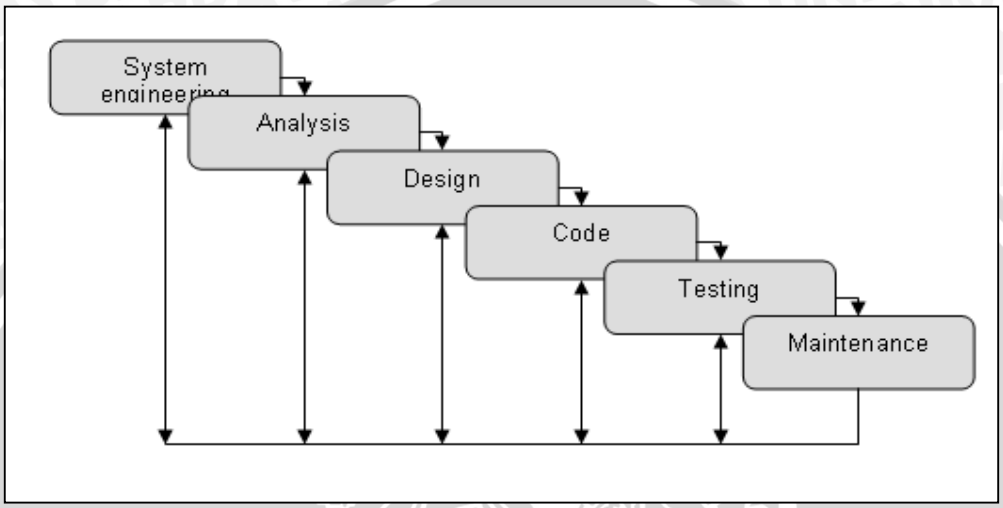
1.6. Metode Pengembangan Perangkat Lunak

Rekayasa perangkat lunak pada prinsipnya menekankan pada tahapan-tahapan pengembangan suatu perangkat lunak yakni : Analisis, Desain, Implementasi, Testing dan Maintenance. Pada tahap yang lebih luas Rekayasa Perangkat Lunak mengacu pada Manajemen Proyek pengembangan Perangkat Lunak itu sendiri dengan tetap memperhatikan tahapan-tahapan pengembangan sebelumnya.

Dalam pengembangannya perangkat lunak memiliki berbagai model yaitu model water fall, model prototype, model sequensial linear, model RAD (Rapid Aplication Model), dan model formal. Disini sebelum diadakannya implementasi terlebih dahulu rancangan model yang dibuat diverifikasi terlebih dahulu sehingga tidak ada lagi kesalahan - kesalahan pada saat implementasi.

Dalam melakukan proses pengembangan suatu perangkat lunak, diperlukan suatu metode yang digunakan sebagai panduan untuk menghasilkan sebuah perangkat lunak yang benar dan berkualitas. Sehingga perangkat lunak yang dihasilkan bisa menguntungkan pihak *user* maupun *developer*. *User* bisa

mendapatkan perangkat lunak yang sesuai dengan permintaan dan kebutuhan yang diinginkan, dan di pihak *developer* dapat menyelesaikan sebuah perangkat lunak dengan terjadwal, serta mendapatkan kemudahan dalam melakukan *maintenance* (pemeliharaan). Dalam hal inilah urgensi dari sebuah rekayasa terhadap perangkat lunak *(Software Engineering).*



Gambar 2.20 Siklus pengembangan perangkat lunak model *Waterfall*
Sumber: [PRE-92:25]

Pada Gambar 2.20 terlihat siklus pengembangan suatu sistem perangkat lunak. Proses pengembangan perangkat lunak akan selalu melakukan proses pengumpulan terhadap kebutuhan *(requirement)* dari *user* (pengguna) atau *customer* (pelanggan), memahami dan menetapkan kebutuhan-kebutuhan tersebut. Langkah ini menjadi suatu pekerjaan yang sangat penting. Tahap yang selanjutnya adalah melakukan analisis terhadap kebutuhan dilanjutkan dengan perancangan sistem perangkat lunak yang akan dibangun. Hasil dari perancangan dijadikan acuan dalam proses *coding* untuk merealisasikannya ke dalam bentuk yang bisa diterjemahkan oleh mesin. Proses selanjutnya adalah pengujian terhadap perangkat lunak yang telah dihasilkan.

Terdapat dua *class*ifikasi metode pengembangan perangkat lunak, yaitu metode struktural dan metode berorientasi pada objek. Pada skripsi ini digunakan metode pengembangan perangkat lunak berorientasi objek.

1.6.1. Analisis Kebutuhan (Requirements Analysis)

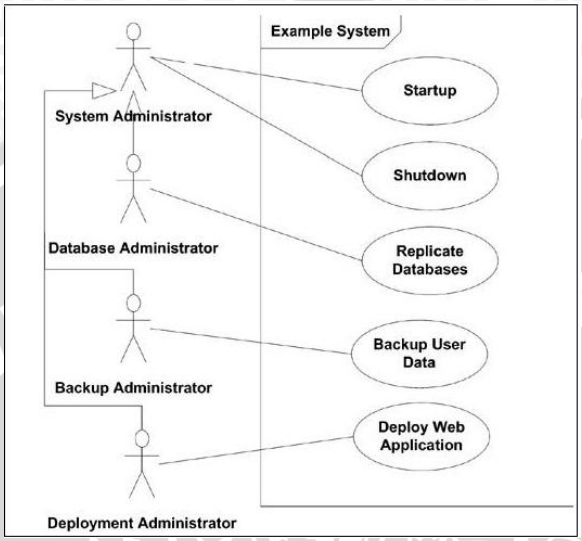
Metode analisis yang digunakan dalam mengembangkan perangkat lunak pada skripsi ini adalah metode analisis berorientasi objek *(Object Oriented Analysis).* Sasaran dari *Object Oriented Analysis (OOA)* adalah mengembangkan sederetan model yang menggambarkan perangkat lunak komputer pada saat perangkat itu bekerja untuk memenuhi serangkaian persyaratan yang ditentukan oleh pelanggan [PRE-02:686]. Proses OOA tidak dimulai dengan suatu pemikiran mengenai objek, melainkan dengan memahami cara sistem akan digunakan oleh manusia, bila sistem adalah *human interface*, oleh mesin bila sistem dilibatkan di dalam control proses, atau oleh program lain bila sistem berkoordinasi dan mengontrol aplikasi. Sekali skenario kegunaan didefinisikan, maka pemodelan perangkat lunak dimulai.

Proses analisis ini mengambil acuan dari hasil pengumpulan, pemahaman dan penetapan kebutuhan-kebutuhan *(requirements)* yang ingin didapatkan oleh pengguna, dan harus mempu disediakan oleh sistem perangkat lunak. Pada skripsi ini, digunakan *use case diagram* untuk memodelkan hasil analisis terhadap kebutuhan tersebut.

Berdasarkan kebutuhan-kebutuhan tersebut, perekayasa perangkat lunak dapat menciptakan serangkaian skenario yang masing-masing mengidentifikasi urutan pemakaian bagi sistem yang akan dibangun. Skenario tersebut yang sering disebut *use case* [PRE-02:697]. *Use case* adalah konstruksi untuk mendeskripsikan bagaimana *system* terlihat di mata pengguna potensial. *Use case* terdiri dari sekumpulan *scenario* yang dilakukan oleh seorang *actor* (orang, perangkat keras, urutan waktu atau *system* yang lain). Sedangkan *use case diagram* memfasilitasi komunikasi di antara analis dan pengguna serta di antara analis dan klien.

Sebuah *use case* dapat meng-*include* fungsionalitas *use case* lain sebagai bagian dari proses dalam dirinya. Secara umum diasumsikan bahwa *use case* yang di-*include* akan dipanggil setiap kali *use case* yang meng-*include* dieksekusi secara normal. Sebuah *use case* dapat di-*include* oleh lebih dari satu *use case* lain, sehingga duplikasi fungsionalitas dapat dihindari dengan cara menarik keluar fungsionalitas yang *common* [DHA-03:4].

Sebuah *use case* juga dapat meng-*extend use case* lain dengan behaviour-nya sendiri. Sementara hubungan generalisasi antar *use case* menunjukkan bahwa use case yang satu merupakan spesialisasi dari yang lain [DHA-03:4].



Gambar 2.21 Contoh *use case diagram*
Sumber : [PIL-05]

1.6.2. Perancangan *(Design)*

Metode perancangan yang digunakan untuk merancang sistem *software* pada skripsi ini adalah perancangan berbasis objek *(Object Oriented Design).* Perancangan berorientasi objek (OOD) mentransformasi model analisis yang dibuat dengan menggunakan *object oriented analysis (OOA)* ke dalam suatu model desain yang berfungsi sebagai cetak biru bangunan perangkat lunak. Tidak seperti metode desain perangkat lunak konvensional, OOD menghasilkan desain yang mencapai sejumlah atau tingkat yang berbeda dari modularitas. Komponen sistem mayor dikumpulkan dalam "modul" tingkat sistem yang disebut subsistem. Data dan operasi yang memanipulasi data tersebut dienkapsulasi ke dalam objek yang merupakan bentuk modular yang merupakan blok bangunan dari sebuah sistem berorientasi objek [PRE-02:723].

1.6.2.1. Class Diagram

Diagram *class* digunakan untuk menampilkan *class-class* atau paket-paket di dalam sistem dan relasi antar mereka. Ia memberikan gambaran sistem secara

statis. Biasanya, dibuat beberapa diagram *class* untuk satu sistem. Satu diagram *class* menampilkan subset dari *class-class* dan relasinya. Yang lainnya, mungkin menampilkan *class-class* termasuk atribut dan operasi dalam *class-class*. Dan yang lain lagi, mungkin menampilkan paket-paket *class* dan relasi antar paket-paket. Dapat dibuat beberapa diagram sesuai yang diinginkan untuk mendapatkan gambaran lengkap terhadap sistem yang dibangun. Tidak ada aturan yang mengharuskan berapa banyaknya diagram *class* harus dibuat.

Class memiliki tiga area pokok [DHA-03]:

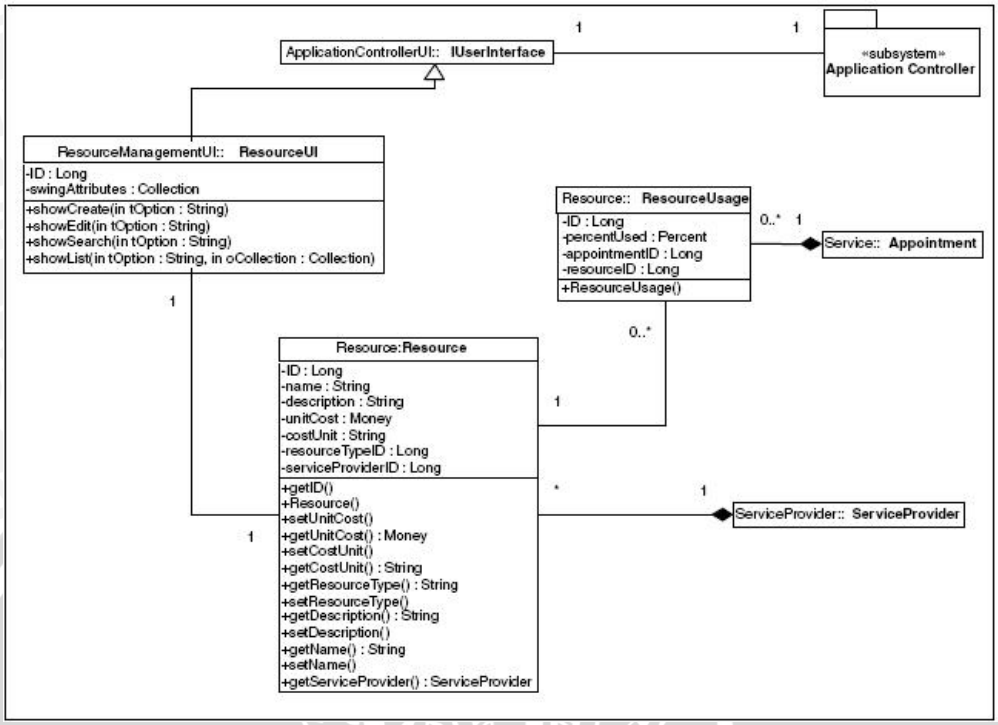
1. Nama (dan *stereotype*)
2. Atribut
3. Metoda

Atribut dan metoda dapat memiliki salah satu sifat berikut [DHA-03]:

1. **Private**, tidak dapat dipanggil dari luar class yang bersangkutan.
2. **Protected**, hanya dapat dipanggil oleh class yang bersangkutan dan anak-anak yang mewarisinya.
3. **Public**, dapat dipanggil oleh siapa saja.

1.6.2.1.1. Hubungan Antar *Class*

1. **Asosiasi**, yaitu hubungan statis antar *class*. Umumnya menggambarkan *class* yang memiliki atribut berupa *class* lain, atau *class* yang harus mengetahui eksistensi *class* lain. Panah *navigability* menunjukkan arah *query* antar *class*.
2. **Agregasi**, yaitu hubungan yang menyatakan bagian (“terdiri atas..”).
3. **Pewarisan**, yaitu hubungan hirarkis antar *class*. *Class* dapat diturunkan dari *class* lain dan mewarisi semua atribut dan metoda *class* asalnya dan menambahkan fungsionalitas baru, sehingga ia disebut anak dari *class* yang diwarisinya. Kebalikan dari pewarisan adalah generalisasi.
4. **Hubungan dinamis**, yaitu rangkaian pesan (*message*) yang di-*passing* dari satu *class* kepada *class* lain. Hubungan dinamis dapat digambarkan dengan menggunakan *sequence diagram* yang akan dijelaskan kemudian.



Gambar 2.22 Contoh *class diagram*
Sumber: [KEY-05]

2.6.2.2 *Sequence Diagram*

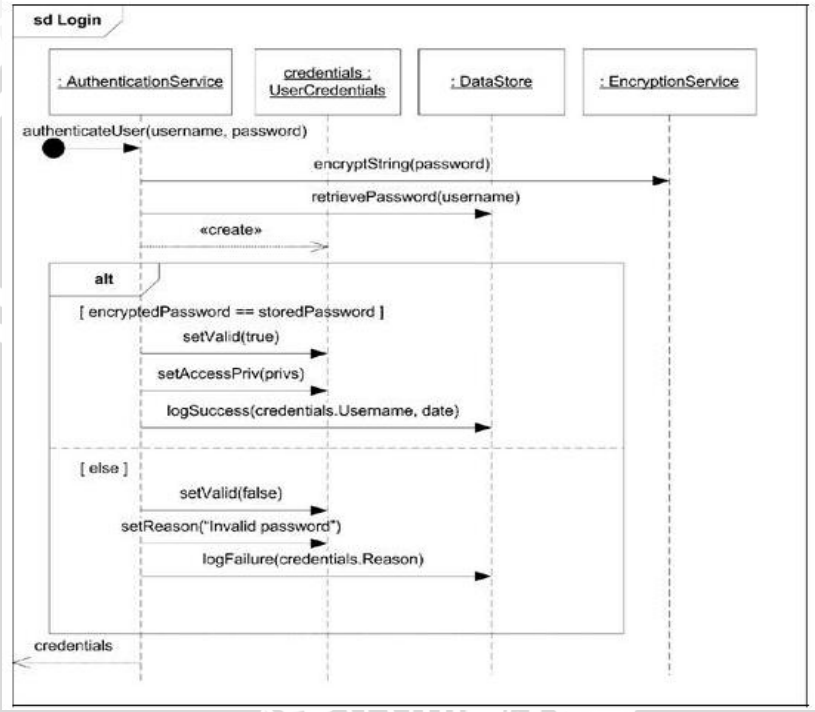
Sequence diagram digunakan untuk menggambarkan perilaku pada sebuah skenario. Diagram ini menunjukkan sejumlah contoh obyek dan *message* (pesan) yang diletakkan di antara obyek-obyek ini di dalam *use case*.

Komponen utama *squence diagram* terdiri atas obyek yang dituliskan dengan kotak segiempat bernama. *Message* diwakili oleh garis dengan tanda panah dan waktu yang ditunjukkan dengan progress vertical.

Sequence diagram menggambarkan interaksi antar objek di dalam dan di sekitar sistem (termasuk pengguna, *display*, dan sebagainya) berupa *message* yang digambarkan terhadap waktu. *Sequence diagram* terdiri atar dimensi vertikal (waktu) dan dimensi horizontal (objek-objek yang terkait) [DHA-03:8].

Sequence diagram biasa digunakan untuk menggambarkan skenario atau rangkaian langkah-langkah yang dilakukan sebagai respons 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 [DHA-03:8].

Masing-masing objek, termasuk aktor, memiliki *lifeline* vertikal. *Message* digambarkan sebagai garis berpanah dari satu objek ke objek lainnya. Pada fase desain berikutnya, message akan dipetakan menjadi operasi/metoda dari *class*. *Activation bar* menunjukkan lamanya eksekusi sebuah proses, biasanya diawali dengan diterimanya sebuah *message*. Untuk objek-objek yang memiliki sifat khusus, standar UML mendefinisikan icon khusus untuk objek *boundary*, *controller* dan *persistent entity* [DHA-03:8-9].



Gambar 2.23 Contoh *sequence diagram*
Sumber: [PIL-05]

1.6.3. Pengujian (*Testing*)

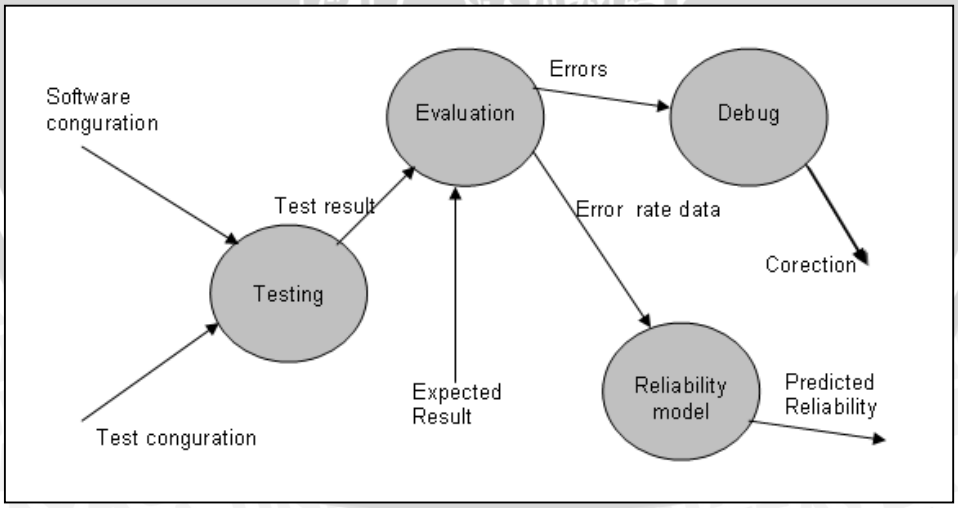
Ujicoba software merupakan elemen yang kritis dari SQA dan merepresentasikan tinjauan ulang yang menyeluruh terhadap spesifikasi, desain dan pengkodean. Ujicoba merepresentasikan ketidaknormalan yang terjadi pada pengembangan software. Selama definisi awal dan fase pembangunan, pengembang berusaha untuk membangun software dari konsep yang abstrak sampai dengan implementasi yang memungkin.

Para pengembang membuat serangkaian uji kasus yang bertujuan untuk ”membongkar” software yang mereka bangun. Ujicoba yang diperlukan oleh

pengembang adalah untuk melihat kebenaran dari software yang dibuat dan konflik yang akan terjadi bila kesalahan tidak ditemukan. Glen Myers menetapkan beberapa aturan yang dapat dilihat sebagai tujuan dari ujicoba:

1. Ujicoba merupakan proses eksekusi program dengan tujuan untuk menemukan kesalahan.
2. Sebuah ujicoba kasus yang baik adalah yang memiliki probabilitas yang tinggi dalam menemukan kesalahan-kesalahan yang belum terungkap.
3. Ujicoba yang berhasil adalah yang mengungkap kesalahan yang belum ditemukan.

Sehingga tujuan dari ujicoba ini adalah mendesain serangkaian tes yang secara sistematis mengungkap beberapa jenis kesalahan yang berbeda dan melakukannya dalam waktu dan usaha yang minimum. Jika pengujian diselenggarakan dengan sukses, maka akan membongkar kesalahan yang ada didalam perangkat lunak, manfaat lain dari pengujian adalah menunjukkan bahwa fungsi perangkat lunak telah bekerja sesuai dengan spesifikasi, dan kebutuhan fungsi telah tercapai. Sebagai tambahan, data yang dikumpulkan pada saat pengujian dilaksanakan akan menyediakan suatu indikasi keandalan perangkat lunak yang baik dan beberapa indikasi mutu perangkat lunak secara keseluruhan. Aliran informasi pada proses pengujian ditunjukkan pada Gambar 2.24.



Gambar 2.24 Aliran informasi proses pengujian
Sumber: [AYU-09]

Alur informasi untuk ujicoba mengikuti pola seperti gambar diatas. Dua kategori input yang disediakan untuk proses ujicoba adalah :

1. *Software configuration* yang terdiri dari spesifikasi kebutuhan software, spesifikasi desain dan kode sumber,
2. *Test configuration* yang terdiri dari rencana dan prosedur ujicoba, *Tools* ujicoba apapun yang dapat digunakan, dan kasus ujicoba termasuk hasil yang diharapkan. Pada kenyataannya, konfigurasi ujicoba merupakan subset dari konfigurasi software.

Setiap lingkaran merepresentasikan transformasi yang lebih kompleks. Ujicoba dilakukan dan hasilnya dievaluasi, kemudian hasil ujicoba dibandingkan dengan hasil yang diharapkan . Ketika ditemukan data yang keliru, maka *error* ditemukan dan *debug* dimulai. Ketika hasil ujicoba dikumpulkan dan dievaluasi, indikasi kualitatif dari kualitas dan reliabilitas software mulai terlihat. Jika terjadi kesalahan fatal dan memerlukan modifikasi desain ditemukan secara reguler, maka kualitas dan reliabilitas software akan dipertanyakan dan diperlukan ujicoba lanjutan.

Sebaliknya jika fungsi software bekerja sebagaimana mestinya dan kesalahan yang terjadi dapat diatasi dengan mudah maka, dapat diambil 1 dari 2 kesimpulan dapat dibuat, yaitu : (1) Kualitas dan reliabilitas software dapat diterima, atau (2) Ujicoba tidak cukup untuk menemukan kesalahan yang fatal.

Akhirnya, jika ujicoba tidak menghasilkan kesalahan, maka harus terjadi keraguan bahwa konfigurasi ujicoba tersebut tidak berhasil dan masih terjadi kesalahan dalam software. Hal ini, akan dibuktikan oleh user dan akan diperbaiki oleh pengembang dalam fase pemeliharaan. Hasil-hasil yang dikumpulkan selama ujicoba dapat dievaluasi dengan cara formal.

1.6.3.1. Teknik Pengujian

Sasaran dilakukannya pengujian adalah untuk mengungkap kesalahan dalam waktu dan usaha yang minimum. Dalam melakukan pengujian, diperlukan perancangan *test case* (kasus uji) yang akan digunakan untuk menguji perangkat lunak. Untuk melakukan perancangan *test case* tersebut, terdapat dua metode yaitu metode pengujian *white box* dan pengujian *black box.*

1.6.3.1.1. *White Box Testing*

Ujicoba *Whitebox* merupakan metode desain uji kasus yang menggunakan struktur kontrol dari desain prosedural untuk menghasilkan kasus-kasus uji.

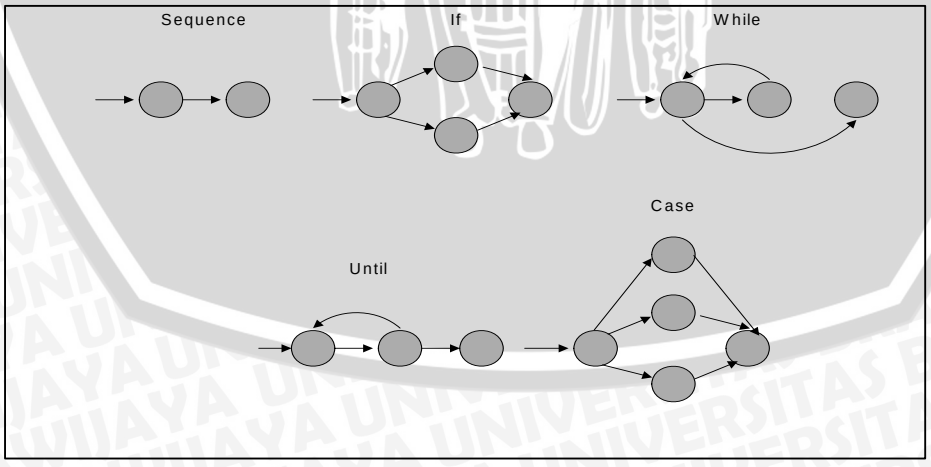
Dengan menggunakan metode ujicoba *whitebox*, para pengembang software dapat menghasilkan kasus-kasus uji yang :

a. Menjamin bahwa seluruh *independent paths* dalam modul telah dilakukan sedikitnya satu kali
b. Melakukan seluruh keputusan logikal baik dari sisi benar maupun salah
c. Melakukan seluruh perulangan sesuai batasannya dan dalam batasan operasionalnya
d. Menguji struktur data internal untuk memastikan validitasnya

Ada beberapa metode untuk merealisasikan pengujian *white box*, diantaranya adalah *basis path testing, condition testing, loop tesing* serta *data flow testing.*

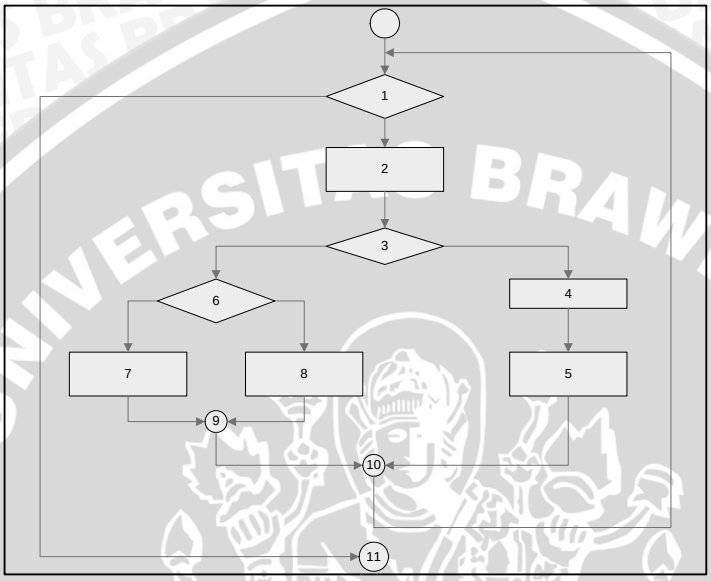
Pada skripsi ini, metode yang digunakan untuk merealisasikan *white box testing* adalah teknik *basis path testing.* Metode ini memungkinkan perancang test case mendapatkan ukuran kekompleksan logical dari perancangan prosedural dan menggunakan ukuran ini sebagai petunjuk untuk mendefinisikan basis set dari jalur pengerjaan. Test case yang didapat digunakan untuk mengerjakan basis set yang menjamin pengerjaan setiap perintah minimal satu kali selama uji coba.

Untuk melakukan teknik pengujian ini diperlukan penelusuran terhadap kontrol logika untuk menentukan *test case* dalam proses pengujian. Untuk menggambarkan aliran kontol logika, digunakan diagram alir (grafik alir) dengan notasi ditunjukkan pada Gambar 2.25.

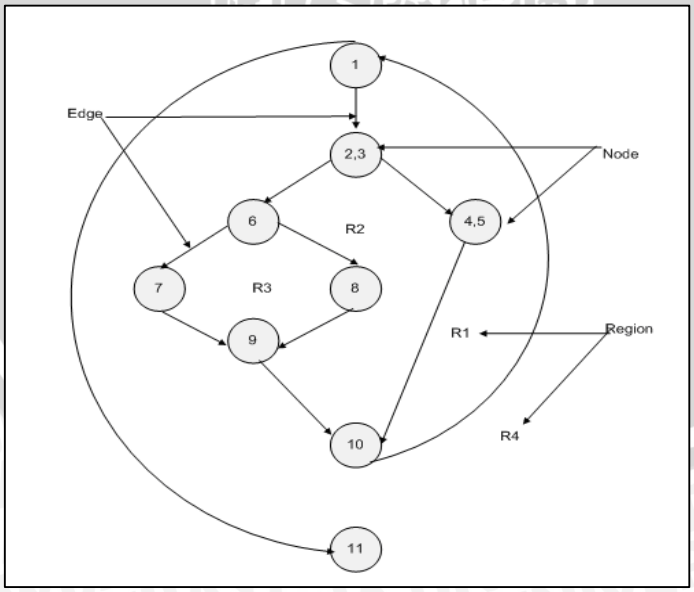


Gambar 2.25 Notasi grafik alir
Sumber: [PRE-02:446]

Salah satu cara untuk mendapatkan grafik alir adalah melalui struktur *flowchart*. Sebagai contoh, dari *flowchart* seperti yang ditunjukkan pada Gambar 2.26 bisa dimodelkan ke dalam sebuah grafik alir pada Gambar 2.27. *node* menggambarkan simbol proses, *edges* menggambarkan aliran kontrol, Area yang dibatasi oleh *edge* dan *node* disebut *region*.



Gambar 2.26 *Flowchart*
Sumber: [PRE-02:447]



Gambar 2.27 Grafik alir dari *flowchart* pada Gambar 2.26
Sumber: [PRE-02:447]

Untuk menentukan jumlah jalur independen dalam basis set suatu program, serta memberikan batas atas bagi jumlah pengujian yang harus dilakukan untuk memastikan bahwa semua pernyataan telah dieksekusi sedikitnya satu kali, digunakan kompleksitas siklomatis. Kompleksitas siklomatis merupakan metrics perangkat lunak yang memberikan pengukuran kuantitatif terhadap kompleksitas logis atas suatu program. Untuk menentukan kompleksitas siklomatis bisa dilakukan dengan beberapa cara, diantaranya:

1. Jumlah *region* grafik alir sesuai dengan kompleksitas siklomatis.
2. Kompleksitas siklomatis V(G), untuk grafik G adalah V(G) = E – N + 2, dimana E adalah jumlah *edge*, dan N adalah jumlah *node.*
3. V(G) = P + 1, dimana P adalah jumlah simpul predikat yang diisikan dalam grafik alir G.

Langkah-langkah untuk melakukan *test case* adalah sebagai berikut:

1. Dengan menggunakan desain atau *source code* sebagai dasar untuk memodelkan ke dalam grafik alir.
2. Dari hasil pemodelan grafik alir, ditentukan kompleksitas siklomatis V(G).
3. Menentukan sebuah basis set dari jalur independent secara liner. Harga kompleksitas siklomatis memberikan jumlah jalur independent secara linier melalui struktur control program.
4. Menyiapkan *test case* untuk tiap-tiap jalur.

1.6.3.1.2. *Black Box Testing*

Metode ujicoba *blackbox* memfokuskan pada keperluan fungsional dari *software*. Karna itu ujicoba *blackbox* memungkinkan pengembang *software* untuk membuat himpunan kondisi input yang akan melatih seluruh syarat-syarat fungsional suatu program. Ujicoba *blackbox* bukan merupakan alternatif dari ujicoba *whitebox,* tetapi merupakan pendekatan yang melengkapi untuk menemukan kesalahan lainnya, selain menggunakan metode *whitebox.*

Ujicoba *blackbox* berusaha untuk menemukan kesalahan dalam beberapa kategori, diantaranya :

1. Fungsi-fungsi yang salah atau hilang
2. Kesalahan interface
3. Kesalahan dalam struktur data atau akses database eksternal

4. Kesalahan performa
5. kesalahan inisialisasi dan terminasi

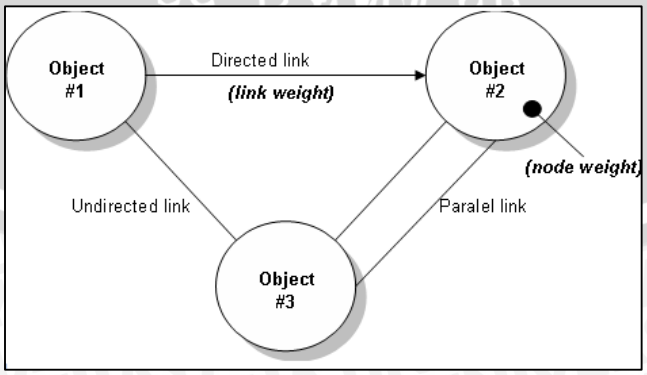
Tidak seperti metode *whitebox* yang dilaksanakan diawal proses, ujicoba *blackbox* diaplikasikan dibeberapa tahapan berikutnya. Karena ujicoba *blackbox* dengan sengaja mengabaikan struktur kontrol, sehingga perhatiannya difokuskan pada informasi domain.

Beberapa teknik yang dipakai untuk melakukan pengujian *Black Box* diantaranya adalah sebagai berikut:

a. Metode Pengujian *Graph-Based*

Langkah pertama pada *black box testing* adalah memahami objek-objek yang dimodelkan dalam perangkat lunak dan relasi-relasi yang berhubungan dengan objek tersebut. Langkah selanjutnya adalah untuk melakukan pengujian yang akan melakukan verifikasi "semua objek telah mempunyai hubungan dengan yang lain". Pengujian dimulai dengan pembuatan *graph* dari objek-objek yang penting dan hubungannya masing-masing, kemudian dilanjutkan dengan melakukan serangkaian pengujian yang akan mencakup seluruh bagian *graph*, sehingga semua objek dan relasinya diuji dan akan didapatkan *error*. Teknik ini disebut sebagai *graph based testing.*

Graph merupakan sekumpulan *nodes* yang menggambarkan objek, *links* yang menggambarkan hubungan antar objek. Keterangan tiap *nodes* menunjukkan properti tiap *nodes* (seperti sebuah nilai data yang spesifik, atau perilaku sebuah keadaan), dan keterngan *links* menunjukkan karakteristik relasi. Gambar 2.28 memberikan contoh *graph.*



Gambar 2.28 Notasi *graph*
Sumber : [PRE-02:461]

Directed link (digambarkan dengan mata anak panah) mengindikasikan hubungan yang satu arah. *Bidirectional link* atau sering juga disebut *symmetric link* menggambarkan hubungan dua arah. *Paralel link* digunakan untuk menggambarkan beberapa hubungan yang berbeda antara dua *node*

b. Metode Pengujian Partisi Ekivalensi

Partisi ekivalensi adalah metode pengujian *black box* yang membagi domain input dari suatu program ke dalam *class* data dari mana *test case* dapat dilakukan. *Test case* yang ideal mengungkap kesalahan (misalnya, pemrosesan yang tidak benar terhadap semua data karakter) yang akan memerlukan banyak kasus untuk diekseskusi sebelum kesalahan umum diamati. Partisi ekivalensi berusaha menentukan sebuah *test case* yang mengungkap *class-class* kesalahan, sehingga mengurangi jumlah total *test case* yang harus dikembangkan.

Desain *test case* pada partisi ekivalensi didasarkan pada evaluasi terhadap *class* ekivalensi untuk suatu kondisi input. *Class* ekivalensi merepresentasikan serangkaian keadaan valid atau yang tidak valid untuk kondisi input. Secara khusus, suatu kondisi input dapat berupa harga numeris, suatu rentang harga, atau serangkaian harga terkait, atau sebuah kondisi Boolean.

Class ekivalensi dapat ditentukan sesuai pedoman berikut ini:

1. Bila kondisi input menentuka suatu *range* maka suatu *class* ekivalensi valid dan dua yang tidak valid ditentukan.
2. Bila suatu kondisi input membutuhkan suatu harga khusus maka satu *class* ekivalensi valid dan dua yang tidak valid ditentukan.
3. Bila suatu kondisi menentukan anggota suatu himpunan, maka satu *class* ekivalensi valid atau dua yang tidak valid ditentukan.
4. Bila suatu kondisi input adalah Boolean, maka satu *class* valid dan satu yang tidak valid ditentukan.

Yang pertama kali dilakukan adalah identifikasi kondisi input dari suatu sistem. Kemudian dengan mengaplikasikan pedoman untuk derivasi, *test case* untuk masing-masing item data domain input dapat dikembangkan dan dieksekusi.

c. Metode Analisis Nilai Batas *(Boundary Value Analysis)*

Teknik pengujian *Boundary Value Analysis (BVA)* dikembangkan untuk memunculkan pemilihan *test case* yang menggunakan nilai batas. Teknik ini melengkapi pengujian partisi ekivalensi. BVA lebih mengarahkan kepada pemilihan *test case* pada *edge* dari *class*. Dalam banyak hal, pedoman untuk BVA sama dengan yang diberikan untuk partisi ekivalensi:

1. Bila suatu kondisi input mengkhususkan suatu *range* dibatasi oleh nilai *a* dan *b* , maka *test case* harus didesain dengan nilai *a* dan *b*, persis di atas dan di bawah *a* dan *b*, secara bersesuaian.
2. Bila suatu kondisi input mengkhususkan sejumlah nilai, maka *test case* harus dikembangkan dengan menggunakan jumlah minimum dan maksimum. Nilai tepat di atas dan di bawah minimum dan maksimum juga diuji.
3. Pedoman 1 dan 2 diaplikasikan ke kondisi output.
4. Bila struktur data program telah memesan batasan (misalnya, suatu *array* memiliki suatu batas yang ditentukan dari 100 masukan), maka *case* didesain untuk struktur data tersebut pada batasnya.

1.6.3.2. Strategi Pengujian

Strategi uji coba software memudahkan para perancang untuk menentukan keberhasilan sistem yang telah dikerjakan. Hal yang harus diperhatikan adalah langkah-langkah perencanaan dan pelaksanaan harus direncanakan dengan baik dan berapa lama waktu, upaya dan sumber daya yang diperlukan.

Strategi untuk melakukan pengujian perangkat lunak, dimulai dari dengan “pengujian kecil” bergerak menuju ke “pengujian besar”. Demikian juga dalam konteks pengujian berorientasi objek, dalam hal ini pengujian dimulai dari pengujian unit, bergerak menuju pengujian integrasi dan berakhir pada proses validasi [PRE-02:762].

a) Pengujian Unit

Pengujian unit difokuskan pada usaha verifikasi pada unit terkecil dari desain software, yakni modul. Pengujian unit selalu berorientasi pada white box testing dan dapat dikerjakan paralel atau beruntun dengan modul lainnya.

Pada saat perangkat lunak berorientasi objek diperhatikan, konsep mengenai unit menjadi berubah. Enkapsulasi mengendalikan definisi *class* dan

objek. Ini berarti bahwa masing-masing *class* dan contoh suatu *class* (objek) mengemas (data) dan operasi yang memanipulasi data-data tersebut. Selain modul individual, unit terkecil yang dapat diuji merupakan data atau objek enkapsulasi. *Class* dapat berisi sejumlah operasi yang berbeda, dan operasi khusus dapat muncul sebagai bagian dari *class-class* yang berbeda.

Class pengujian untuk menguji perangkat lunak OO ekivalen dengan pengujian unit untuk perangkat lunak konvesional. Tidak seperti pengujian unit perangkat lunak konvesional, yang cenderung berfokus pada detail algoritma dari suatu modul dan data yang mengalir pada interface modul, pengujian *class* untuk perangkat lunak OO dikendalikan oleh operasi yang dienkapsulasi oleh *class* dan tingkah laku keadaan dari *class* tersebut.

b) Pengujian Integrasi

Pengujian terintegrasi adalah teknik yang sistematis untuk penyusunan struktur program, pada saat bersamaan dikerjakan uji coba untuk memeriksa kesalahan yang nantinya digabungkan dengan interface. Terdapat dua metode pengujian yaitu *Top down integration* dan *Bottom up integration*

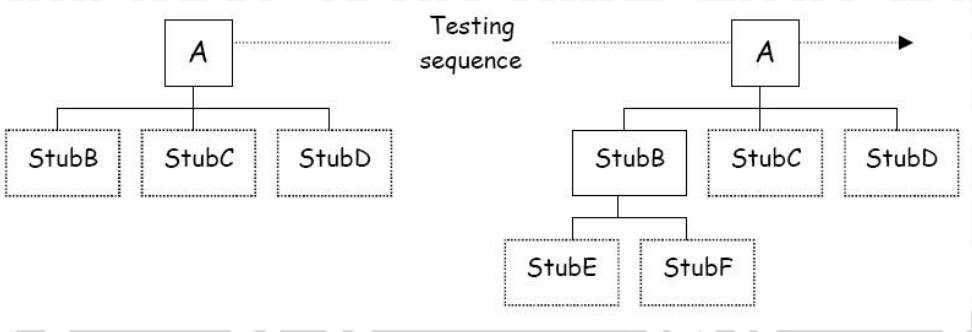
1. *Top Down Integration*

Merupakan pendekatan inkrimental untuk penyusunan struktur program. Modul dipadukan dengan bergerak ke bawah melalui kontrol hirarki dimulai dari modul utama. Modul subordinat ke modul kontrol utama digabungkan ke dalam struktur baik menurut *depth first* atau *breadth first.*

Secara rinci proses integrasi dapat dijelaskan sebagai berikut:

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

Driver adalah program yang menerima data untuk kasus uji dan menyalurkan ke modul yang diuji dan mencetak hasilnya sedangkan Stub melayani pemindahan modul yang akan dipanggil untuk diuji.



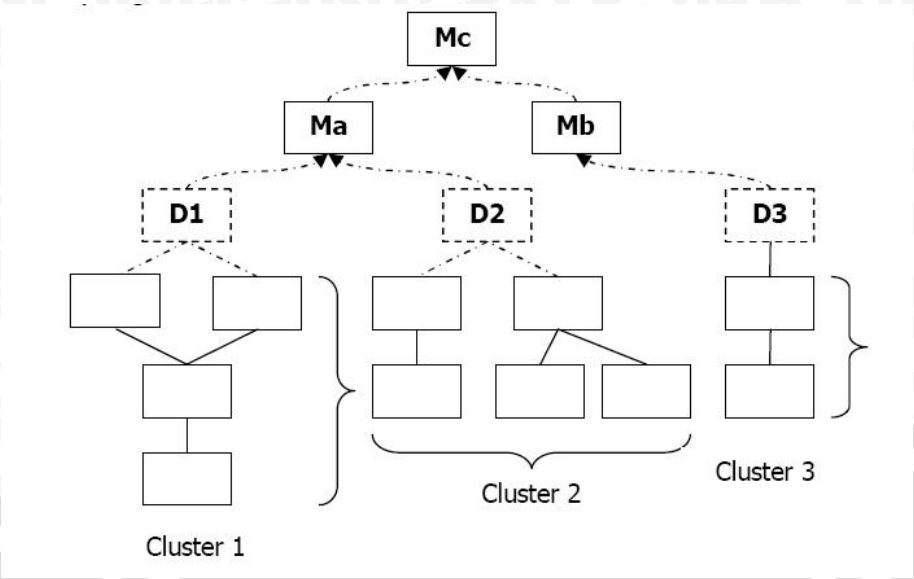
Gambar 2.29 *Top-Down Integration*
Sumber : [AYU-09]

2. Bottom Up Integration

Pengujian bottom up dinyatakan dengan penyusunan yang dimulai dan diujicobakan dengan atomic modul (yaitu modul tingkat paling bawah pada struktur program). Karena modul dipadukan dari bawah ke atas, proses yang diperlukan untuk modul subordinat yang selalu diberikan harus ada dan diperlukan untuk stub yang akan dihilangkan.

Strategi pengujian :

1. Modul tingkat bawah digabungkan ke dalam cluster yang memperlihatkan subfungsi software
2. Driver (program kontrol pengujian) ditulis untuk mengatur input test case dan output
3. Cluster diuji
4. Driver diganti dan cluster yang dikombinasikan dipindahkan ke atas pada struktur program.



Gambar 2.30 *Bottom Up Integration*
Sumber : [AYU-09]

c) Pengujian Validasi

Setelah semua kesalahan diperbaiki maka langkah selanjutnya adalah validasi testing. Pengujian validasi dikatakan berhasil bila fungsi yang ada pada software sesuai dengan yang diharapkan pemakai. Validasi software merupakan kumpulan seri uji coba black box yang menunjukkan sesuai dengan yang diperlukan.

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

BAB III
METODOLOGI PENELITIAN

Dengan mengacu pada rumusan masalah yang telah diuraikan pada Bab I, pada bab ini akan dibahas berbagai metode yang digunakan untuk merancang dan membuat Sistem layanan informasi lowongan kerja berbasis RSS melalui media SMS dengan menggunakan Protokol SMPP

. Diantaranya adalah studi literatur, metode untuk melakukan perancangan, implementasi dan pengujian. Pada akhir proses pengembangan akan ditarik kesimpulan dan saran untuk proses perbaikan ke depan.

3.1. Studi Literatur

Berupa kajian pustaka terhadap sumber-sumber bacaan yang relevan, dengan tujuan untuk mendapatkan landasan teori yang berhubungan dengan perancangan dan pembuatan aplikasi Sistem layanan informasi lowongan kerja berbasis RSS melalui media SMS dengan menggunakan Protokol SMPP beserta teori-teori pendukungnya, antara lain :

a. Aplikasi berbasis Web
   - Java Servlet
   - JSP (*Java Server Pages*)
   - Struts framework
   - MYSQL
b. RSS
c. XML (*Extensible Markup Language)*
d. Komunikasi Seluler
   - *GSM (Global System for Mobile Communication)*
   - SMS
e. Protokol SMPP
f. Metode Pengembangan Perangkat Lunak

Sumber bacaan tersebut dapat berupa *text book*, buku panduan pemrograman, tugas akhir, *paper*, tutorial pemrograman, dan sumber bacaan *softcopy* lain yang didapatkan dari Internet. Sumber-sumber bacaan tersebut diletakkan pada daftar pustaka.

3.2. Perancangan

Tahap ini terdiri dari 2 bagian, yaitu analisis kebutuhan *(Requirement Analysis)* dan desain (perancangan).

3.2.1. Analisis Kebutuhan *(Requirement Analysis)*

Tahap ini ditujukan untuk mendapatkan kebutuhan yang diperlukan dan yang ingin didapatkan dari sistem yang akan dibangun *(Requirement Capture)*, dan kemudian menganalisanya. Hasil dari analisis kebutuhan dimodelkan dalam diagram *Use Case*, dan alur proses dari masing-masing *Use Case* diberikan dalam table Skenario *Use Case.*

1. Pemodelan Diagram *Use Case*.

Perancangan *use-case* diagram diawali dengan menentukan aktor-aktor yang terlibat dalam sistem yang akan dibangun. Kemudian menentukan *action-action* apa saja yang dapat dilakukan oleh masing-masing aktor tersebut.

2. Perancangan Skenario *Use-Case*

Skenario *use case* skenario merupakan kemungkinan-kemungkinan skenario setiap *action* yang dilakukan oleh masing-masing aktor. Skenario *use case* tentunya didasarkan pada diagram *use case* yang sudah dirancang sebelumnya.

3.2.2. Perancangan (Desain)

Langkah – langkah yang dilakukan dalam merancang sistem ini adalah sebagai berikut:

1. Perancangan Sistem

Perancangan sistem digunakan untuk menentukan desain arsitektur global aplikasi yang akan dibangun.

2. Perancangan Diagram *Class*

Diagram *Class* merupakan sebuah diagram yang berisi klas-klas yang terdapat dalam sebuah sistem. Diagram ini juga menunjukkan atribut dan operasi setiap klas, dan juga hubungan antar klas.

3. Perancangan Diagram *Sequence*

Diagram *sequence* menunjukkan bagaimana urutan perilaku sistem untuk masing-masing skenario.

4. Perancangan Database

Perancangan database dimulai dengan analisis kebutuhan untuk menentukan entitas-entitas yang terlibat, kemudian menentukan atribut dari setiap entitas dan menentukan relasi antar entitas yang pada akhirnya akan terbentuk diagram ER (*Entity Relationship*).

5. Perancangan Antarmuka

Perancangan antarmuka digunakan untuk membuat desain tampilan sistem yang akan berhadapan langsung dengan pengguna. Karena sistem yang dibangun adalah sistem berbasis web, maka antarmuka didesain sebagai tampilan halaman web.

3.3. Implementasi

Tahap ini bertujuan untuk merealisasikan *blue print* sistem yang telah disusun pada tahap perancangan agar menjadi sebuah sistem nyata yang dapat digunakan. Hal ini dilakukan dengan meng-kodekan rancangan sesuai dengan metode implementasi yang digunakan. Pada tugas akhir ini, metode implementasi yang digunakan adalah Pemrograman Berorientasi Objek *(Object Oriented Programming / OOP)* dengan menggunakan bahasa pemrograman Java.

3.4. Pengujian dan Analisis

Pengujian dilakukan pada aplikasi hasil implementasi untuk memastikan bisa berfungsi dengan benar dan bisa memenuhi semua kebutuhan yanag diinginkan pada awal analisis kebutuhan. Pada proses pengujjian digunakan tiga buah strategi yaitu pengujian unit, pengujian integrasi dan pengujian validasi. Pada pengujian unit dan pengujian integrasi digunakan teknik *white box testing*, sedangkan pada pengujian validasi digunakan teknik *black box testing.*

3.5. Pengambilan Kesimpulan dan Saran

Pada tahap ini diambil kesimpulan dari hasil pengujian dan analisis terhadap sistem layanan informasi lowongan kerja yang telah dilakukan pada tahap sebelumnya.

Tahap yang paling akhir adalah pengambilan saran terhadap penelitian yang selanjutnya sehingga bisa menyempurnakan kekurangan yang ada dan mengembangkan pada tingkat pokok kajian yang lebih lanjut.

BAB IV
PERANCANGAN SISTEM

Pada bab ini akan dibahas mengenai perancangan sistem yang meliputi dua tahap, yaitu analisis kebutuhan *(requirement analysis)* dan perancangan (desain). Pada tahap analisis kebutuhan digunakan pemodelan dalam bentuk diagram *usecase.* Pada tahap perancangan digunakan pemodelan dalam bentuk diagram klas *(class diagram),* diagram sekuensial *(sequence diagram)*, perancangan basis data, dan perancangan antarmuka.

4.1. Analisis Kebutuhan

Proses analisis kebutuhan mengambil acuan dari hasil pengumpulan, pemahaman, dan penetapan kebutuhan-kebutuhan *(requirements)* yang ingin didapatkan oleh pengguna, dan harus mampu disediakan oleh sistem perangkat lunak. Pada analisis kebutuhan ini diawali dengan identifikasi aktor, penjabaran daftar kebutuhan dan kemudian memodelkannya ke dalam suatu diagram *usecase.* Analisis kebutuhan ditujukan untuk menggambarkan kebutuhan-kebutuhan yang harus disediakan oleh sistem agar dapat memecahkan permasalahan yang dihadapi oleh pengguna.

4.1.1. Identifikasi Aktor

Tahap ini mempunyai tujuan untuk melakukan identifikasi terhadap aktor-aktor yang akan berinteraksi dengan sistem. Tabel 4.1 menunjukkan tiga buah aktor beserta penjelasannya masing-masing yang merupakan hasil dari proses identifikasi aktor.

Tabel 4.1 Deskripsi aktor

Aktor	Deskripsi Aktor
Guest	Guest merupakan semua aktor pengguna yang belum terdaftar sebagai pelanggan/member. Guest hanya bisa melihat informasi lowongan kerja yang tersedia dalam sistem. Guest bisa berubah menjadi member dengan melakukan proses registrasi yang disediakan di dalam sistem itu juga.
Member	Member merupakan aktor pengguna yang sudah terautentikasi melalui *login*. Member bisa melihat informasi lowongan kerja yang tersedia dalam sistem. Member bisa berlangganan informasi lowongan kerja Via SMS.
Administrator	Administrator merupakan aktor pengguna yang sudah terautentikasi melalui *login* dan bertugas untuk mengatur fungsional sistem yang digunakan oleh Guest dan Member.

Sumber: Analisis kebutuhan

4.1.2. Daftar Kebutuhan

Daftar kebutuhan ini terdiri dari sebuah kolom yang menguraikan kebutuhan yang harus disediakan oleh sistem, dan pada kolom yang lain akan menunjukkan nama *usecase* yang akan menyediakan fungsionalitas masing-masing kebutuhan tersebut. Daftar kebutuhan sistem yang dikembangkan pada tugas akhir ini ditunjukkan pada Tabel 4.2.

Tabel 4.2 Daftar kebutuhan

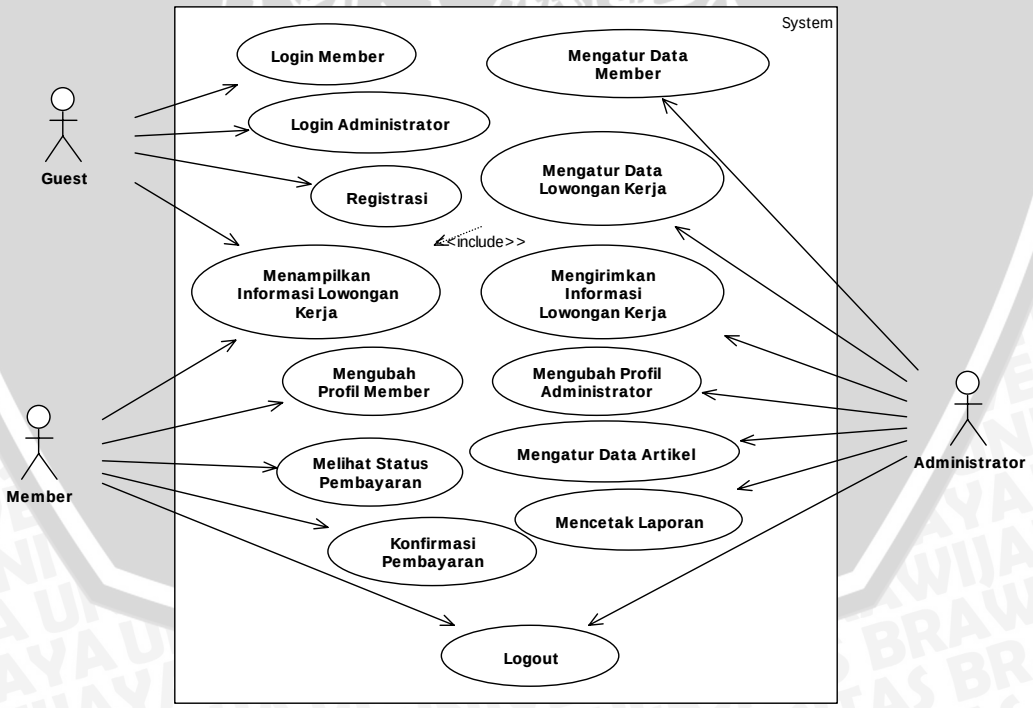
No	Requirements	Aktor	Nama *Usecase*
1	Sistem harus menyediakan fasilitas registrasi yang dapat digunakan bagi Guest untuk mendaftar dan memberikan *request* akun Member kepada sistem.	Guest	Registrasi
2	Sistem harus menyediakan fasilitas *login* yang digunakan untuk melakukan proses autentikasi terhadap pengguna yang ingin menggunakan fasilitas sistem sebagai Member	Guest	Login Member
3	Sistem harus menyediakan fasilitas *login* yang digunakan untuk melakukan proses autentikasi terhadap pengguna yang ingin menggunakan fasilitas sistem sebagai Administatror	Guest	Login Administrator
4	Sistem harus bisa menampilkan informasi lowongan kerja yang dapat diakses oleh Guest, maupun Member	Guest, Member	Menampilkan Informasi Lowongan Kerja
5	Sistem harus dapat menyediakan fasilitas bagi Member untuk mengubah profil	Member	Mengubah Profil Member
6	Sistem harus dapat menyediakan fasilitas bagi Member untuk melihat sisa deposit	Member	Melihat Sisa Deposit
7	Sistem harus dapat memberikan fasilitas bagi Member untuk melakukan konfirmasi pembayaran yang telah dilakukan sebelumnya melalui bank.	Member	Konfirmasi Pembayaran
8	Sistem harus dapat menyediakan fasilitas bagi Administrator untuk mengubah profil	Administrator	Mengubah Profil Administrator
9	Sistem harus dapat menyediakan fasilitas untuk mengupdate koleksi data Member (menambah, mengurangi, mengubah, dan mengkonfirmasi).	Administrator	Mengatur Data Member
10	Sistem harus dapat menyediakan fasilitas untuk memasukkan informasi lowongan kerja, baik yang berasal dari RSS web lain maupun dari informasi yang dimasukkan secara manual.	Administrator	Mengatur Data Informasi Lowongan Kerja
11	Sistem harus dapat menyediakan fasilitas untuk memilih dan mengirimkan data informasi lowongan kerja kepada Member.	Administrator	Mengirimkan Informasi Lowongan Kerja
12	Sistem harus dapat menyediakan fasilitas untuk memasukkan artikel yang berupa kabar maupun tips dan menampilkannya	Administrator	Manajemen Artikel

13	Sistem harus dapat menyediakan fasilitas untuk mencetak laporan baik data Member, lowongan kerja maupun keuangan.	Administrator	Mencetak Laporan
14	Sistem harus dapat menyediakan fasilitas *logout* yang digunakan untuk menghapus session Member atau Administrator dan keluar dari menu utama Member atau menu utama Administrator.	Member, Administrator	*Logout*

Sumber: Analisis Kebutuhan

4.1.3. Diagram *Usecase*

Kebutuhan-kebutuhan fungsional yang diperlukan oleh pengguna dan harus disediakan oleh sistem akan dimodelkan pada diagram *usecase*. Pada sistem ini terdapat dua belas buah *usecase*, yaitu *usecase* Registrasi, Login Member, Login Administrator, Menampilkan Informasi lowongan Kerja, Mengubah Profil Member, Melihat Sisa Deposit, Konfirmasi Pembayaran, Mengubah Profil Administrator, Mengatur Data Member, Mengatur Data Informasi Lowongan Kerja, Mengirimkan Informasi lowongan Kerja, dan *Logout*. Gambar 4.1 merupakan diagram *usecase* sistem ini.



Gambar 4.1 Diagram *Usecase* Sistem
Sumber : Analisis Kebutuhan

4.1.4. Skenario *Usecase*

Secara lebih mendetail, masing-masing *usecase* yang terdapat pada diagram *usecase*, dijabarkan dalam skenario *usecase*. Di dalam skenario *usecase*, akan diberikan uraian nama *usecase*, aktor yang berhubungan dengan *usecase* tersebut, tujuan dari *usecase*, deskripsi global tentang *usecase*, pra-kondisi yang harus dipenuhi dan pos-kondisi yang diharapkan setelah berjalannya fungsional *usecase*. Selain itu juga akan diberikan ulasan yang berkaitan dengan tanggapan dari sistem atas suatu aksi yang diberikan oleh aktor (aliran utama), serta kejadian alternatif yang akan terjadi jika suatu kondisi tidak bisa terpenuhi (aliran alternatif).

Kebutuhan fungsional untuk melakukan registrasi dipresentasikan oleh *usecase* Registrasi. Skenario *usecase* Registrasi ditunjukkan pada Tabel 4.3.

Tabel 4.3 Skenario *Usecase* Registrasi

Usecase	Registrasi
Aktor	Guest
Tujuan	Memberikan permintaan akun Member dari Guest kepada sistem.
Deskripsi	Untuk melakukan proses registrasi, aktor harus mengisikan data-data pribadi yang diperlukan dan melengkapi persyaratan sebagai member.
Pra-kondisi	Halaman utama web harus dalam keadaan aktif atau aktor masuk ke dalam halaman Registrasi.
Pos-kondisi	Sistem mendapatkan permintaan akun Member dari Guest, yang kemudian akan dilakukan verifikasi dan konfirmasi melalui e-mail setelah permintaan tersebut di-*approve* oleh Administrator.
Aliran Utama	
Aksi dari Aktor	**Tanggapan dari Sistem**
1. Pada menu utama, aktor memilih untuk melakukan proses registrasi dengan cara menekan pilihan ”Daftar”.	2. Menampilkan halaman ”registrasi” yang terdiri dari beberapa *field* yang harus diisi data calon member. *Field* isian diantaranya adalah Nama Lengkap, Alamat, E-Mail, Tempat/Tanggal Lahir, Nomor HP, Level Pendidikan Terakhir, Universitas, Fakultas, Jurusan, Konsentrasi, Username, Password dan Konfirmasi Password. Serta terdapat tombol ”Daftar” dan ”Batal”.
3. Aktor mengisi *field* berupa data pribadi kemudian menekan tombol ”Daftar”.	4. Memasukkan data calon Member ke database calon Member dan menampilkan pernyataan bahwa Administrator akan memberikan konfirmasi melalui email.
Aliran Alternatif 1: Eksepsi jika terdapat *field* yang kosong	
1. Aktor tidak melengkapi semua *field* isian dan menekan tombol ”Daftar”	2. Menampilkan pernyataan peringatan kesalahan bahwa terdapat kesalahan berupa isian *field* yang belum lengkap
Aliran Alternatif 2: Eksepsi jika terdapat isian *fied* E-Mail dengan format yang tidak tepat	

1. Aktor mengisi *field* dengan format isian yang salah pada format E-Mail dan menekan tombol ”Daftar”	2. Menampilkan pernyataan peringatan kesalahan bahwa terdapat kesalahan berupa kesalahan pada format E-Mail yang telah diisikan.
Aliran Alternatif 3: Eksepsi jika terdapat isian *fied* Nomor HP dengan format yang tidak tepat	
1. Aktor mengisi *field* dengan format isian yang salah pada format Nomor HP dan menekan tombol ”Daftar”	2. Menampilkan pernyataan peringatan kesalahan bahwa terdapat kesalahan berupa kesalahan pada format Nomor HP yang telah diisikan.
Aliran Alternatif 4: Eksepsi jika terdapat isian *fied* Username dengan format yang tidak tepat	
1. Aktor mengisi *field* dengan format isian yang salah dalam Username dan menekan tombol ”Daftar”	2. Menampilkan pernyataan peringatan kesalahan bahwa terdapat kesalahan berupa kesalahan pada format Username yang telah diisikan.
Aliran Alternatif 5: Eksepsi jika terdapat isian *fied* Username yang sama dengan Username yang sudah terdaftar sebelumnya	
1. Aktor mengisi *field* dengan format isian yang salah dan menekan tombol ”Daftar”	2. Menampilkan pernyataan peringatan bahwa Username yang sudah dimasukkan sudah terpakai
Aliran Alternatif 6: Eksepsi jika isian Konfirmasi Password tidak sama dengan Password	
1. Aktor mengisi *field* dengan format isian yang salah pada Konfirmasi Password dan menekan tombol ”Daftar”	2. Menampilkan pernyataan peringatan kesalahan bahwa Konfirmasi Password tidak sesuai dengan Password
Aliran Alternatif 7: Jika ditekan tombol ”Batal”	
1. Aktor menekan tombol ”Batal”	2. Membatalkan proses registrasi dan kembali ke halam utama web.

Sumber: Analisis Kebutuhan

Kebutuhan fungsional selanjutnya yang harus disediakan oleh sistem adalah kebutuhan untuk melakukan autentikasi pengguna yang akan menggunakan fasilitas sistem sebagai Member. Kebutuhan tersebut direpresentasikan oleh *usecase* Login Member. Skenario *usecase* Login Member ditunjukkan pada Tabel 4.4.

Tabel 4.4 Skenario *Usecase* Login Member

Usecase	Login Member
Aktor	Guest
Tujuan	Melakukan autentikasi pengguna yang akan menggunakan fasilitas sistem sebagai Member.
Deskripsi	Untuk memulai proses *login*, terlebih dahulu aktor harus memasukkan *field* isian “username” dan “password”, setelah itu proses dilakukan.
Pra-kondisi	Halaman utama web dalam keadaan aktif
Pos-kondisi	Guest berubah status menjadi Member dan halaman Member dalam keadaan aktif.
Aliran Utama	
Aksi dari Aktor	**Tanggapan dari Sistem**
1. Pada halaman utama web, pada isian Login, aktor	2. Melakukan proses validasi akun yang telah

memasukkan username dan password. Kemudian menekan tombol Login.	dimasukkan oleh aktor.
	3. Jika aktor teridentifikasi sebagai Member maka sistem menampilkan halaman Member yang berisi menu ▪ ”Edit Akun”, untuk mengubah data member, “username” dan “password”. ▪ “Status Pembayaran”, untuk melihat sisa deposit dari member. ▪ ”Konfirmasi Pembayaran”, Untuk memberi konfirmasi kepada administrator bahwa member telah melakukan pembayaran ▪ ”*Logout*”, untuk keluar dari menu utama Member.
Aliran Alternatif 1: Eksepsi jika terdapat *field* yang kosong	
1. Aktor tidak melengkapi *field* Login	2. Menampilkan pernyataan peringatan kesalahan bahwa terdapat kesalahan berupa isian *field* yang belum lengkap.
Aliran Alternatif 2: Eksepsi *user* tidak valid	
1. Isian Username dan Password tidak valid dengan data Member	2. Menampilkan pernyataan peringatan bahwa kombinasi Username dan Password tidak terdapat pada daftar Member.

Sumber: Analisis Kebutuhan

Kebutuhan fungsional selanjutnya yang harus disediakan oleh sistem adalah kebutuhan untuk melakukan autentikasi pengguna yang akan menggunakan fasilitas sistem sebagai Administrator. Kebutuhan tersebut direpresentasikan oleh *usecase* Login Administrator. Skenario *usecase* Login Administrator ditunjukkan pada Tabel 4.5.

.**Tabel 4.5** Skenario *Usecase* Login Administrator

Usecase	Login Administrator
Aktor	Guest
Tujuan	Melakukan autentikasi pengguna yang akan menggunakan fasilitas sistem sebagai Administrator.
Deskripsi	Untuk memulai proses *login* Administrator, terlebih dahulu aktor harus memasukkan *field* isian “username” dan “password”, setelah itu proses dilakukan.
Pra-kondisi	Halaman utama web dalam keadaan aktif.
Pos-kondisi	Guest berubah status menjadi Administrator dan halaman Administator dalam keadaan aktif.
Aliran Utama	
Aksi dari Aktor	**Tanggapan dari Sistem**
1. Pada *browser*, aktor mengetikkan alamat *url* halaman ”login administrator” lalu menekan enter.	2. Menampilkan halaman ”login administrator” yang terdiri atas *field* ”username” dan ”password” serta tombol ”Login” untuk melakukan proses validasi.
3. Aktor mengisi *field* ”username” dan ”password”	4. Melakukan proses validasi *field* isian ”username” dan password untuk memeriksa

kemudian menekan tombol ”Login”.	aktor berhak menggunakan fasilitas sistem atau tidak. Jika aktor teridentifikasi sebagai Administrator maka ditampilkan menu utama Administrator antara lain: ▪ ”Edit Akun”, untuk mengubah nilai *email, username*, dan *password* Administrator. ▪ ”Manajemen Data Member”, untuk mengatur data member yaitu mengkonfirmasi pembayaran dari member, melihat data tanggungan member, menampilkan , menambah, mengubah dan menghapus data member ▪ ”Manajemen Informasi Lowongan Kerja” untuk mengatur penyeleksian informasi lowongan kerja yang akan masuk ke *database* baik secara otomatis maupun manual. ▪ ”Kirim SMS” untuk mengirim informasi lowongan kerja ke *handphone* member. ▪ ”*Logout*”, untuk keluar dari menu utama Administrator.
Aliran Alternatif 1: Eksepsi jika terdapat *field* yang kosong	
1. Aktor tidak melengkapi *field* Login	2. Menampilkan pernyataan peringatan kesalahan bahwa terdapat kesalahan berupa isian *field* yang belum lengkap.
Aliran Alternatif 2: Eksepsi *user* tidak valid	
1. Isian Username dan Password tidak valid dengan data Administrator	2. Menampilkan pernyataan peringatan bahwa kombinasi Username dan Password tidak terdapat pada daftar Administrator.

Sumber: Analisis Kebutuhan

Kebutuhan fungsional agar sistem dapat menyediakan fasilitas bagi Member atau Guest untuk menampilkan informasi lowongan kerja, direpresentasikan oleh *usecase* Menampilkan Informasi Lowongan Kerja. Tabel 4.6 merupakan skenario *usecase* Menampilkan Informasi Lowongan Kerja.

Tabel 4.6 Skenario *Usecase* Menampilkan Informasi Lowongan Kerja

Usecase	Menampilkan Informasi Lowongan Kerja
Aktor	Guest, Member
Tujuan	Menampilkan informasi lowongan kerja yang telah diseleksi oleh Administrator
Deskripsi	Fungsionalitas ini disediakan dalam sebuah halaman web yang khusus untuk menampilkan semua daftar lowongan kerja.
Pra-kondisi	Halaman utama web dalam keadaan aktif.
Pos-kondisi	Ditampilkan informasi lowongan kerja.
Aliran Utama	
Aksi dari Aktor	**Tanggapan dari Sistem**
1. Pada halaman utama, aktor memilih menu untuk menampilkan lowongan Kerja dengan cara menekan pilihan ”Lowongan”.	2. Menampilkan halaman yang berisi dua buah kategori yaitu ”Daftar Loker RSS Baru” yang berisi daftar informasi lowongan kerja yang didapat dari RSS web lain dan ”Daftar Loker Hasil Input Manual” yang berisi informasi lowongan kerja yang diinputkan secara

	manual oleh administrator.
Aliran Alternatif 1 : Aktor melihat daftar loker RSS baru	
1. Aktor melihat informasi lowongan kerja dengan menekan piihan ”Daftar Loker RSS Baru”	2. Menampilkan halaman yang berisi daftar informasi lowongan kerja yang didapat dari RSS web lain
Aliran Alternatif 2 : Aktor melihat daftar loker hasil input manual	
1. Aktor melihat informasi lowongan kerja dengan menekan piihan ”Daftar Loker Hasil Input Manual”	2. Menampilkan halaman yang berisi informasi lowongan kerja yang diinputkan secara manual oleh administrator.
Aliran Alternatif 3 : Aktor melihat detail informasi lowongan kerja	
1. Aktor bisa memilih salah satu informasi lowongan kerja dan melihat detailnya dengan menekan judul informasi lowongan kerja	2. Menampilkan informasi lowongan kerja secara detail

Sumber: Analisis Kebutuhan

Kebutuhan fungsional agar sistem dapat menyediakan fasilitas bagi Member untuk mengubah profilnya, direpresentasikan oleh *usecase* Mengubah Profil Member. Skenario *usecase* Mengubah Profil Member ditunjukkan pada Tabel 4.7.

Tabel 4.7 Skenario *Usecase* Mengubah Profil Member

Usecase	Mengubah Profil Member
Aktor	Member
Tujuan	Memberikan fasilitas kepada Member untuk mengedit profil
Deskripsi	Setelah Member berhasil login, Member bisa mengubah profil yang sebelumnya sudah dimasukkan pada waktu registrasi.
Pra-kondisi	Halaman Member sudah dalam keadaan aktif.
Pos-kondisi	Data profil Member berubah sesuai dengan data yang dimasukkan terakhir.
Aliran Utama	
Aksi dari Aktor	**Tanggapan dari Sistem**
1. Pada halaman utama Member, Aktor memilih menu untuk ”Ubah Akun”	2. Menampilkan halaman untuk mengupdate data Member yang terdiri dari beberapa *field,* yaitu Nama Lengkap, Alamat, E-Mail, Tempat/Tanggal Lahir, Nomor HP, Level Pendidikan Terakhir, Universitas, Fakultas, Jurusan, Konsentrasi, Username, Password dan Konfirmasi Password. Serta terdapat tombol ”Ubah” dan ”Batal”.
3. Aktor memasukkan data yang perlu diubah, kemudian menekan tombol ”Simpan”	4. Menyimpan data profil Member
Aliran Alternatif 1: Eksepsi jika terdapat *field* yang kosong	
1. Aktor tidak melengkapi semua *field* isian dan menekan tombol ”Simpan”	2. Menampilkan pernyataan peringatan kesalahan bahwa terdapat kesalahan berupa isian *field* yang belum lengkap.
Aliran Alternatif 2: Eksepsi jika terdapat isian *fied* E-Mail dengan format yang tidak tepat	
1. Aktor mengisi *field* dengan format isian yang salah	2. Menampilkan pernyataan peringatan

pada format E-Mail dan menekan tombol ”Simpan”	kesalahan bahwa terdapat kesalahan berupa kesalahan pada format E-Mail yang telah diisikan.
Aliran Alternatif 3: Eksepsi jika terdapat isian *fied* Nomor HP dengan format yang tidak tepat	
1. Aktor mengisi *field* dengan format isian yang salah pada format Nomor HP dan menekan tombol ” Simpan”	2. Menampilkan pernyataan peringatan kesalahan bahwa terdapat kesalahan berupa kesalahan pada format Nomor HP yang telah diisikan.
Aliran Alternatif 4: Eksepsi jika terdapat isian *fied* Username dengan format yang tidak tepat	
1. Aktor mengisi *field* dengan format isian yang salah dalam Username dan menekan tombol ” Simpan”	2. Menampilkan pernyataan peringatan kesalahan bahwa terdapat kesalahan berupa kesalahan pada format Username yang telah diisikan.
Aliran Alternatif 5: Eksepsi jika terdapat isian *fied* Username yang sama dengan Username yang sudah terdaftar sebelumnya	
1. Aktor mengisi *field* dengan format isian yang salah dan menekan tombol ” Simpan”	2. Menampilkan pernyataan peringatan bahwa Username yang sudah dimasukkan sudah terpakai
Aliran Alternatif 6: Eksepsi jika isian Konfirmasi Password tidak sama dengan Password	
1. Aktor mengisi *field* dengan format isian yang salah pada Konfirmasi Password dan menekan tombol ” Simpan”	2. Menampilkan pernyataan peringatan kesalahan bahwa Konfirmasi Password tidak sesuai dengan Password
Aliran Alternatif 7: Jika ditekan tombol ”Batal”	
1. Aktor menekan tombol ”Batal”	2. Membatalkan proses edit akun dan kembali ke halaman utama aktor.

Sumber: Analisis Kebutuhan

Kebutuhan fungsional agar sistem dapat menyediakan fasilitas bagi Member untuk melihat sisa deposit, direpresentasikan oleh *usecase* Melihat Status Pembayaran. Tabel 4.8 merupakan skenario untuk *usecase* Melihat Status Pembayaran.

Tabel 4.8 Skenario *Usecase* Melihat Status Pembayaran

Usecase	Melihat Sisa Deposit
Aktor	Member
Tujuan	Menyediakan fasilitas bagi Member untuk melihat sisa deposit
Deskripsi	Setelah Member berhasil login, Member dapat mengecek status pembayaran dengan melihat sisa deposit
Pra-kondisi	Halaman Member dalam keadaan aktif.
Pos-kondisi	Ditampilkan status pembayaran member
Aliran Utama	
Aksi dari Aktor	**Tanggapan dari Sistem**
1. Pada halaman utama Member, aktor memilih menu ”Status Pembayaran”	2. Menampilkan halaman status pembayaran yang terdiri dari tanggal member daftar, tanggal member terakhir bayar, dan sisa deposit.

Sumber: Analisis Kebutuhan

Kebutuhan fungsional agar sistem dapat menyediakan fasilitas bagi Member untuk melakukan konfirmasi pembayaran, direpresentasikan oleh *usecase* Konfirmasi Pembayaran. Skenario untuk *usecase* Konfirmasi Pembayaran ditunjukkan pada Tabel 4.9.

Tabel 4.9 Skenario *Usecase* Konfirmasi Pembayaran

Usecase	Konfirmasi Pembayaran
Aktor	Member
Tujuan	Menyediakan fasilitas bagi Member untuk memberikan konfirmasi berkaitan dengan pembayaran yang telah dilakukan oleh Member melalui bank.
Deskripsi	Sebelum memberikan konfirmasi pembayaran, Member melakukan pembayaran terlebih dulu melalui bank (dengan mentransfer ke rekening administrator), kemudian memberikan data bukti transfer melalui halaman "Konfirmasi Pembayaran"
Pra-kondisi	Halaman Member dalam keadaan aktif.
Pos-kondisi	Sistem mendapatkan data konfirmasi pembayaran dari Member
Aliran Utama	
Aksi dari Aktor	**Tanggapan dari Sistem**
1. Pada halaman utama Member, aktor memilih menu "Konfirmasi Pembayaran"	2. Menampilkan halaman konfirmasi pembayaran yang di dalamnya terdapat *field* Tanggal bayar, Jumlah bayar, No.rekening pengirim.
3. Aktor mengisi semua *field* isian dan menekan tombol "Konfirmasi"	4. Sistem mengirimkan data tersebut ke basis data di server
Aliran Alternatif 1: Eksepsi jika terdapat *field* yang kosong	
1. Aktor tidak melengkapi semua *field*	2. Menampilkan pernyataan peringatan bahwa semua field harus diisi.

Sumber: Analisis Kebutuhan

Kebutuhan fungsional agar sistem dapat menyediakan fasilitas bagi Administrator untuk mengubah profilnya, direpresentasikan oleh *usecase* Mengubah Profil Administrator. Skenario *usecase* Mengubah Profil Administrator ditunjukkan pada Tabel 4.10.

Tabel 4.10 Skenario *Usecase* Mengubah Profil Administrator

Usecase	Mengubah Profil Administrator
Aktor	Administrator
Tujuan	Memberikan fasilitas kepada Administrator untuk mengedit profil
Deskripsi	Setelah Administrator berhasil login, Administrator bisa mengubah profil yang sebelumnya sudah dimasukkan pada waktu registrasi.
Pra-kondisi	Halaman Administrator sudah dalam keadaan aktif.
Pos-kondisi	Data profil Administrator berubah sesuai dengan data yang dimasukkan terakhir.
Aliran Utama	

Aksi dari Aktor	Tanggapan dari Sistem
1. Pada halaman utama Administrator, Aktor memilih menu untuk mengubah profil dengan cara memilih tombol ”Akun”	2. Menampilkan halaman untuk mengupdate data Administrator yang terdiri dari beberapa *field,* yaitu Nama Lengkap, Alamat, E-Mail, Username, Password dan Konfirmasi Password. Serta terdapat tombol ”Ubah” dan ”Reset”.
3. Aktor memasukkan data yang perlu diubah, kemudian menekan tombol ”Simpan”	4. Menyimpan data profil Administrator
Aliran Alternatif 1: Eksepsi jika terdapat *field* yang kosong	
1. Aktor tidak melengkapi semua *field* isian dan menekan tombol ”Simpan”	2. Menampilkan pernyataan peringatan kesalahan bahwa terdapat kesalahan berupa isian *field* yang belum lengkap.
Aliran Alternatif 2: Eksepsi jika terdapat isian *fied* E-Mail dengan format yang tidak tepat	
1. Aktor mengisi *field* dengan format isian yang salah pada format E-Mail dan menekan tombol ”Simpan”	2. Menampilkan pernyataan peringatan kesalahan bahwa terdapat kesalahan berupa kesalahan pada format E-Mail yang telah diisikan.
Aliran Alternatif 3: Eksepsi jika isian Konfirmasi Password tidak sama dengan Password	
1. Aktor mengisi *field* dengan format isian yang salah pada Konfirmasi Password dan menekan tombol ” Simpan”	2. Menampilkan pernyataan peringatan kesalahan bahwa Konfirmasi Password tidak sesuai dengan Password
Aliran Alternatif 4: Jika ditekan tombol ”Reset”	
1. Aktor menekan tombol ”Batal”	2. Membatalkan proses edit akun dan kembali ke halaman utama aktor.

Sumber: Analisis Kebutuhan

Kebutuhan fungsional agar sistem dapat menyediakan fasilitas bagi Member untuk mengatur data member, direpresentasikan oleh *usecase* Mengatur Data Member. Tabel 4.11 merupakan skenario untuk *usecase* Mengatur Data Member.

Tabel 4.11 Skenario *Usecase* Mengatur Data Member.

Usecase	Mengatur Data Member
Aktor	Administrator
Tujuan	Untuk menyediakan fasilitas bagi Administrator untuk mengatur data member
Deskripsi	Fungsionalitas ini disediakan bagi Administrator untuk mengkonfirmasi pembayaran member, melihat tanggungan member, menambah atau mengurangi data Member. Ketika ada permintaan akun Member dari Guest, Administrator dapat meng-approve permintaan tersebut dan memasukkan Guest ke daftar Member melalui fungsionalitas ini.
Pra-kondisi	Halaman Administrator dalam keadaan aktif.
Pos-kondisi	Ada perubahan di koleksi data Member.
Aliran Utama	
Aksi dari Aktor	**Tanggapan dari Sistem**
1. Pada halaman Administrator, aktor masuk ke halaman Atur Data Member dengan memilih menu ”Member”.	2. Menampilkan halaman yang berisi empat buah submenu, yaitu Lihat konfirmasi pembayaran member, Lihat Tanggungan Member, Data

	Calon Member dan Data Member. Pada isi halman ini menampilkan data konfirmasi bayar yang baru dari member. Konfirmasi pembayaran berisi data member yang telah membayar dan menunggu konfirmasi dari Administrator, Lihat Tanggungan Member berisi data member yang belum membayar/sisa deposit 0, Data Calon Member merupakan data pelanggan yang masih melakukan permintaan akun Member dan belum di-approve oleh Administrator. Data Member berisi data pelanggan yang sudah menjadi Member.
3. Aktor bisa memilih salah satu member dan memutuskan untuk ”konfirmasi” atau ”Hapus”	4. Jika ”Konfirmasi” berarti sistem akan melakukan update deposit member. Jika ”Hapus” berarti sistem menghapus data member
Aliran Alternatif 1 : Aktor melihat Tanggungan Member	
1. Aktor Memilih menu ”Lihat Tanggungan Member”	2. Menampilkan halaman yang berisi data member yang belum membayar atau memliki deposit 0
Aliran Alternatif 2 : Aktor melihat Data Calon Member	
1. Aktor Memilih menu ”Data Calon Member”	2. Menampilkan halaman yang berisi koleksi data calon Member. Aktor bisa memilih salah satu data untuk dilihat detail profilnya, dan bisa memutuskan untuk menerima sebagai Member atau menolaknya.
3. Aktor memilih salah satu Calon Member	4. Menampilkan detail profil dari Calon Member tersebut.
5. Aktor memilih ”Menerima” atau ”Menolak”	6. Jika ”Menerima”, maka data Calon Member ditambahkan ke database Member dan menghapusnya dari Calon Member. Serta mengirimkan konfirmasi melalui E-Mail bahwa permintaan akun sudah diterima dan diaktifkan. Jika ”Menolak” berarti data Calon Member dihapus tanpa dimasukkan ke Member, dan diberikan konfirmasi penolakan atas permintaan akunnya.
Aliran Alternatif 3 : Aktor melihat Data Member	
1. Aktor Memilih menu ”Data Member”	2. Menampilkan halaman yang berisi koleksi data Member.
3. Aktor bisa memilih salah satu Member untuk dilihat detail profilnya	4. Menampilkan halaman yang berisi detail profil dari Member, disertai pilihan untuk ”Lihat detil member” atau ”Hapus”
5. Aktor memilih ” Lihat detil member” atau ”Hapus”	6. Jika ” Lihat detil member” berarti sistem menampilkan detail data member. Jika ”Hapus” berarti sistem menghapus data dari database Member.

Sumber: Analisis Kebutuhan

Kebutuhan fungsional agar sistem dapat menyediakan fasilitas bagi Member untuk mengatur data informasi lowongan kerja, direpresentasikan oleh *usecase* Mengatur Data Informasi Lowongan Kerja. Tabel 4.12 merupakan skenario untuk *usecase* Mengatur Data Informasi Lowongan Kerja.

Tabel 4.12 Skenario *Usecase* Mengatur Data Informasi Lowongan Kerja

Usecase	Mengatur Data Informasi Lowongan Kerja
Aktor	Administrator
Tujuan	Untuk menambahkan data informasi lowongan kerja, baik dari data RSS yang terbaca dari web lain serta informasi lowongan kerja yang dimasukkan secara manual (bukan dari RSS web lain) oleh Administrator.
Deskripsi	Data RSS info lowongan kerja yang didapat dari web lain, tidak otomatis masuk ke dalam daftar info lowongan kerja. Tapi Administrator yang akan meng-approve info lowongan yang sesuai untuk dimasukkan ke dalam daftar info lowongan. Daftar lowongan kerja ini yang akan ditampilkan dan diinformasikan ke pelanggan. Selain itu Administrator juga bisa menghapus suatu informasi lowongan kerja yang telah masuk dalam daftar.
Pra-kondisi	Halaman Administrator dalam keadaan aktif.
Pos-kondisi	Terjadi perubahan (penambahan ataupun pengurangan) pada daftar informasi lowongan kerja.
Aliran Utama	
1. Pada halaman Administrator, aktor masuk ke halaman atur data informasi lowongan kerja dengan memilih menu ”Loker”.	2. Menampilkan halaman ”Data RSS Loker Baru”, yang berisi empat buah submenu, yaitu : • ”Data RSS Loker Baru” yang berisi informasi lowongan baru yang didapat dari membaca RSS web lain dan belum dikualifikasikan dalam spesifikasi akademik. • ”Data RSS Loker Tersimpan” merupakan kumpulan informasi lowongan kerja yang telah diseleksi oleh Administrator dan telah diberi kualifikasi akademik. • ”Input Loker Manual” yaitu halaman yang digunakan untuk memasukkan informasi lowongan kerja secara manual oleh Administrator • ”Lihat loker Hasil Input Manual” merupakan kumpulan informasi lowongan kerja yang diinputkan secara manual oleh Administrator. Pada halaman ini ditampilkan data lowongan kerja dari RSS yang baru. Pada masing-masing item yang ditampilkan, terdapat 2 opsi yaitu ”Simpan” dan ”Hapus”
3. Pada halaman ”Data RSS Loker Baru” ini, aktor memilih untuk menyimpan informasi lowongan kerja dengan cara menekan pilihan ”simpan”.	4. Menampilkan halaman yang terdiri dari beberapa *field* yang harus diisi. *Field* isian diantaranya adalah perusahaan, batas akhir, posisi dan kualifikasi akademik, serta terdapat tombol ”simpan”
5. Aktor mengisi semua field kemudian menekan tombol ”Simpan”	6. Melakukan proses penyimpanan ke database
Aliran Alternatif 1: Eksepsi jika terdapat *field* yang kosong	
1. Aktor menekan tombol ”Simpan” tanpa mengisi *field* (langkah ke-3 pada aliran alternatif 1).	2. Menampilkan pernyataan peringatan kesalahan bahwa terdapat kesalahan berupa isian *field* kosong.
Aliran Alternatif 2 : Aktor melihat Data RSS Loker Tersimpan	
1. Aktor Memilih menu ”Data RSS Loker Tersimpan”	2. Menampilkan halaman yang berisi informasi lowongan kerja yang didapat dari RSS web lain dan sudah diseleksi dan diberi kualifikasi akademik oleh Administrator
3. Aktor bisa memilih salah satu informasi lowongan kerja untuk dilihat detailnya dengan cara menekan	4. Menampilkan halaman web RSS tersebut

judul informasi tersebut	berasal
5. Aktor bisa mengedit atau menghapus informasi lowongan kerja dan memilih ”Edit” atau ”Hapus”	6. Jika ”Edit” berarti aktor bisa merubah data meliputi perusahaan, batas akhir, posisi dan kualifikasi akademik dan menyimpan perubahan tersebut. Jika ”Hapus” berarti sistem menghapus data dari database Loker tersimpan.
Aliran Alternatif 3 : Aktor memasukkan informasi lowongan kerja secara manual	
1. Aktor memilih menu ”Input Loker Manual”	2. Menampilkan halaman yang terdiri dari beberapa *field* yang harus diisi. *Field* isian diantaranya adalah perusahaan, batas akhir, posisi dan kualifikasi akademik, detail serta terdapat tombol ”simpan”
3. Aktor mengisi semua field kemudian menekan tombol ”Simpan”	4. Melakukan proses penyimpanan ke database
Aliran Alternatif 4: Eksepsi jika terdapat *field* yang kosong	
3. Aktor menekan tombol ”Simpan” tanpa mengisi *field* (langkah ke-3 pada aliran alternatif 4).	4. Menampilkan pernyataan peringatan kesalahan bahwa terdapat kesalahan berupa isian *field* kosong.
Aliran Alternatif 5 : Aktor melihat Loker Hasil input Manual	
1. Aktor Memilih menu ”Lihat Loker Hasil Input Manual”	2. Menampilkan halaman yang berisi informasi lowongan kerja yang diinputkan secara manual oleh Administrator.
3. Aktor bisa memilih salah satu informasi lowongan kerja untuk dilihat detailnya dengan cara menekan judul informasi tersebut	4. Menampilkan halaman informasi lowongan kerja secara detail
5. Aktor bisa mengedit atau menghapus informasi lowongan kerja dan memilih ”Edit” atau ”Hapus”	6. Jika ”Edit” berarti aktor bisa merubah data meliputi perusahaan, batas akhir, posisi dan kualifikasi akademik dan menyimpan perubahan tersebut. Jika ”Hapus” berarti sistem menghapus data dari database Loker tersimpan.

Sumber: Analisis Kebutuhan

Kebutuhan fungsional agar sistem dapat menyediakan fasilitas bagi Member untuk mengirimkan informasi lowongan kerja, direpresentasikan oleh *usecase* Mengirimkan Informasi Lowongan Kerja. Skenario untuk *usecase* Mengirimkan Informasi Lowongan Kerja ditunjukkan pada Tabel 4.13.

Tabel 4.13 Skenario *Usecase* Mengirimkan Informasi Lowongan Kerja

Usecase	Mengirimkan Informasi Lowongan Kerja
Aktor	Administrator
Tujuan	Mengirimkan informasi informasi lowongan kerja kepada member Via SMS
Deskripsi	Informasi lowongan kerja yang telah masuk database dikirimkan ke member sesuai kualifikasi akademik member.
Pra-kondisi	Halaman administrator dalam keadaan aktif.
Pos-kondisi	Informasi berhasil dikirim ke member
Aliran Utama	
Aksi dari Aktor	**Tanggapan dari Sistem**

1. Pada halaman administrator, aktor masuk ke halaman kirim informasi lowongan kerja dengan menekan tombol ”Kirim SMS”	2. Menampilkan halaman dengan pilihan pencarian berdasakan kualifikasi akademik dan jenis loker.
3. Aktor memilih kualifikasi akademik dan jenis loker kemudian menekan tombol ”cari”	4. Menampilkan semua informasi lowongan kerja yang telah dikelompokkan berdasarkan kualifikasi akademik dan jenis loker.
Aliran Alternatif 1 : Aktor mengirim informasi lowongan kerja per member	
1. Aktor bisa memilih salah satu informasi lowongan kerja kemudian mengirimkannya per member dengan menekan pilihan ”kirim per member”	2. Menampilkan daftar member yang memiliki kualifikasi akademik yang sesuai dengan kualifikasi akademik informasi lowongan kerja
3. Aktor bisa memilih member yang akan dikirimi informasi lowongan kerja dengan mengklik baris member yang diinginkan	4. Menampilkan halaman informasi hasil pengiriman loker Via SMS berserta daftar user dan id informasi lowongan kerja.
Aliran Alternatif 2 : Aktor mengirim informasi lowongan kerja ke semua member	
1. Aktor bisa memilih salah satu informasi lowongan kerja kemudian mengirimkannya ke semua member yang sesuai dengan kualifikasi akademik lowongan kerja dengan menekan pilihan ”kirim semua”	2. Menampilkan halaman informasi hasil pengiriman loker Via SMS berserta daftar user dan id informasi lowongan kerja.

Sumber: Analisis Kebutuhan

Kebutuhan fungsional selanjutnya yang harus disediakan oleh sistem adalah kebutuhan untuk memasukkan artikel ke database baik berupa kabar maupun tips dan menampilkannya. Kebutuhan tersebut direpresentasikan oleh *usecase* Mengatur Data Artikel. Skenario *usecase* Mengatur Data Artikel diberikan pada Tabel 4.14.

Tabel 4.14 Skenario *Usecase* Mengatur Data Artikel

Usecase	Manajemen Artikel
Aktor	Administrator
Tujuan	Menghapus *session* Member atau Administrator dan keluar dari halaman Member atau Administrator.
Deskripsi	Untuk melakukan proses *logout*, terlebih dahulu aktor memilih pilihan “*logout*” kemudian proses tersebut dilakukan.
Pra-kondisi	Halaman Administrator dalam keadaan aktif.
Pos-kondisi	Administrator berhasil memasukkan artikel ke database dan menampilkannya kepada Member maupun Guest
Aliran Utama	
Aksi dari Aktor	**Tanggapan dari Sistem**
1. Pada halaman Administrator, aktor masuk ke halaman manajemen artikel dengan memilih menu ”Artikel”.	2. Menampilkan halaman yang berisi tiga buah kategori, yaitu : • ”Input Artikel” merupakan halaman untuk memasukkan artikel baik yang berupa tips maupun kabar ke dalam database • ”Lihat kabar” merupakan halaman untuk menampilkan kabar terbaru yang telah dimasukkan ke database oleh Administrator • ”Lihat Tips” merupakan halaman untuk menampilkan tips-tips seputar pekerjaan yang telah dimasukkan ke database oleh

	Administrator
Aliran Alternatif 1 : Aktor Memasukkan Artikel	
1. Aktor memilih menu ”Input Artikel”	2. Menampilkan halaman yang terdiri dari beberapa *field* . *Field* isian diantaranya adalah kategori, judul dan isi, serta terdapat tombol ”simpan”
3. Aktor mengisi semua field kemudian menekan tombol ”Simpan”	4. Melakukan proses penyimpanan ke database
Aliran Alternatif 2: Eksepsi jika terdapat *field* yang kosong	
1. Aktor menekan tombol ”Simpan” tanpa mengisi *field* (langkah ke-3 pada aliran alternatif 1).	2. Menampilkan pernyataan peringatan kesalahan bahwa terdapat kesalahan berupa isian *field* kosong.
Aliran Alternatif 3: Aktor melihat artikel kabar	
1. Aktor bisa meihat hasil artikel dengan kategori kabar yang telah dimasukkan ke database dengan memilih menu ”Lihat kabar”	2. Menampilkan halaman yang berisi kabar terbaru yang telah dimasukkan oleh Administrator.
3. Aktor bisa memilih salah satu kabar dan melihat detailnya dengan cara mengklik judul kabar atau memilih menu ”selengkapnya”	4. Menampilkan halaman artikel dengan kategori kabar secara detail.
Aliran Alternatif 4: Aktor melihat artikel tips	
1. Aktor bisa meihat hasil artikel dengan kategori tips yang telah dimasukkan ke database dengan memilih menu ”Lihat kabar”	2. Menampilkan halaman yang berisi tips-tips seputar pekerjaan yang telah dimasukkan oleh Administrator.
3. Aktor bisa memilih salah satu kabar dan melihat detailnya dengan cara mengklik judul kabar atau memilih menu ”selengkapnya”	4. Menampilkan halaman artikel dengan kategori kabar secara detail.

Sumber: Analisis Kebutuhan

Kebutuhan fungsional selanjutnya yang harus disediakan oleh sistem adalah kebutuhan untuk mencetak laporan berdasarkan periode tertentu. Kebutuhan tersebut direpresentasikan oleh *usecase* Mencetak Laporan. Skenario *usecase* Mencetak Laporan diberikan pada Tabel 4.15.

Tabel 4.15 Skenario *Usecase* Mencetak laporan

Usecase	Mencetak Laporan
Aktor	Administrator
Tujuan	Mencetak laporan baik berupa data member, data informasi lowongan kerja maupun keuangan berdasarkan periode tertentu
Deskripsi	Untuk melakukan proses *pencetakan*, terlebih dahulu aktor memilih “jenis report” dan “periode bulan” kemudian akan ditampilkan halaman laporan sesuai permintaan yang bisa dicetak oleh Administrator
Pra-kondisi	Halaman Administrator dalam keadaan aktif.
Pos-kondisi	Administrator berhasil mencetak laporan berdasarkan periode tertentu
Aliran Utama	
Aksi dari Aktor	**Tanggapan dari Sistem**
1. Pada halaman Administrator, aktor memilih untuk mencetak laporan dengan memilih menu ”Report”	2. Menampilkan halaman dengan pilihan pencetakan berdasakan jenis *report* dan

	periode tertentu.
3. Aktor memilih jenis *report* dan periode kemudian menekan tombol ”cetak”	4. Menampilkan data sesuai jenis *report* sesuai dengan periode yang dipilih

Sumber: Analisis Kebutuhan

Kebutuhan fungsional selanjutnya yang harus disediakan oleh sistem adalah kebutuhan untuk menghapus *session* Member atau Administrator. Kebutuhan tersebut direpresentasikan oleh *usecase Logout*. Skenario *usecase Logout* diberikan pada Tabel 4.16.

Tabel 4.16 Skenario *Usecase Logout*

Usecase	*Logout*
Aktor	Member, Administrator
Tujuan	Menghapus *session* Member atau Administrator dan keluar dari halaman Member atau Administrator.
Deskripsi	Untuk melakukan proses *logout*, terlebih dahulu aktor memilih pilihan “*logout*” kemudian proses tersebut dilakukan.
Pra-kondisi	Halaman Member atau Administrator dalam keadaan aktif.
Pos-kondisi	*Session* Member atau Administrator terhapus dan keluar dari halaman Member atau Administrator.
Aliran Utama	
Aksi dari Aktor	**Tanggapan dari Sistem**
5. Pada halaman Member/Administrator, aktor memilih untuk *logout* dengan cara menekan pilihan ”*logout*”.	6. Menghapus *session* Member/ Administrator dan menampilkan halaman baru yeng berisi pesan bahwa aktor telah *logout*.

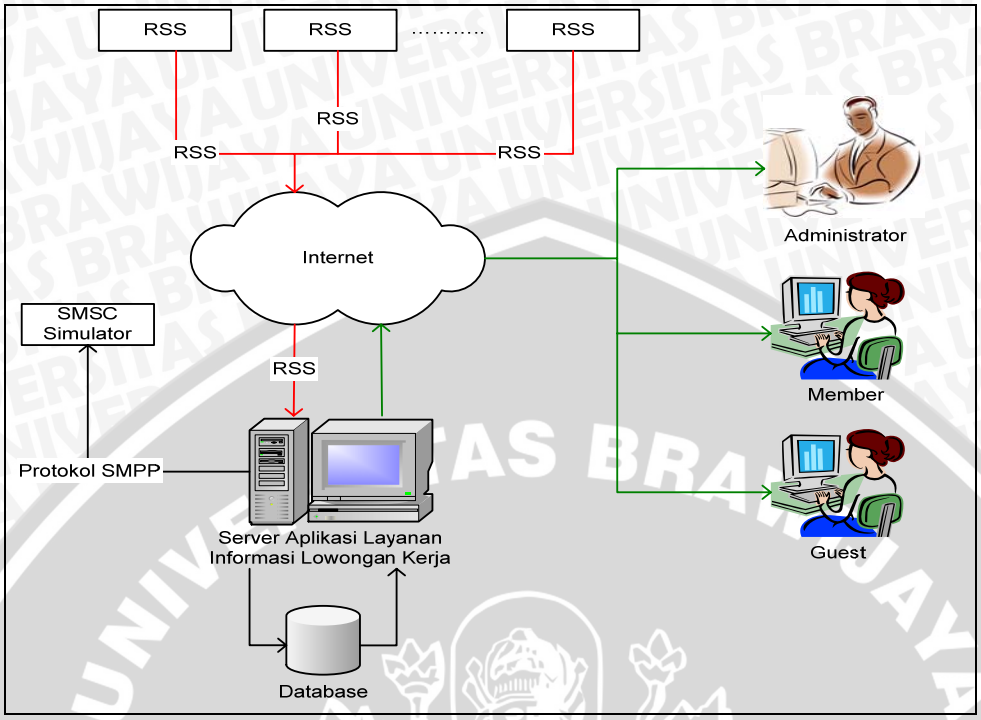
Sumber: Analisis Kebutuhan

4.2. Perancangan

Perancangan perangkat lunak ditujukan untuk memberikan gambaran secara umum mengenai perangkat lunak yang akan dibuat. Pada proses perancangan ini, dilakukan perancangan blok diagram sistem, perancangan detail, perancangan basis data dan perancangan antarmuka.

4.2.1. Perancangan Sistem

Pada blok diagram sistem (Gambar 4.2) digambarkan secara umum tentang sistem aplikasi layanan informasi lowongan kerja. Sistem ini berjalan di *server-side*, *client* mengirim *request* kepada *server* kemudian *server* memberikan *response* kepada *client* sesuai dengan *request* yang dikirimkan.



Gambar 4.2 Blok diagram sistem
Sumber: Perancangan

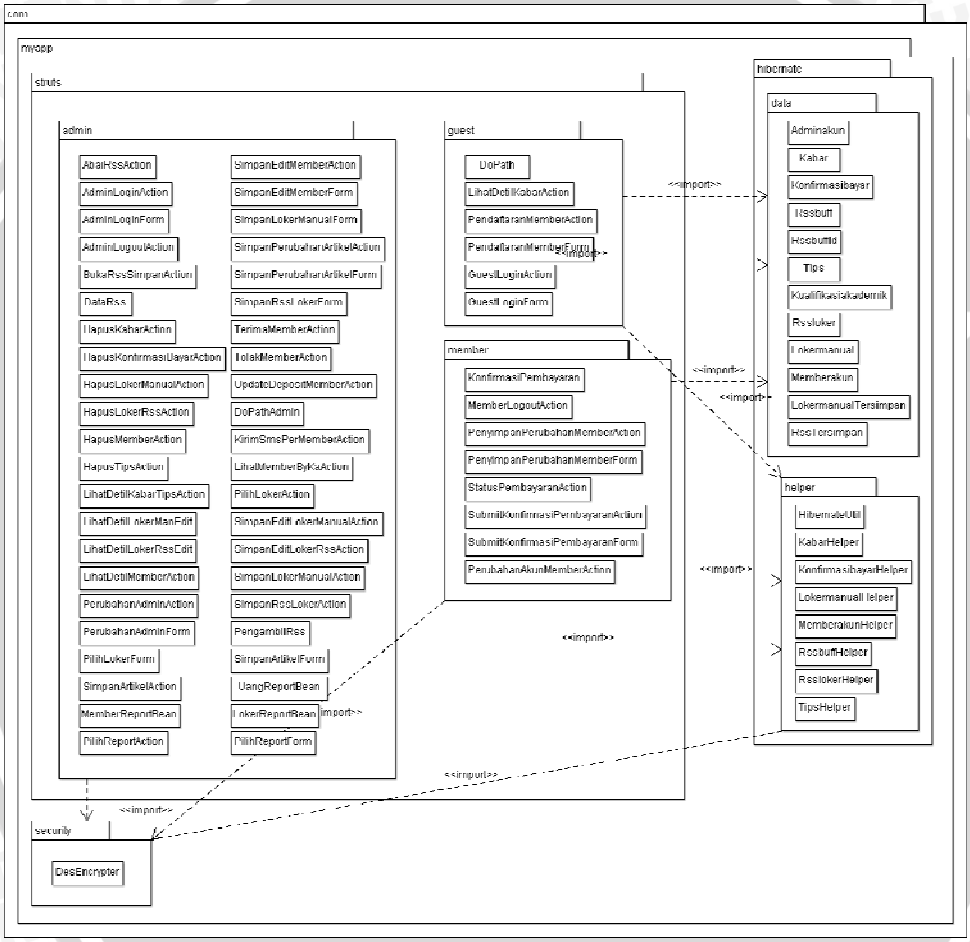
Sistem mengumpulkan RSS lowongan kerja dari berbagai website penyedia info lowongan kerja. Selain didapatkan dari pengumpulan RSS, info lowongan kerja yang dihimpun dalam sistem ini juga berasal dari masukan Administrator. Info lowongan kerja yang telah dihimpun dalam sistem ini, kemudian bisa dilihat baik oleh Member maupun Guest melalui media website. Selain itu, info lowongan ini juga akan dikirimkan ke Member melalui media SMS. Proses pengiriman SMS dilakukan dengan mengirimkan info lowongan kerja sebagai pesan singkat dari server aplikasi ke SMS Center melalui protokol SMPP. Info lowongan kerja yang dihimpun dalam sistem ini disimpan pada suatu *relational database.*

4.2.2. Perancangan Detail

Perancangan lebih detail dilakukan dengan menganalisis klas yang akan digunakan untuk membangun sistem dan interaksi dinamis antar klas sehingga mampu mengimplementasikan fungsionalitas sistem yang telah dimodelkan dalam diagram *usecase*. Pemodelan yang dilakukan pada proses perancangan detail ini adalah pemodelan Diagram Klas dan Diagram Sekuensial.

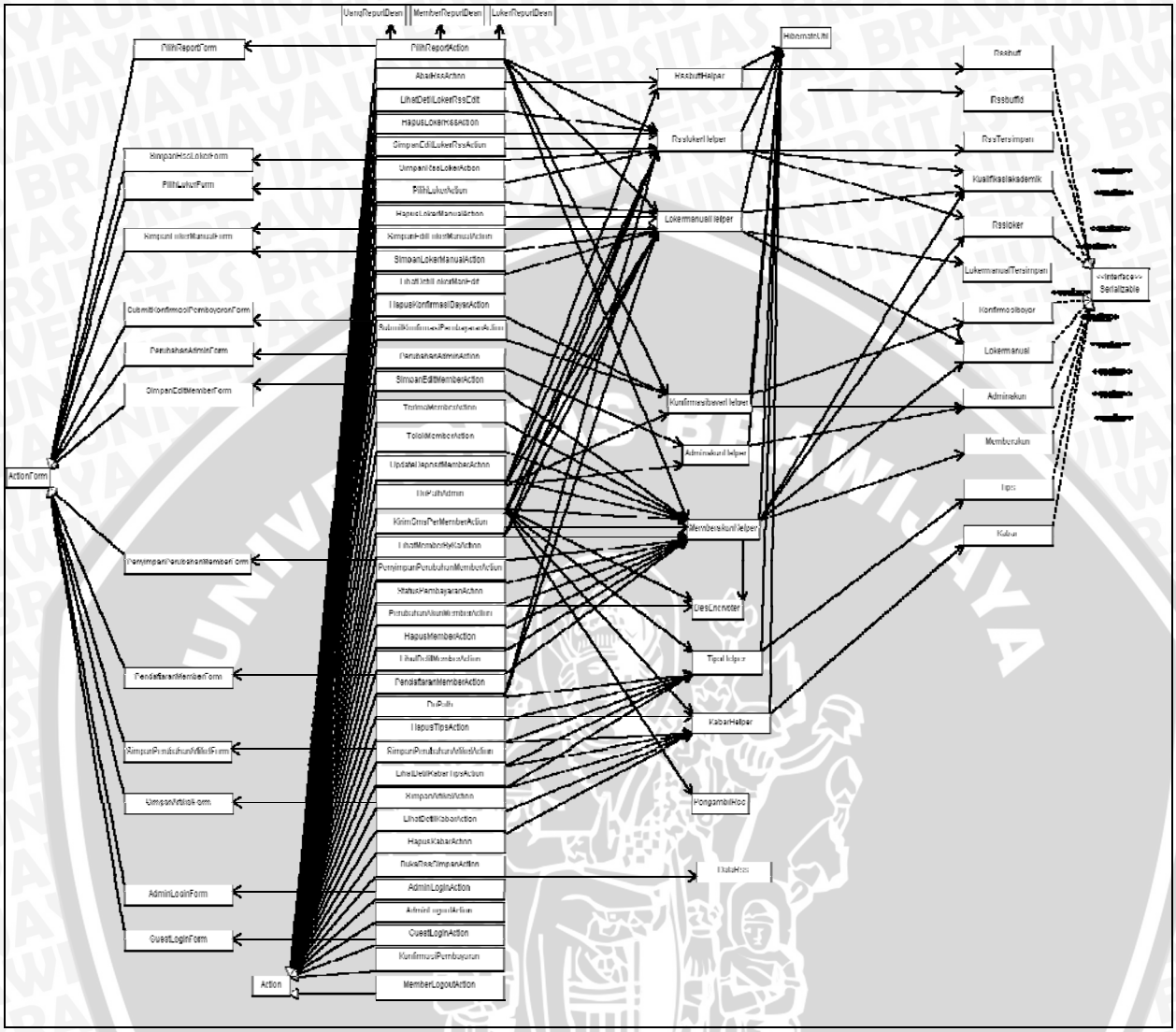
4.2.2.1. Diagram Klas

Klas-klas yang digunakan unuk membangun sistem ini dikelompokkan ke dalam enam paket, yaitu paket **`com.myapp.struts.admin`**, **`com.myapp.struts.guest`**, **`com.myapp.struts.member`**, **`com.myapp.hibernate.data`**, **`com.myapp.hibernate.helper`**, **`com.myapp.security`**. Pengelompokan klas-klas ke dalam paket tersebut ditunjukkan dalam diagram paket pada Gambar 4.3.



Gambar 4.3 Diagram Paket
Sumber: Perancangan

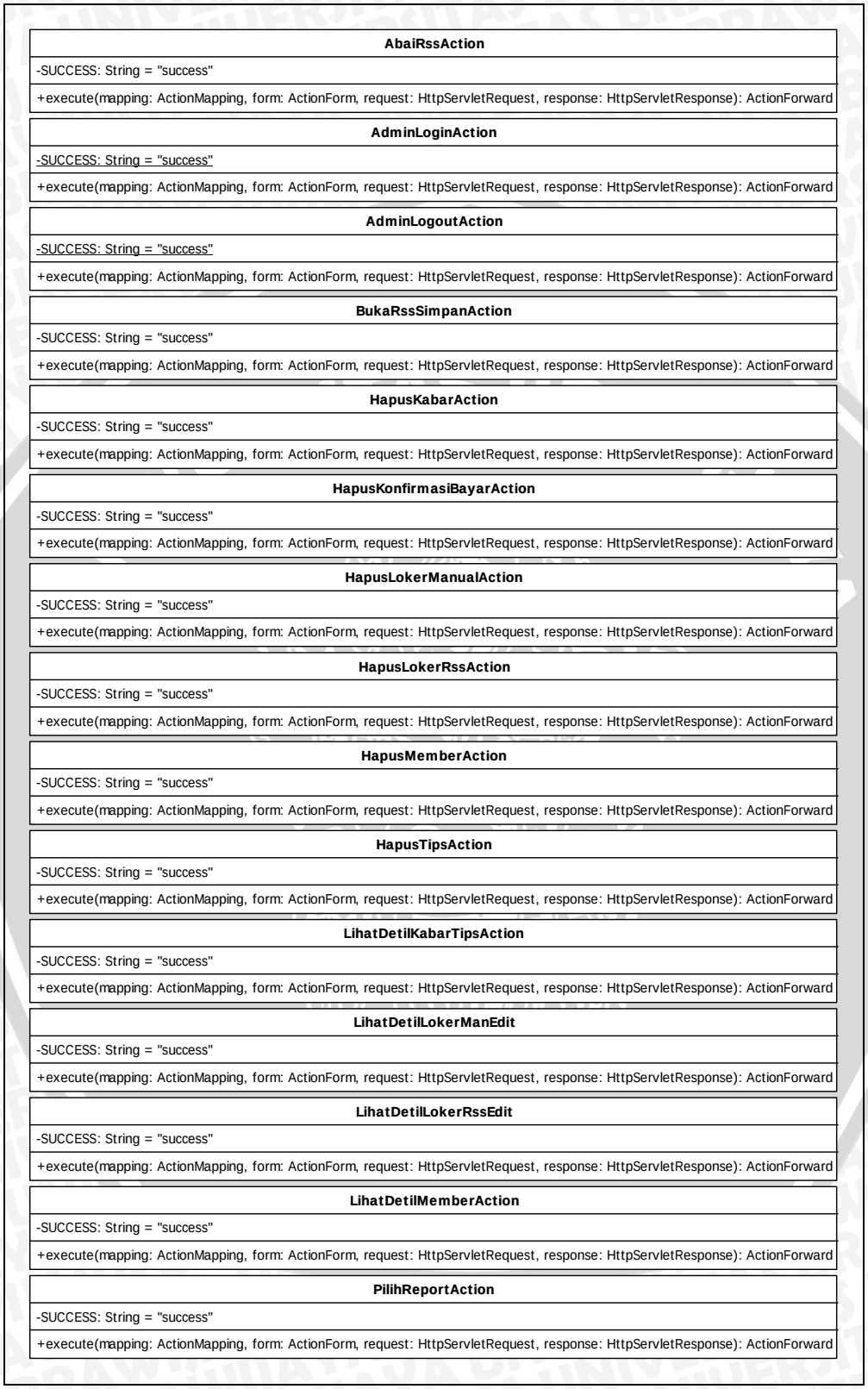
Klas-klas yang yang ada dalam paket-paket saling berelasi untuk membentuk suatu fungsionalitas tertentu pada sistem. Relasi antar klas yang digunakan untuk aplikasi ini dapat dilihat pada Gambar 4.4.



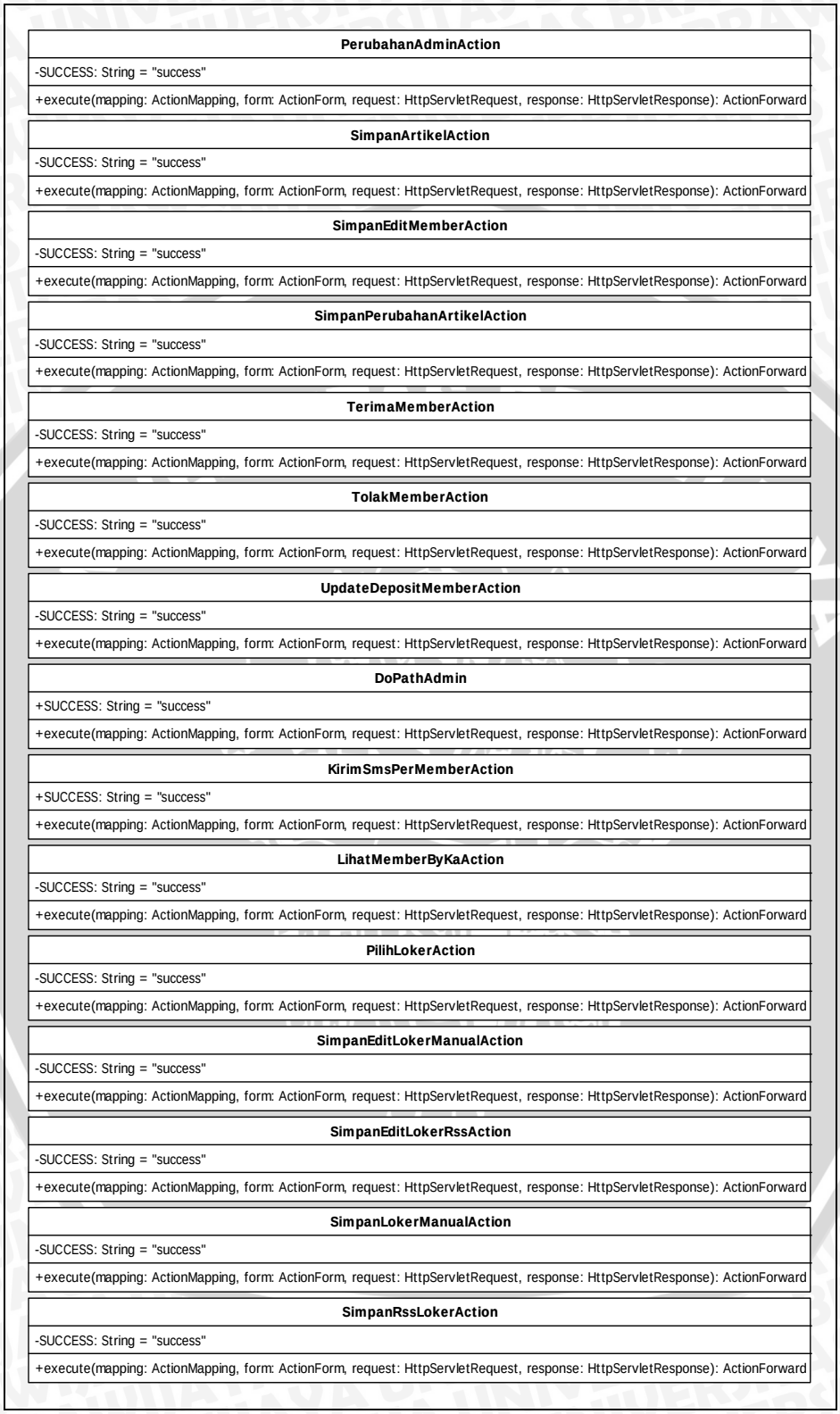
Gambar 4.4 Relasi antar Klas
Sumber: Perancangan

4.2.2.1.1. Diagram Klas pada Paket `com.myapp.struts.admin`

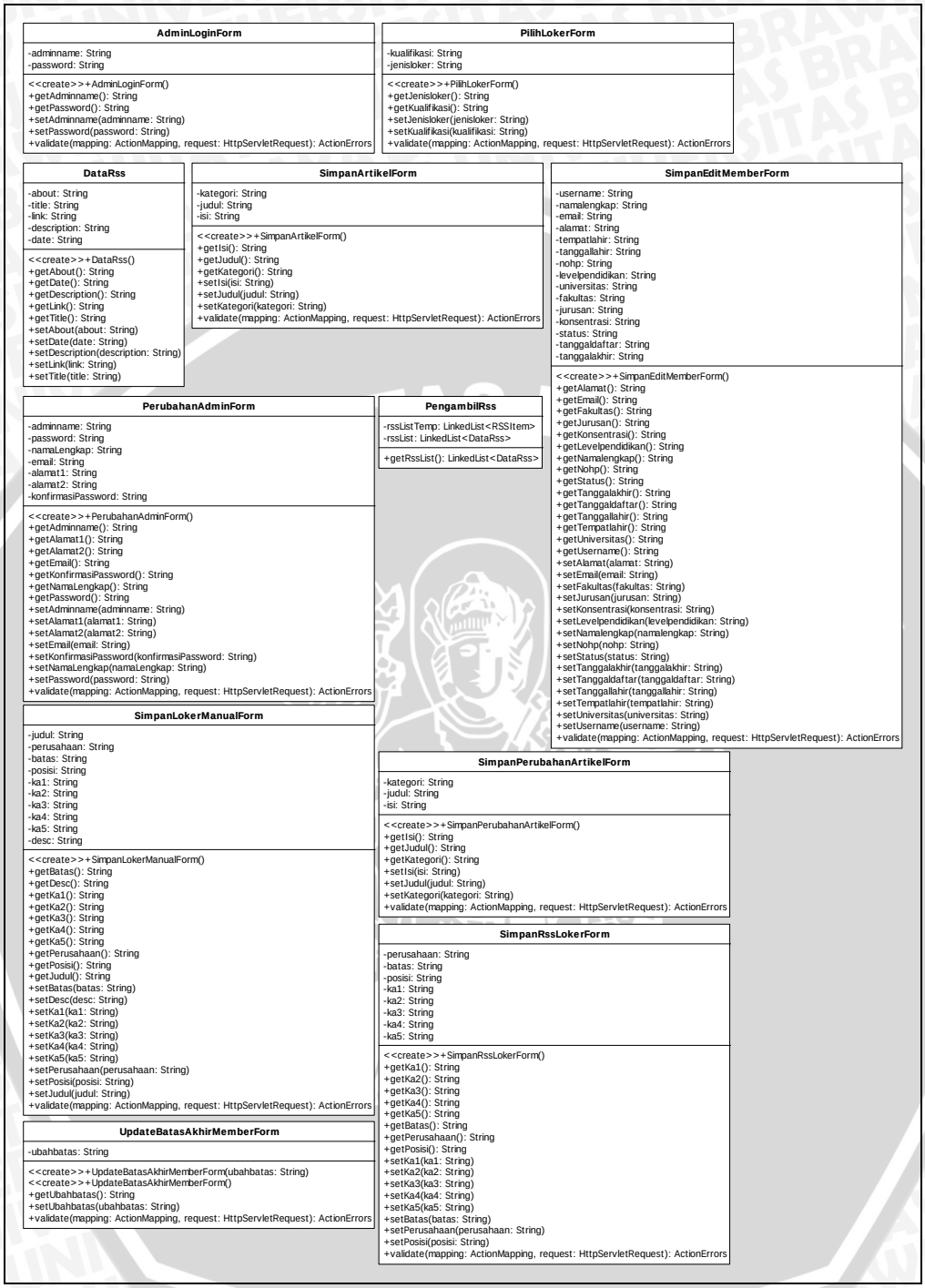
Pada paket **`com.myapp.struts.admin`** terdapat beberapa klas. Atribut dan *method* yang dimiliki oleh masing-masing klas dalam paket **`com.myapp.struts.admin`** digambarkan dalam diagram klas pada Gambar 4.5, 4.6, 4.7 dan 4.8.



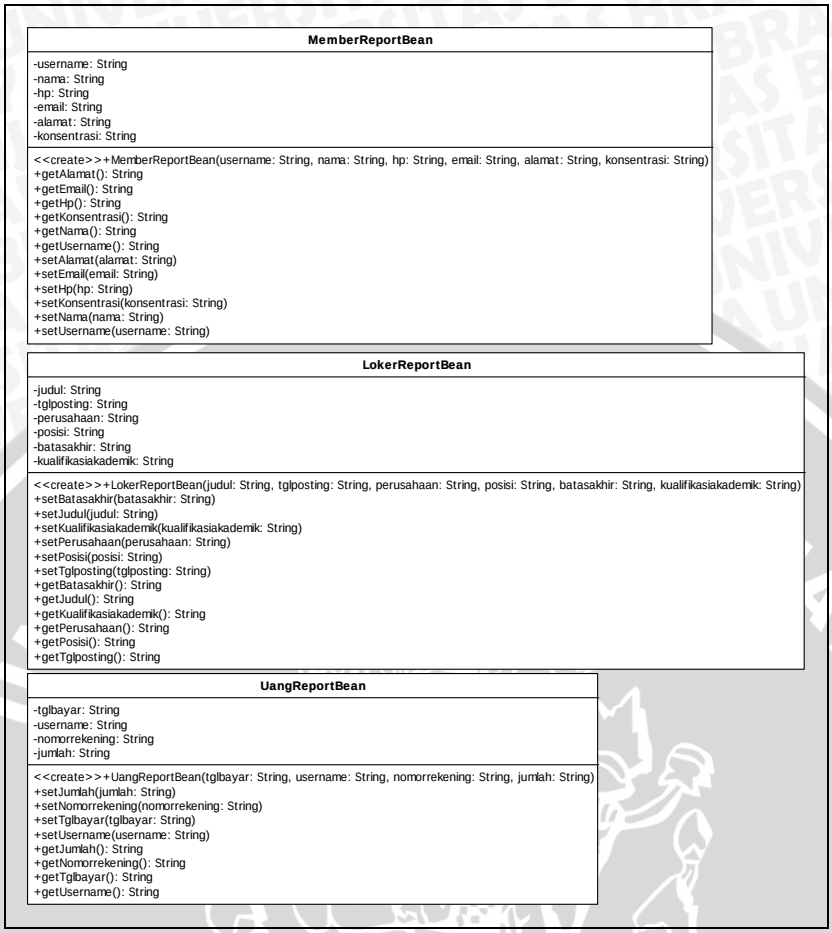
Gambar 4.5 Diagram klas action pada paket **`com.myapp.struts.admin`**
Sumber: Perancangan



Gambar 4.6 Diagram klas action pada paket **`com.myapp.struts.admin`**
Sumber: Perancangan



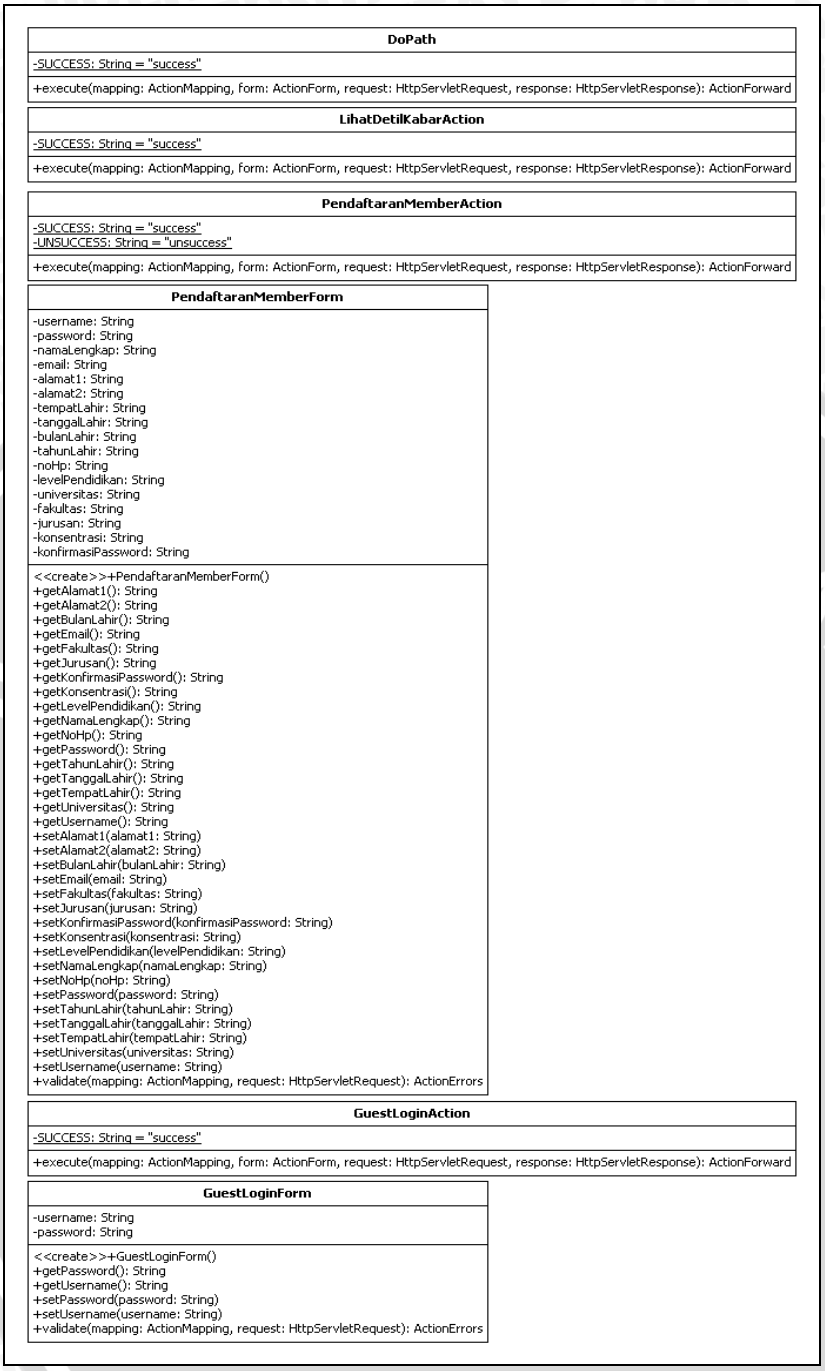
Gambar 4.7 Diagram klas form pada paket **`com.myapp.struts.admin`**
Sumber: Perancangan



Gambar 4.8 Diagram klas *report bean* pada paket **`com.myapp.struts.admin`**
Sumber: Perancangan

4.2.2.1.2. Diagram Klas pada Paket `com.myapp.struts.guest`

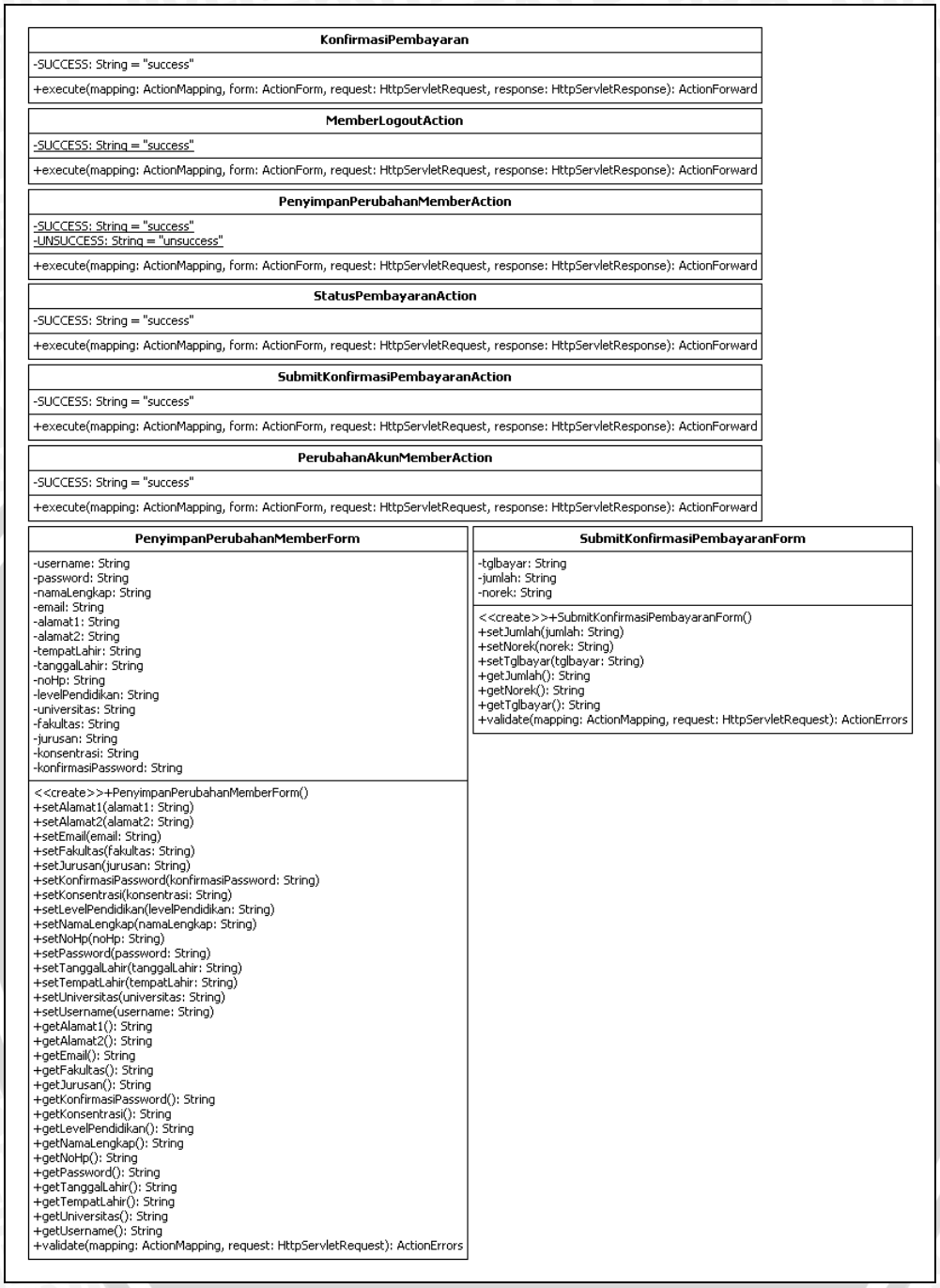
Diagram klas untuk klas-klas yang ada dalam paket **`com.myapp.struts.guest`** ditunjukkan pada Gambar 4.9.



Gambar 4.9 Diagram klas pada paket **`com.myapp.struts.guest`**
Sumber: Perancangan

4.2.2.1.3. Diagram Klas pada Paket `com.myapp.struts.member`

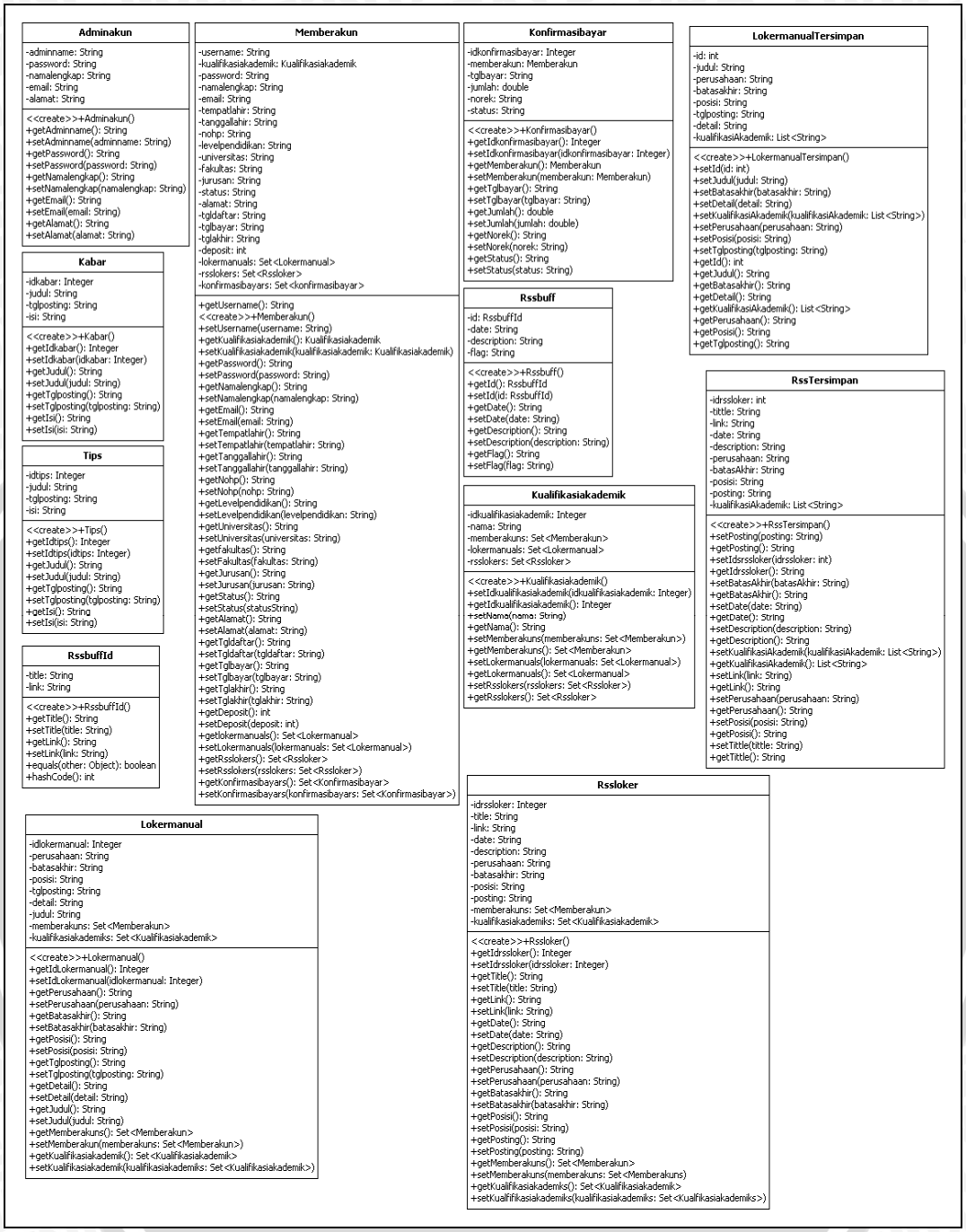
Diagram klas untuk klas-klas yang ada dalam paket **`com.myapp.struts.member`** ditunjukkan pada Gambar 4.10.



Gambar 4.10 Diagram klas pada paket **`com.myapp.struts.member`**
Sumber: Perancangan

4.2.2.1.4. Diagram Klas pada Paket `com.myapp.hibernate.data`

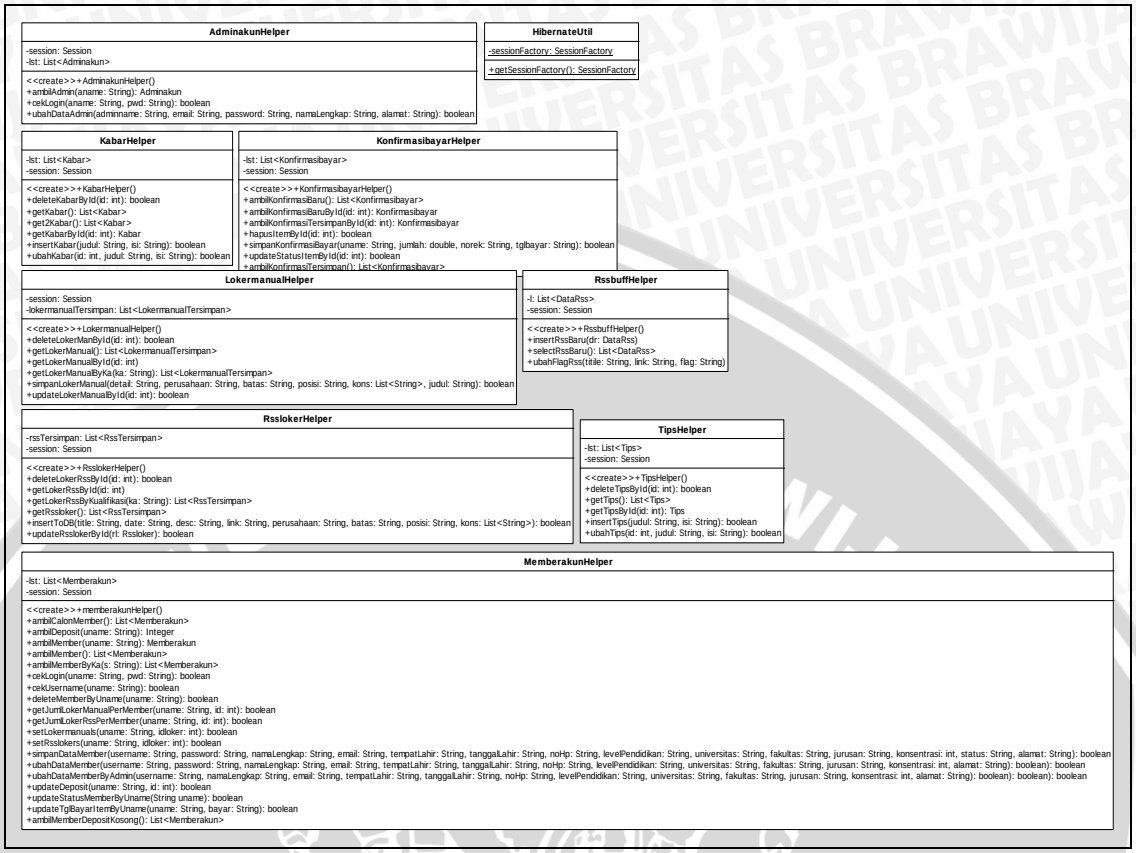
Diagram klas untuk klas-klas yang ada dalam paket **`com.myapp.hibernate.data`** ditunjukkan pada Gambar 4.11



Gambar 4.11 Diagram klas pada paket **`com.myapp.hibernate.data`**
Sumber: Perancangan

4.2.2.1.5. Diagram Klas pada Paket `com.myapp.hibernate.helper`

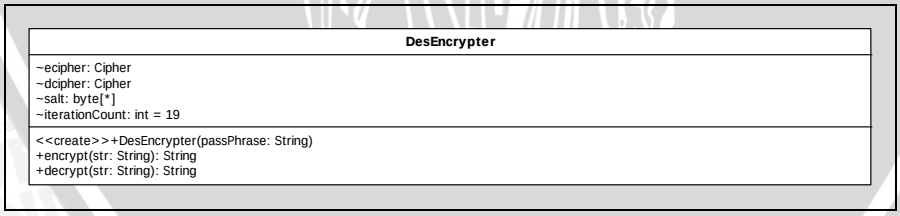
Klas-klas yang ada pada paket **`com.myapp.hibernate.helper`** (Gambar 4.12) merupakan klas implementasi *data access object (DAO)* yang digunakan untuk melakukan manipulasi terhadap data dalam database.



Gambar 4.12 Diagram klas pada paket **`com.myapp.hibernate.helper`**
Sumber: Perancangan

4.2.2.1.6. Diagram Klas pada Paket `com.myapp.security`

Pada paket **`com.myapp.security`** hanya terdapat satu buah *class* yaitu *class* DesEncrypter. Atribut dan operasi yang dimiliki oleh masing-masing *class* tersebut ditunjukkan pada Gambar 4.13



Gambar 4.13 Diagram klas pada paket **`com.myapp.security`**
Sumber: Perancangan

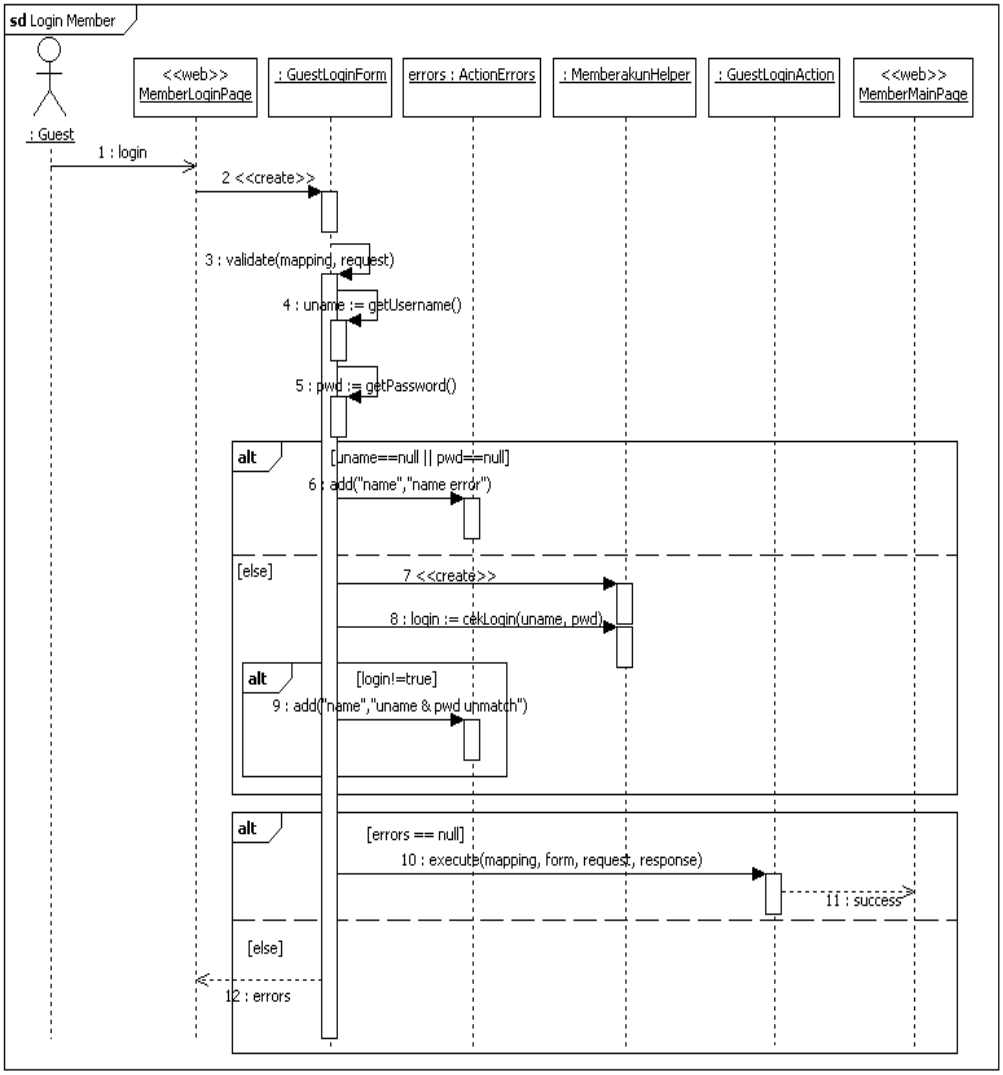
4.2.2.2. Diagram Sekuensial

Diagram sekuensial menunjukkan sebuah interaksi antara objek-objek dari suatu *class* yang disusun ke dalam urutan waktu. Diagram ini secara khusus mengambil acuan pada *usecase* untuk membentuk fungsionalitas sistem. Pada

bagian ini akan digambarkan diagram sekuensial dari masing-masing fungsionalitas sistem yang digambarkan pada *usecase*.

4.2.2.2.1. Diagram Sekuensial Login Member

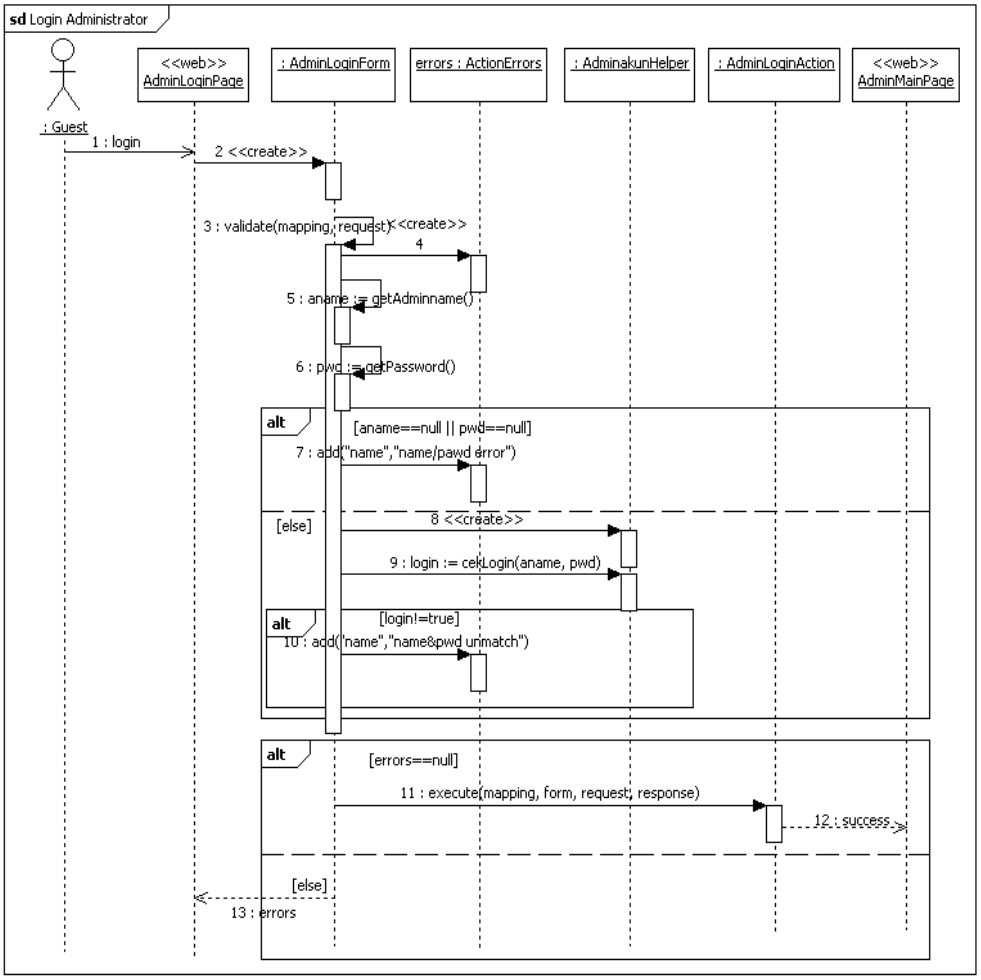
Diagram sekuensial Login Member (Gambar 4.14) menggambarkan interaksi yang terjadi ketika seorang pengguna melakukan *login* sebagai Member.



Gambar 4.14 Diagram sekuensial Login Member
Sumber: Perancangan

4.2.2.2.2. Diagram Sekuensial Login Administrator

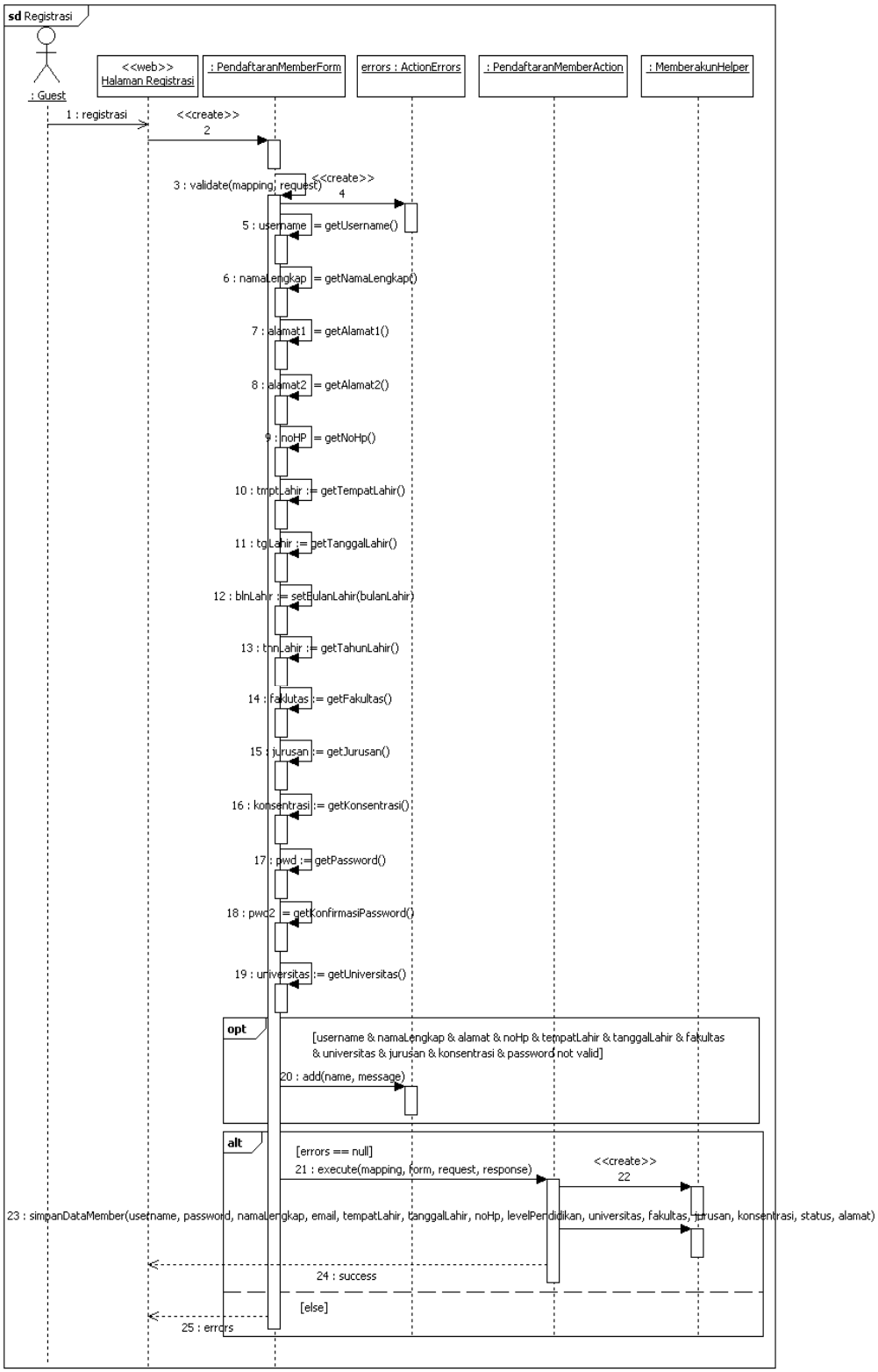
Diagram sekuensial Login Administrator (Gambar 4.15) menggambarkan interaksi yang terjadi ketika seorang pengguna melakukan *login* sebagai Administrator.



Gambar 4.15 Diagram sekuensial Login Administrator
Sumber: Perancangan

4.2.2.2.3. Diagram Sekuensial Registrasi

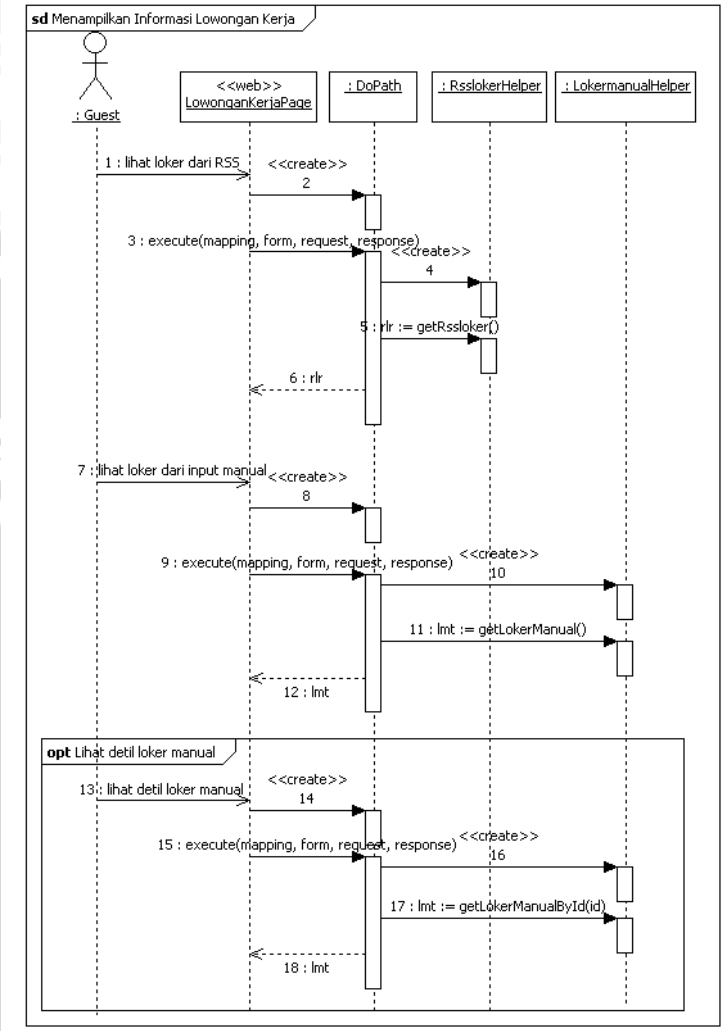
Diagram sekuensial Registrasi menggambarkan interaksi yang terjadi ketika seorang pengguna melakukan registrasi. Gambar 4.16 menunjukkan urutan proses untuk *usecase* Regitrasi.



Gambar 4.16 Diagram sekuensial Registrasi
Sumber: Perancangan

4.2.2.2.4. Diagram Sekuensial Menampilkan Informasi Lowongan Kerja

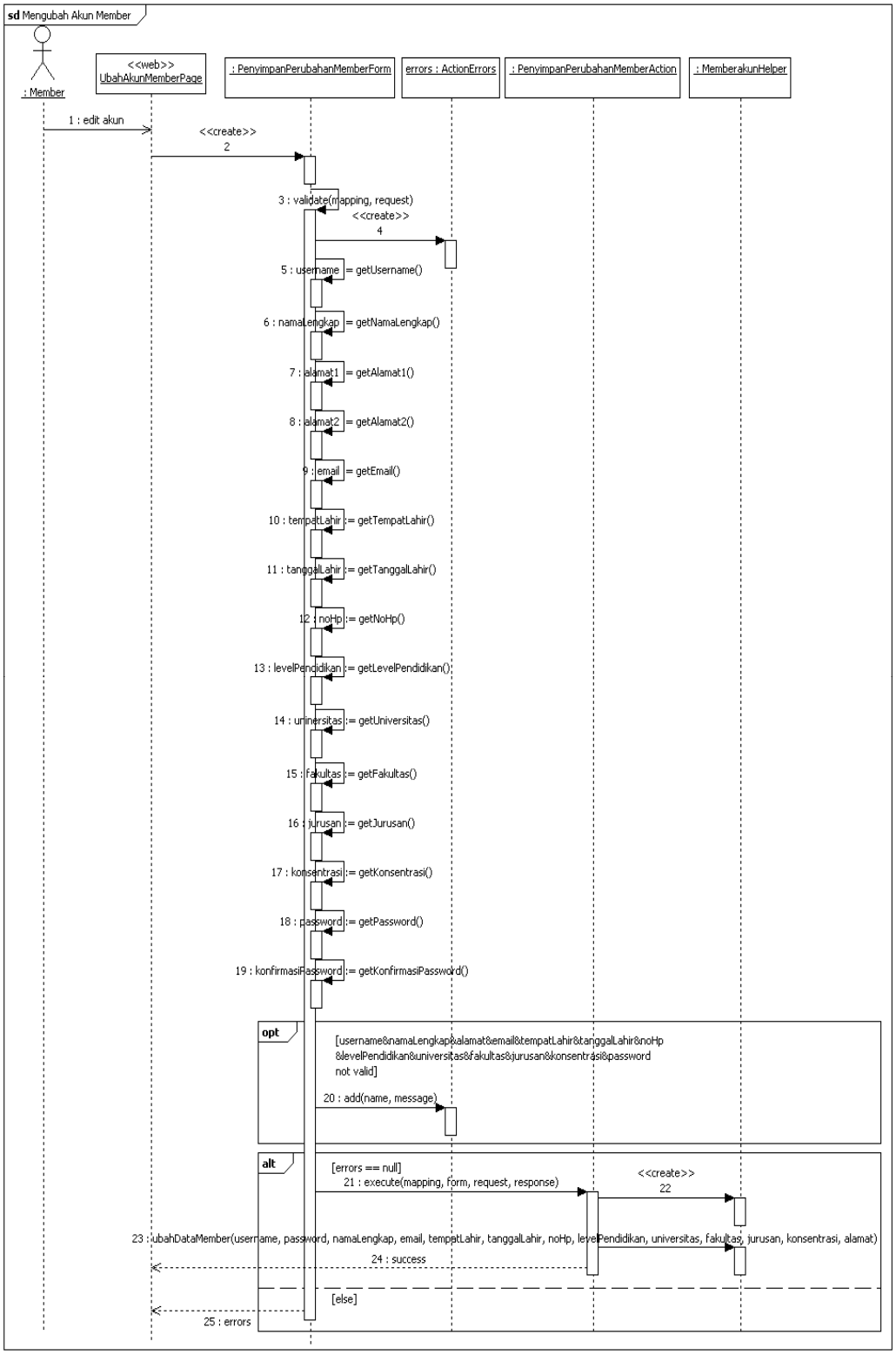
Diagram sekuensial ini (Gambar 4.17) menggambarkan interaksi yang terjadi ketika seorang pengguna menampilkan informasi lowongan kerja.



Gambar 4.17 Diagram sekuensial Menampilkan Informasi Lowongan Kerja
Sumber: Perancangan

4.2.2.2.5. Diagram Sekuensial Mengubah Profil Member

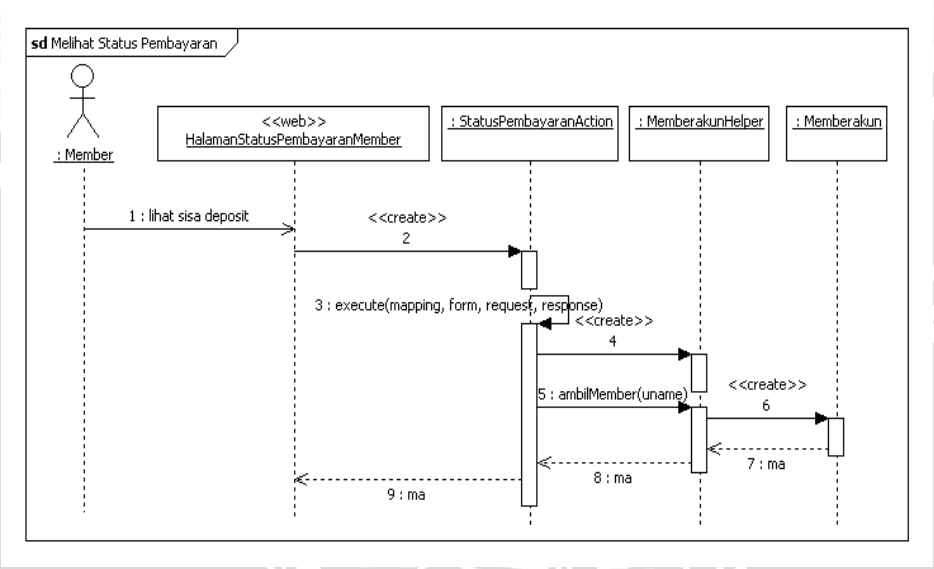
Diagram sekuensial ini (Gambar 4.18) menggambarkan interaksi yang terjadi ketika seorang Member mengubah akunnya.



Gambar 4.18 Diagram sekuensial Mengubah Profil Member
Sumber: Perancangan

4.2.2.2.6. Diagram Sekuensial Melihat Status Pembayaran

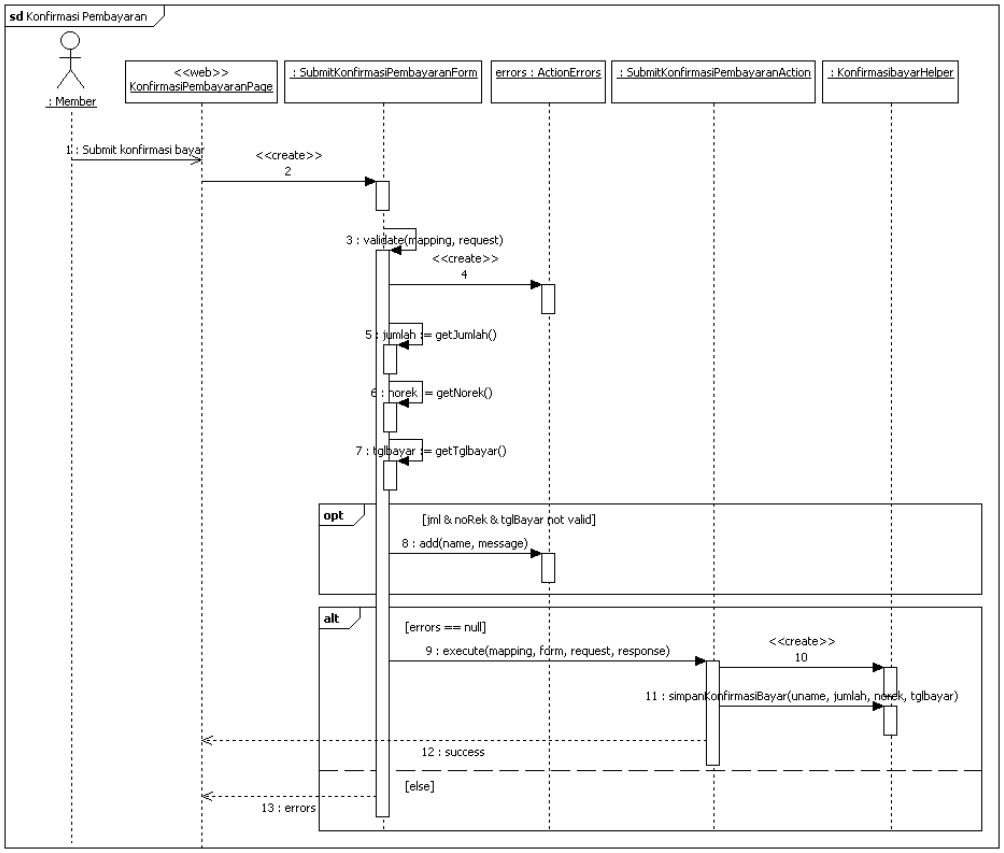
Sistem memberikan fasilitas bagi Member untuk melihat sisa deposit yang dimiliki. Fasilitas ini disediakan oleh *usecase* Melihat Status Pembayaran. Diagram sekuensial untuk *usecase* Melihat Status Pembayaran menggambarkan interaksi yang terjadi ketika seorang Member melihat status pembayaran. Gambar 4.19 merupakan gambar diagram sekuensial untuk *usecase* Melihat Status Pembayaran.



Gambar 4.19 Diagram sekuensial Melihat Status Pembayaran
Sumber: Perancangan

4.2.2.2.7. Diagram Sekuensial Konfirmasi Pembayaran

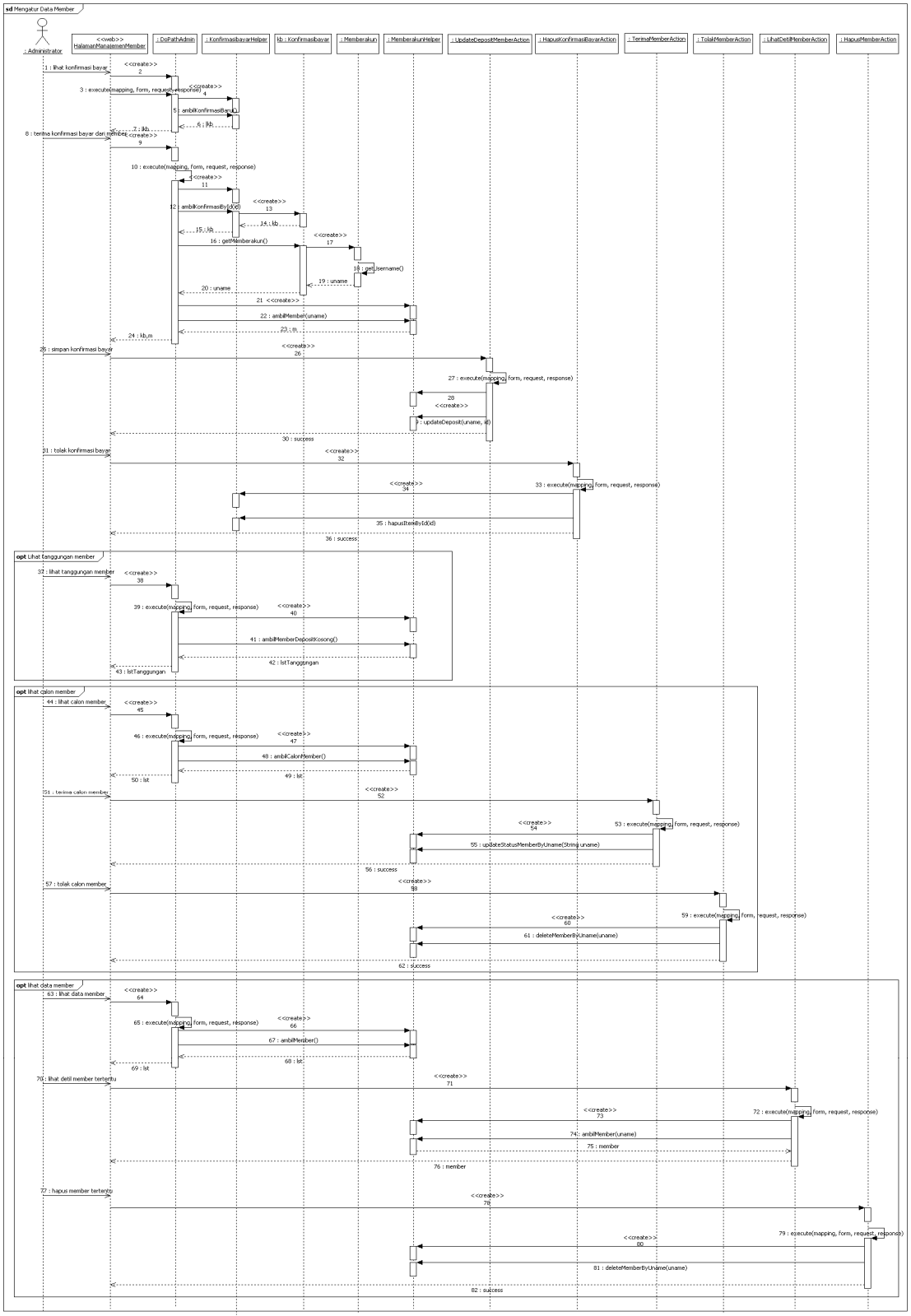
Sistem memberikan fasilitas bagi Member untuk memberikan konfirmasi atas pembayaran yang telah dilakukan kepada Administrator sistem. Fasilitas ini disediakan oleh *usecase* Konfirmasi Pembayaran. Diagram sekuensial Konfirmasi Pembayaran menggambarkan interaksi antar klas yang terjadi ketika seorang Member memberikan konfirmasi atas proses yang telah dilakukan. Urutan interaksi antar klas dalam merealisasikan *usecase* Konfirmasi Pembayaran ditunjukkan pada Gambar 4.20.



Gambar 4.20 Diagram sekuensial Konfirmasi Pembayaran
Sumber: Perancangan

4.2.2.2.8. Diagram Sekuensial Mengatur Data Member

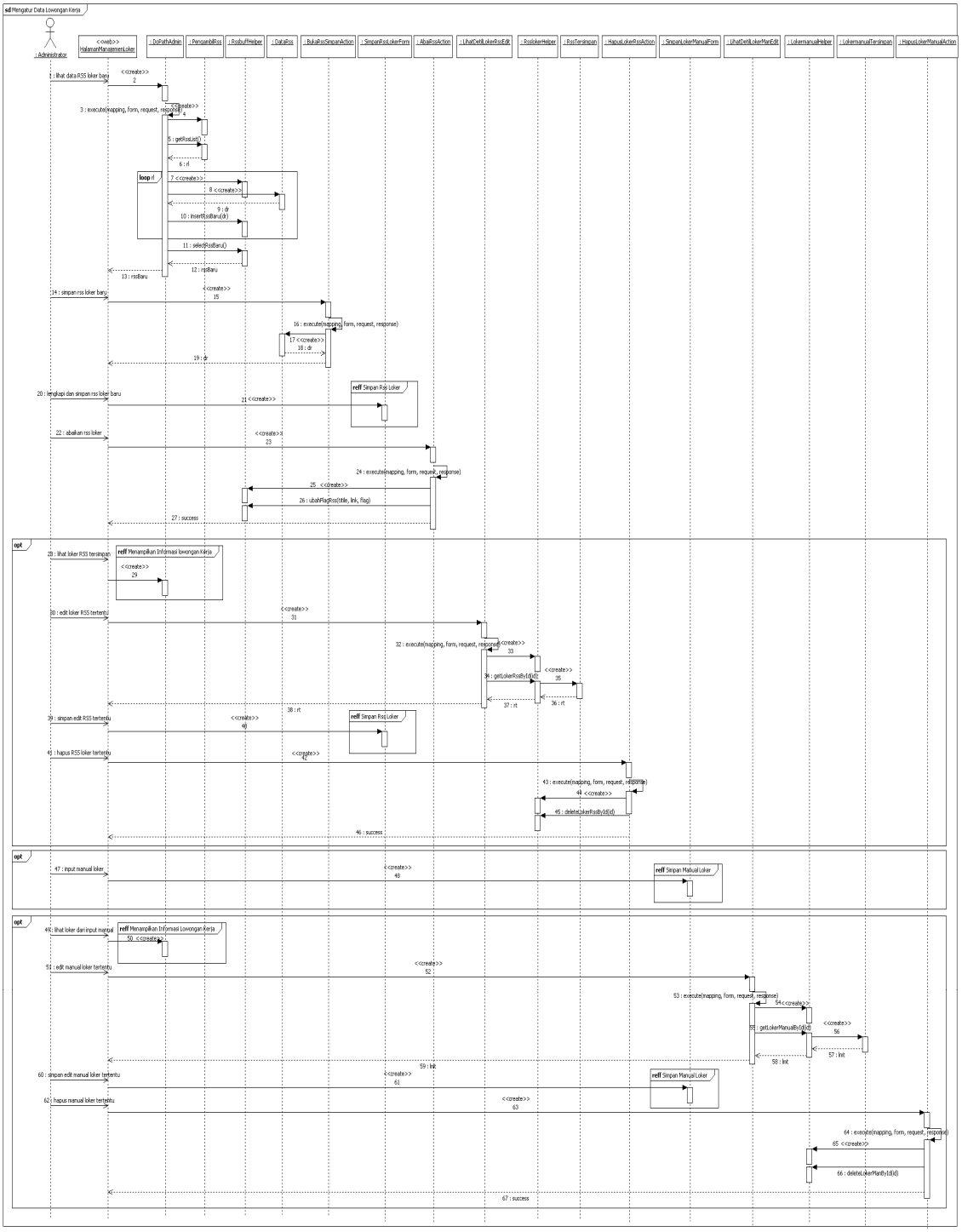
Diagram sekuensial ini (Gambar 4.21) menggambarkan interaksi yang terjadi ketika seorang Administrator mengatur data member. Termasuk di dalam proses mengatur data member diantaranya adalah melihat data konfirmasi pembayaran baru yang diberikan oleh member, melihat data tanggungan pembayaran member, melihat data calon member dan melihat data member.



Gambar 4.21 Diagram sekuensial Mengatur Data Member
Sumber: Perancangan

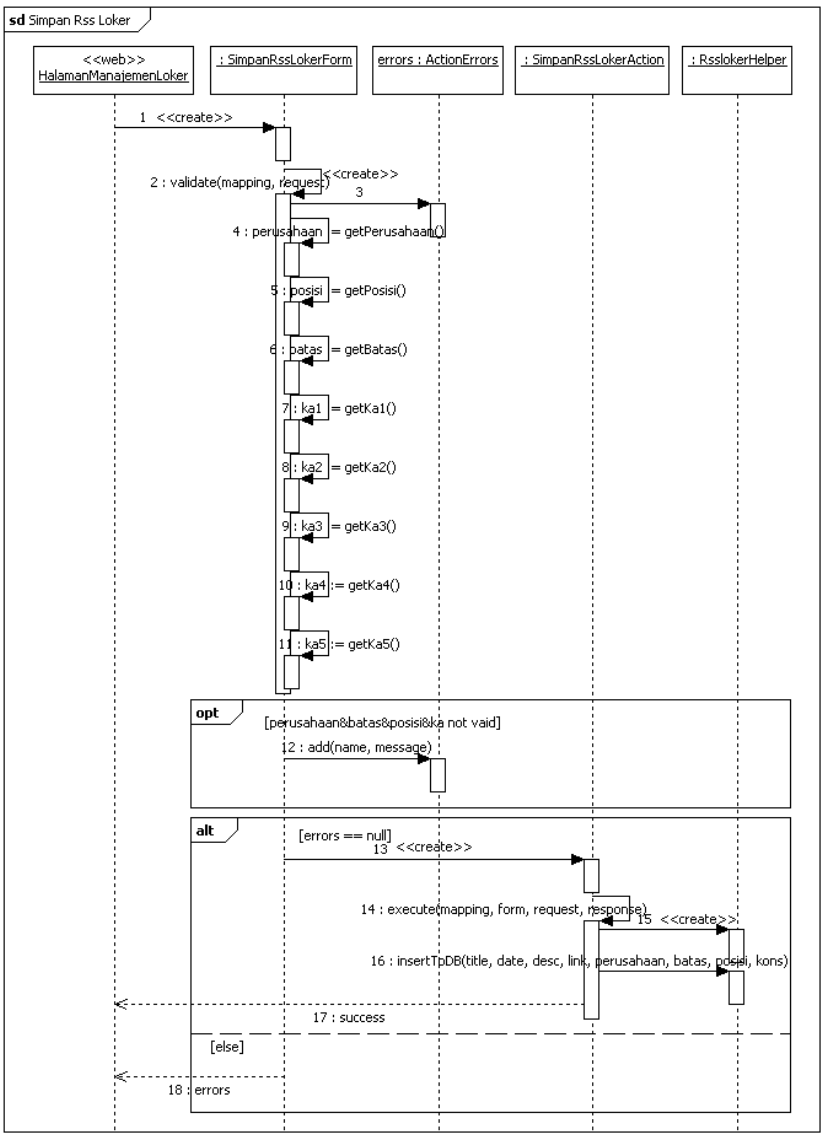
4.2.2.2.9. Diagram Sekuensial Mengatur Data Lowongan Kerja

Diagram sekuensial ini (Gambar 4.22) menggambarkan interaksi yang terjadi ketika seorang Administrator mengatur data lowongan kerja.

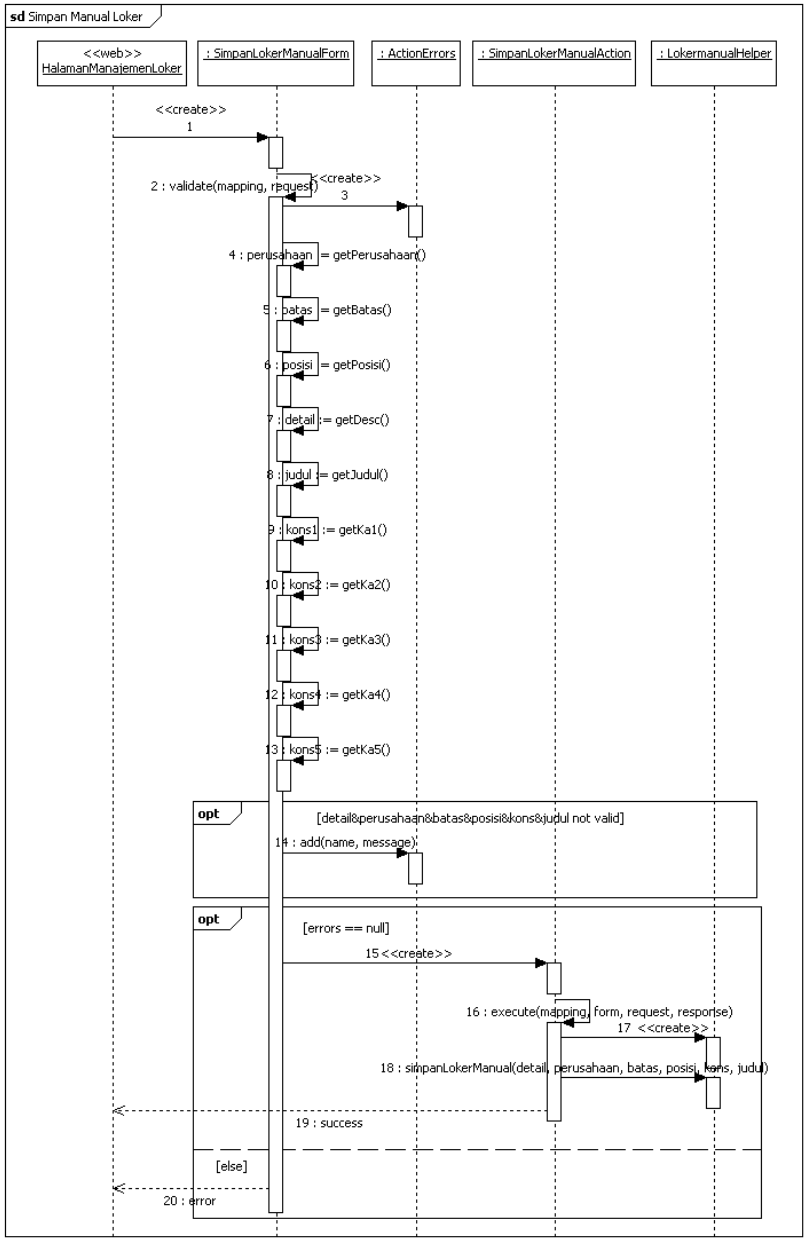


Gambar 4.22 Diagram sekuensial Mengatur Data Lowongan Kerja
Sumber: Perancangan

Diagram sekuensial pada Gambar 4.22 mereferensi pada diagram sekuensial pada Gambar 4.23 dan Gambar 4.24.



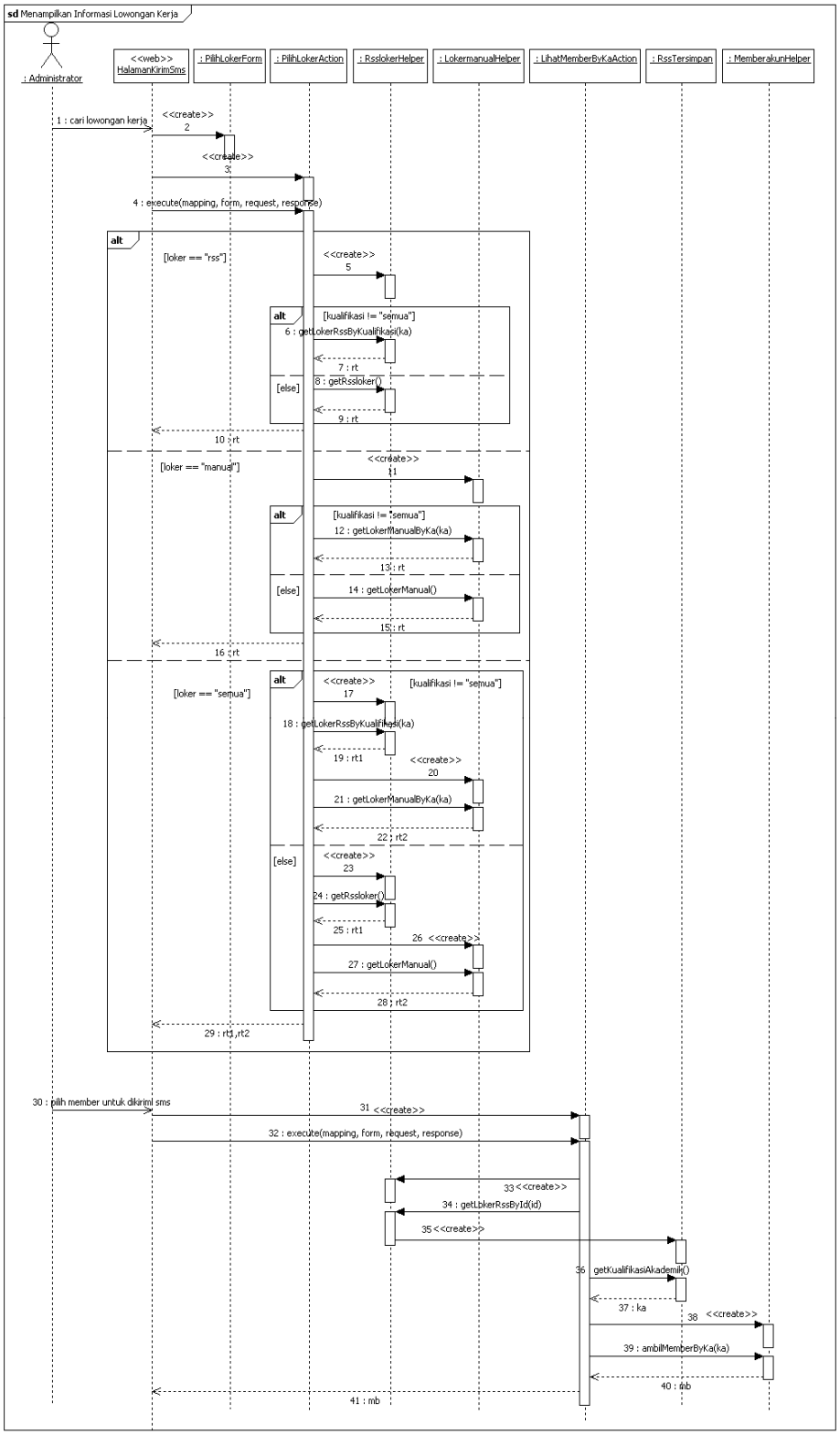
Gambar 4.23 Diagram sekuensial Simpan Rss Loker
Sumber: Perancangan



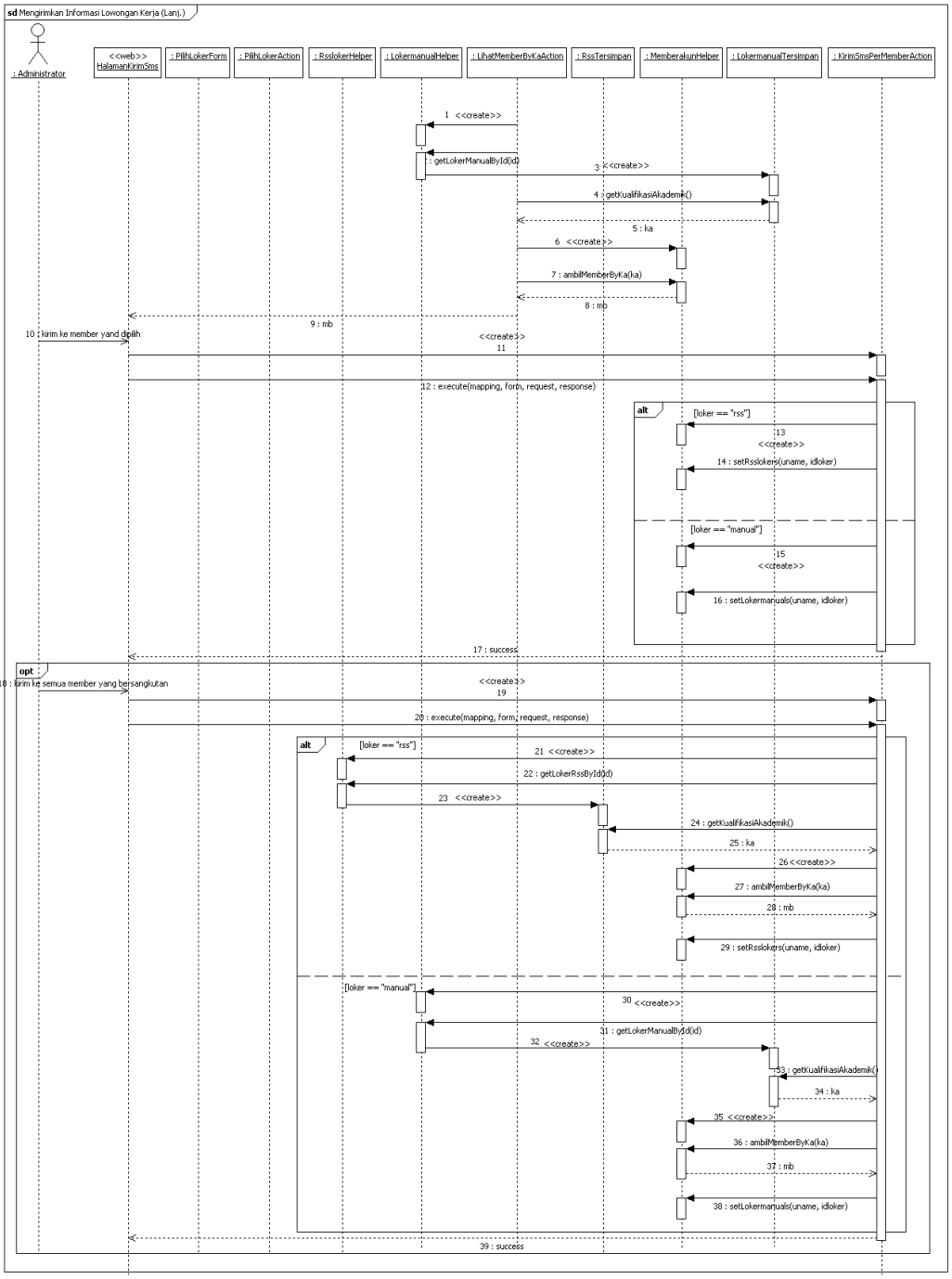
Gambar 4.24 Diagram sekuensial Simpan Manual Loker
Sumber: Perancangan

4.2.2.2.10. Diagram Sekuensial Mengirimkan Informasi Lowongan Kerja

Diagram sekuensial ini (Gambar 4.25, 4.26) menggambarkan interaksi yang terjadi ketika seorang Administrator mengirimkan informasi lowongan keja.



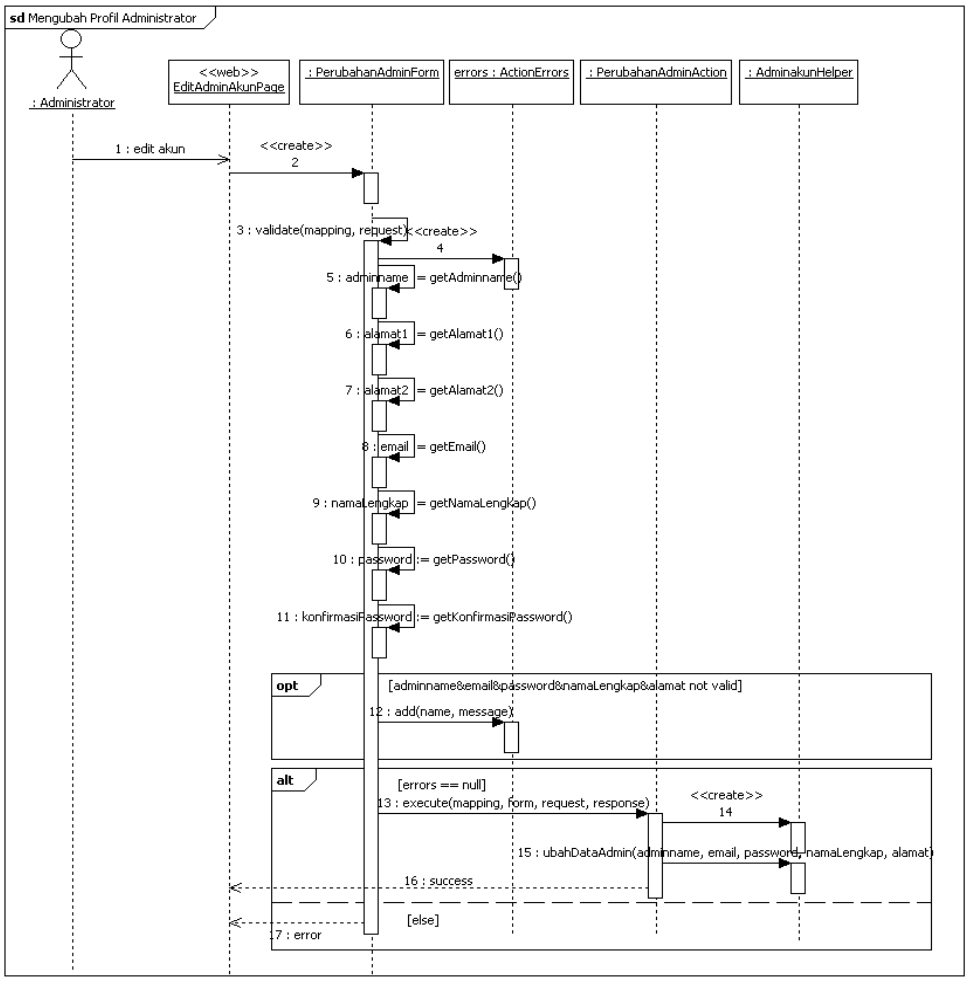
Gambar 4.25 Diagram sekuensial Mengirimkan informasi Lowongan Kerja
Sumber: Perancangan



Gambar 4.26 Diagram sekuensial Lanjutan Mengirimkan Informasi Lowongan Kerja
Sumber: Perancangan

4.2.2.2.11. Diagram Sekuensial Mengubah Profil Administrator

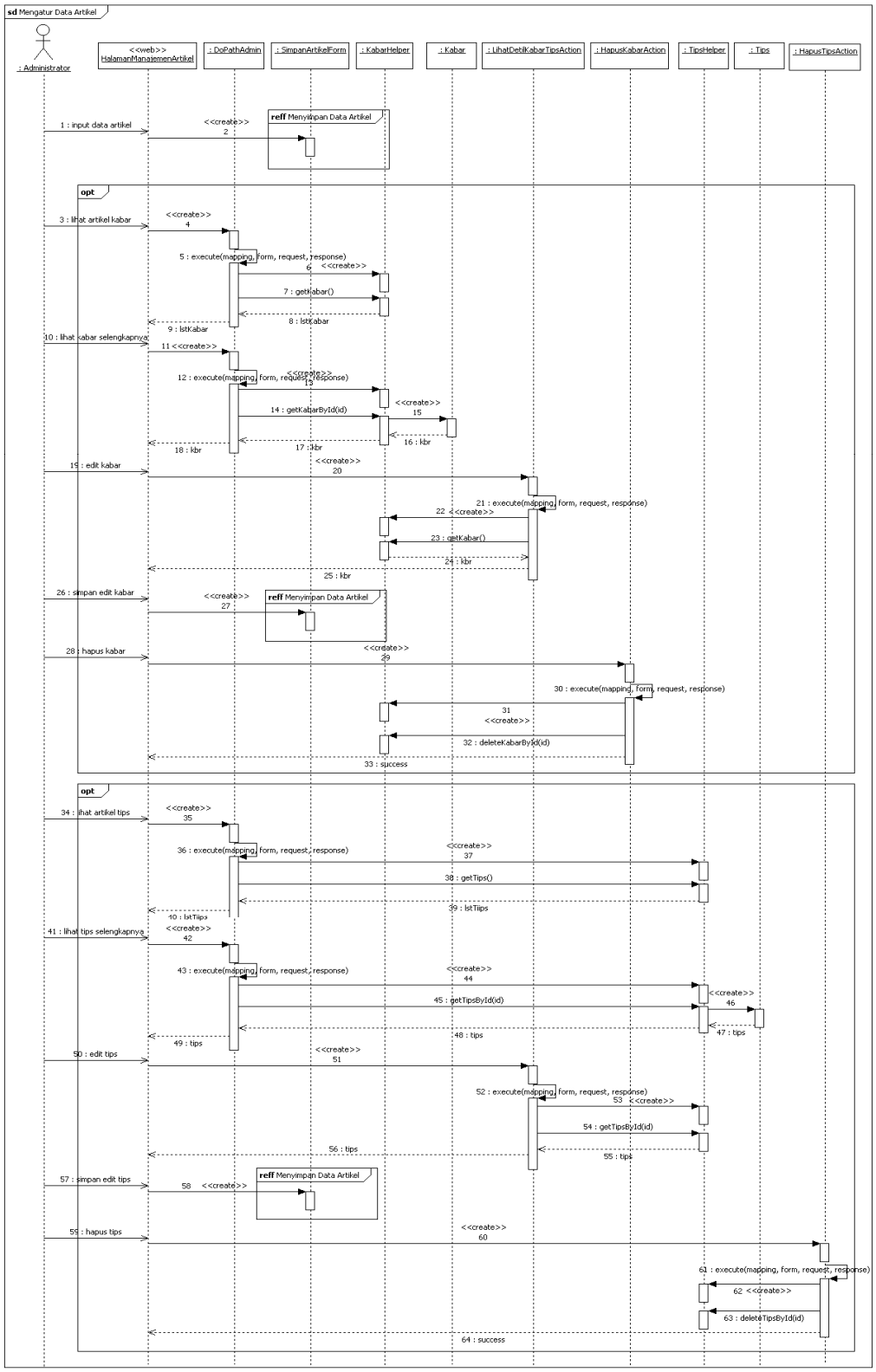
Diagram sekuensial ini (Gambar 4.27) menggambarkan interaksi yang terjadi ketika seorang Administrator mengubah akunnya.



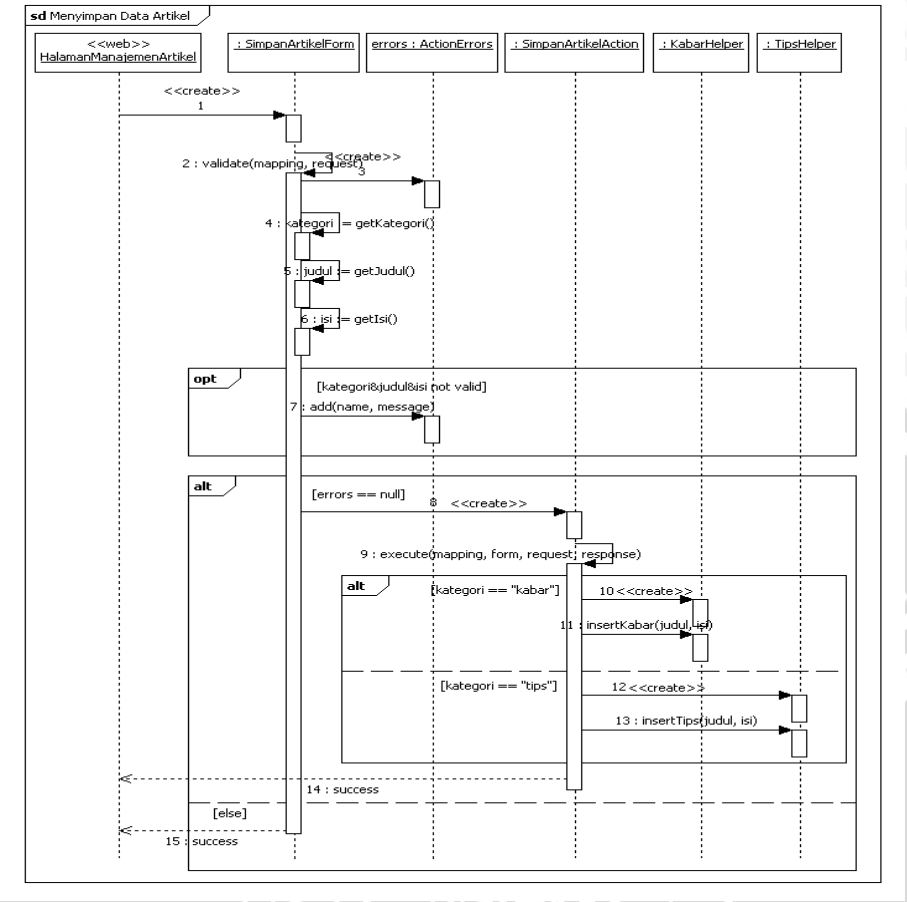
Gambar 4.27 Diagram sekuensial Mengubah Profil Administrator
Sumber: Perancangan

4.2.2.2.12. Diagram Sekuensial Mengatur Data Artikel

Diagram sekuensial ini (Gambar 4.28, 4.29) menggambarkan interaksi yang terjadi ketika seorang Administrator mengatur data artikel.



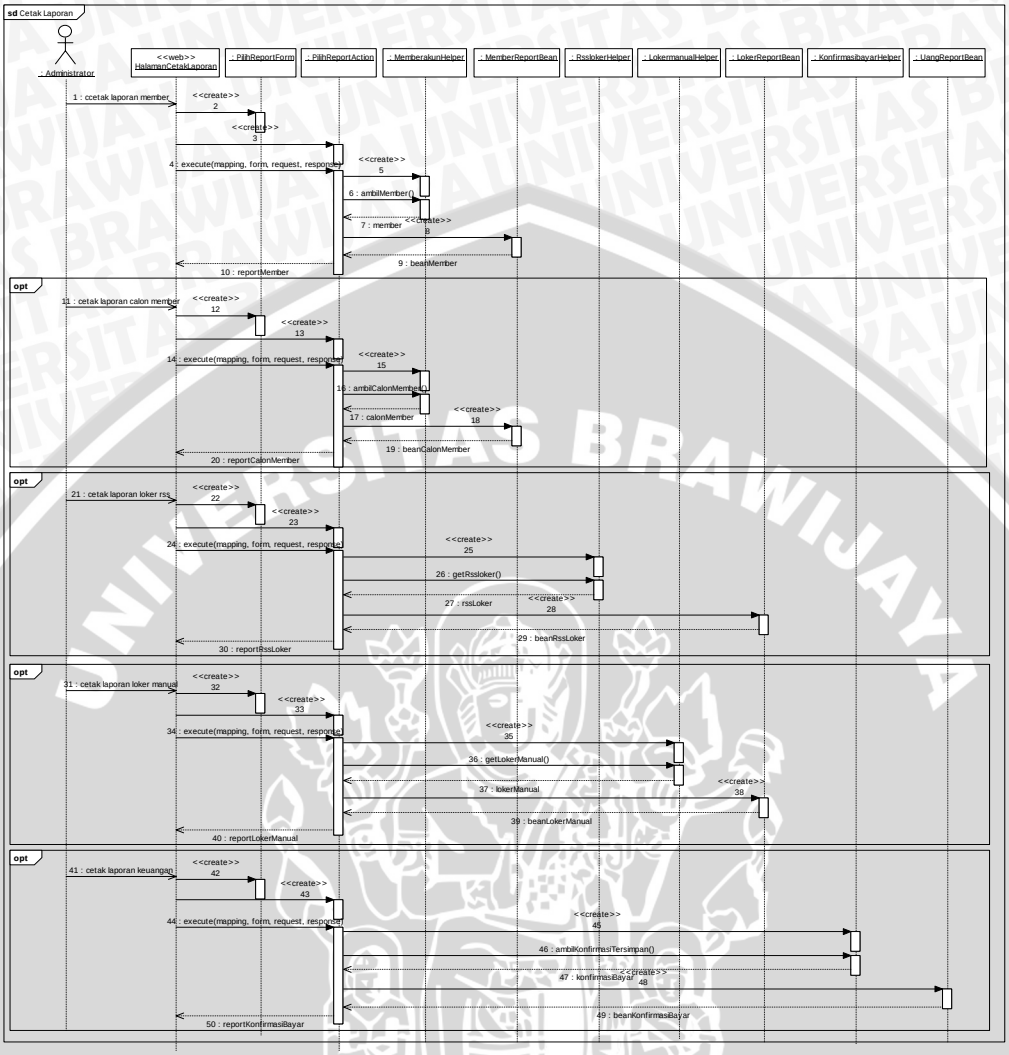
Gambar 4.28 Diagram sekuensial Mengatur Data Artikel
Sumber: Perancangan



Gambar 4.29 Diagram sekuensial Menyimpan Data Artikel
Sumber: Perancangan

4.2.2.2.13. Diagram Sekuensial Mencetak Laporan

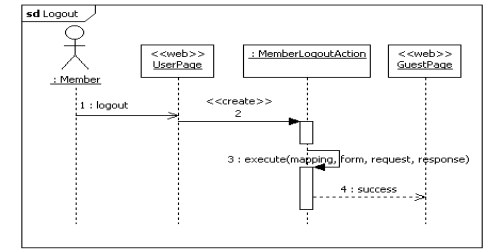
Interaksi antar klas yang mengimplementasikan *usecase* Mencetak Laporan ditunjukkan pada diagram sekuensial dalam Gambar 4.30. Laporan ditampilkan dalam format PDF, dengan opsi operasi disimpan dalam suatu file atau di cetak ke dalam *hardcopy.*



Gambar 4.30 Diagram sekuensial Cetak Laporan
Sumber: Perancangan

4.2.2.2.14. Diagram Sekuensial *Logout*

Diagram sekuensial ini (Gambar 4.31) menggambarkan interaksi yang terjadi ketika seorang Member atau Administrator keluar dari menu utama Member atau Administrator.



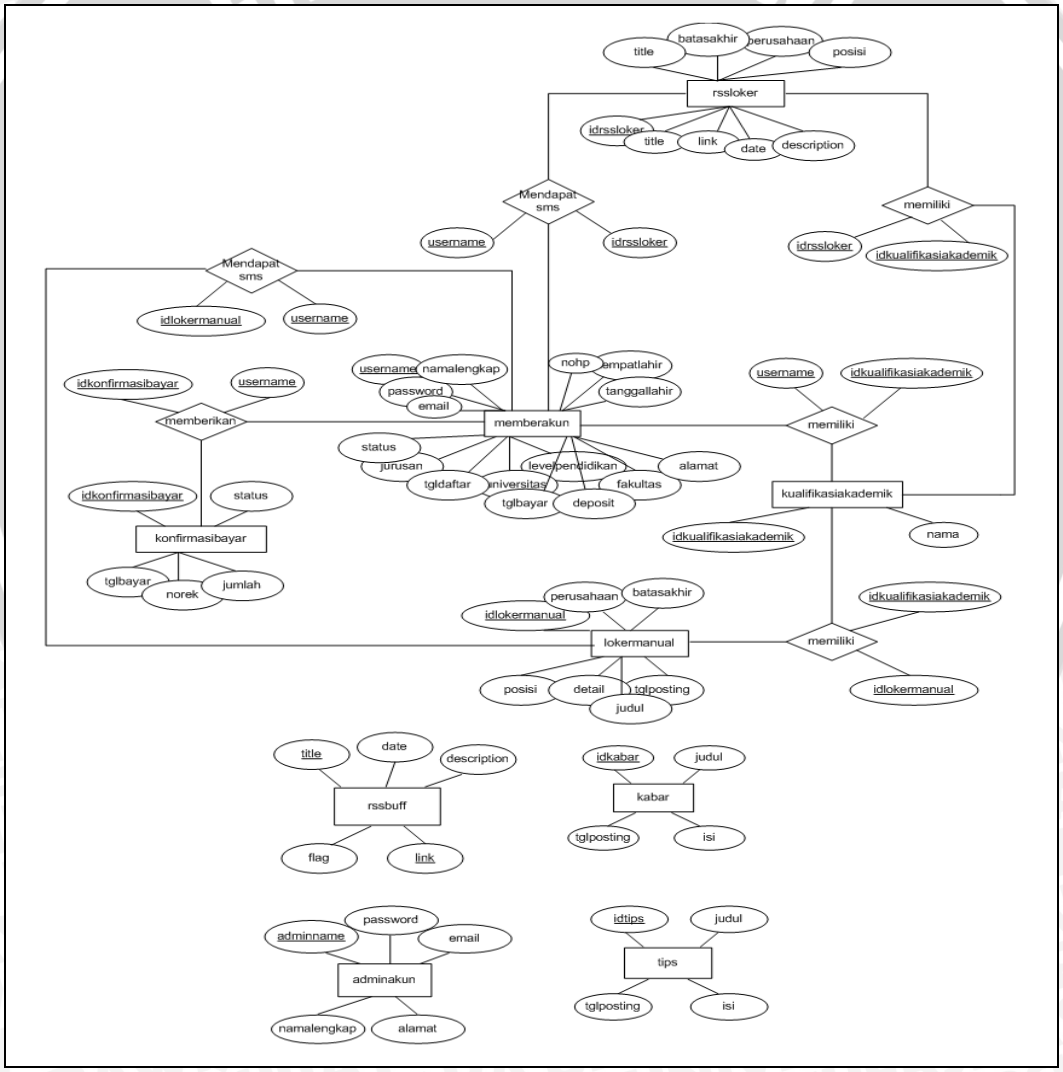
Gambar 4.31 Diagram sekuensial *Logout*
Sumber: Perancangan

4.2.3. Perancangan Basis Data

Perancangan basis data dilakukan agar basis data dapat efisien dalam penggunaan ruang penyimpanan, cepat dalam pengaksesan, dan mudah dalam pemanipulasian data. Perancangan basis data dapat dilakukan dengan menggunakan diagram ER, *data object description* dan normalisasi data.

4.2.3.1. *Entity-Relationship Diagram* (Diagram ER)

Perekayasa perangkat lunak menspesifikasikan basis data yang dipakai dan mendefinisikan hubungan antar tabel yang ada didalam basis data melalui sebuah hubungan entitas. Diagram E-R dari basis data aplikasi ini ditunjukkan pada Gambar 4.32.



Gambar 4.32 ER Diagram
Sumber: Perancangan

4.2.3.2. Deskripsi Data

Desain database yang dimodelkan dalam *Entity Relationship Diagram* pada Gambar 4.32 diimplementasikan menjadi beberapa tabel, yaitu tabel **`adminakun`, `kabar`, `tips`, `konfirmasibayar`, `kualifikasiakademik`, `lokermanual`, `lokermanualka`, `memberakun`, `rssbuff`, `rssloker`, `rsslokerka`, `smslokermanual`, `smslokerrss`.**

4.2.3.2.1. Tabel `adminakun`

Tabel `adminakun` (Tabel 4.17) memiliki lima *field* yaitu *field* `password`, `namalengkap`, `email`, `alamat`, serta `adminname` sebagai *primary key*. *Field* tersebut digunakan untuk menyimpan data *admin* dalam *database.* Semua *field* pada tabel ini tidak diperbolehkan kosong.

Tabel 4.17 *Data Object Description* tabel `adminakun`

Field	Tipe	Panjang	Keterangan
password	varchar	35	NOT NULL
namalengkap	varchar	50	NOT NULL
email	Varchar	45	NOT NULL
alamat	Varchar	80	NOT NULL
adminname	Varchar	25	NOT NULL, PRIMARY KEY

Sumber: Perancangan

4.2.3.2.2. Tabel `kabar`

Tabel `kabar` (Tabel 4.18) memiliki empat *field* yaitu *field* `idkabar` sebagai *primary key*, `judul`, `tglposting`, serta `isi`. *Field* tersebut digunakan untuk menyimpan data *kabar* dalam *database.* Semua *field* pada tabel ini tidak diperbolehkan kosong.

Tabel 4.18 *Data Object Description* tabel `kabar`

Field	Tipe	Panjang	Keterangan
idkabar	Int	6	NOT NULL, PRIMARY KEY, AUTO INCREMENT
judul	Varchar	100	NOT NULL
tglposting	Varchar	30	NOT NULL
isi	Long text		NOT NULL

Sumber: Perancangan

4.2.3.2.3. Tabel `konfirmasibayar`

Tabel `konfirmasibayar` (Tabel 4.19) memiliki enam *field* yaitu *field* `idkonfirmasibayar` sebagai *primary key*, *field* `tglbayar`, `jumlah`, `norek`, `username` sebagai *foreign key* dan `status` digunakan untuk menyimpan data

konfirmasi bayar member dalam *database.* Semua *field* pada tabel ini tidak diperbolehkan kosong.

Tabel 4.19 *Data Object Description* tabel `konfirmasibayar`

Field	Tipe	Panjang	Keterangan
idkonfirmasibayar	integer		NOT NULL, AUTO_INCREMENT, PRIMARY KEY
tglbayar	varchar	30	NULL
jumlah	Double		NOT NULL
norek	Varchar	25	NOT NULL
username	Varchar	25	NULL, FOREIGN KEY
status	Varchar	15	NULL

Sumber: Perancangan

4.2.3.2.4. Tabel `kualifikasiakademik`

Tabel `kualifikasiakademik` (Tabel 4.20) memiliki dua *field* yaitu *field* `idkualifikasiakademik` sebagai *primary key* dan *field* `nama` digunakan untuk menyimpan data kualifikasi akademik dalam *database.* Semua *field* pada tabel ini tidak diperbolehkan kosong.

Tabel 4.20 *Data Object Description* tabel `kualifikasiakademik`

Field	Tipe	Panjang	Keterangan
idkualifikasiakademik	Integer		NOT NULL, AUTO_INCREMENT, PRIMARY KEY
nama	Varchar	30	NOT NULL

Sumber: Perancangan

4.2.3.2.5. Tabel `lokermanual`

Tabel `lokermanual` (Tabel 4.21) memiliki tujuh *field* yaitu *field* `idlokermanual` (*primary key*), `perusahaan`, `batasakhir`, `posisi`, `tglposting`, `detail` dan `judul`.

Tabel 4.21 *Data Object Description* tabel `lokermanual`

Field	Tipe	Panjang	Keterangan
Idlokermanual	int	5	NOT NULL, AUTO_INCREMENT, PRIMARY KEY
Perusahaan	varchar	50	NOT NULL
Batasakhir	varchar	45	NOT NULL
Posisi	varchar	55	NOT NULL
Tglposting	Varchar	30	NOT NULL
Detail	Long text		NOT NULL
Judul	Varchar	60	NOT NULL

Sumber: Perancangan

4.2.3.2.6. Tabel `lokermanualka`

Tabel `lokermanualka` (Tabel 4.22) memiliki dua *field* yaitu *field* `idlokermanual` sebagai *foreign key* dan *field* `idkualifikasiakademik` sebagai *foreign key*. Semua *field* pada tabel ini tidak diperbolehkan kosong.

Tabel 4.22 *Data Object Description* tabel `lokermanualka`

Field	Tipe	Panjang	Keterangan
Idlokermanual	integer		NOT NULL, FOREIGN KEY
idkualifikasiakademik	integer		NOT NULL, FOREIGN KEY

Sumber: Perancangan

4.2.3.2.7. Tabel `memberakun`

Tabel `memberakun` (Tabel 4.23) memiliki delapan belas *field* yaitu *field* `username` sebagai *primary key*, *field* `password`, `namalengkap`, `email`, `tempatlahir`, `tanggallahir`, `nohp`, `levelpendidikan`, `universitas`, `fakultas`, `jurusan`, `idkualifikasiakademik` sebagai *foreign key*, `status`, `alamat`, `tgldaftar`, `tglbayar`, `tglakhir`, dan `deposit` digunakan untuk menyimpan data member dalam *database.*

Tabel 4.23 *Data Object Description* tabel `memberakun`

Field	Tipe	Panjang	Keterangan
Username	varchar	25	NOT NULL, PRIMARY KEY
Password	varchar	35	NOT NULL
namalengkap	varchar	50	NOT NULL
email	varchar	45	NOT NULL
tempatlahir	varchar	20	NOT NULL
tanggallahir	varchar	15	NOT NULL
nohp	varchar	20	NOT NULL
levelpendidikan	varchar	5	NOT NULL
universitas	varchar	20	NOT NULL
fakultas	varchar	30	NOT NULL
jurusan	varchar	30	NOT NULL
idkualifikasiakademik	integer		NOT NULL, FOREIGN KEY
status	varchar	15	NOT NULL
alamat	varchar	80	NOT NULL
tgldaftar	varchar	30	NOT NULL
tglbayar	varchar	30	NOT NULL
tglakhir	varchar	45	NOT NULL
deposit	int	6	NOT NULL

Sumber: Perancangan

4.2.3.2.8. Tabel `rssbuff`

Tabel `rssbuff` (Tabel 4.24) memiliki lima *field* yaitu *field* `title` sebagai *primary key*, *field* `date`, `description`, `link` sebagai *primary key*, dan `flag` digunakan untuk menyimpan data informasi lowongan kerja dari RSS web lain sebelum diseleksi Administrator. Semua *field* pada tabel ini tidak diperbolehkan kosong.

Tabel 4.24 *Data Object Description* tabel `rssbuff`

Field	Tipe	Panjang	Keterangan
title	varchar	70	NOT NULL, PRIMARY KEY
date	varchar	45	NOT NULL
description	varchar	200	NOT NULL
link	varchar	60	NOT NULL, PRIMARY KEY
flag	varchar	15	NOT NULL

Sumber: Perancangan

4.2.3.2.9. Tabel `rssloker`

Tabel `rssloker` (Tabel 4.25) memiliki sembilan *field* yaitu *field* `idrssloker` sebagai *primary key*, *field* `title`, `link`, `date`, `description`, `perusahaan`, `batasakhir`, `posisi`, dan `posting` digunakan untuk menyimpan data informasi lowongan kerja dalam *database*. Semua *field* pada tabel ini tidak diperbolehkan kosong.

Tabel 4.25 *Data Object Description* tabel `rssloker`

Field	Tipe	Panjang	Keterangan
idrssloker	integer		NOT NULL, AUTO_INCREMENT, PRIMARY KEY
title	varchar	70	NOT NULL
link	varchar	60	NOT NULL
date	varchar	45	NOT NULL
description	varchar	200	NOT NULL
perusahaan	varchar	60	NOT NULL
batasakhir	varchar	30	NOT NULL
posisi	varchar	55	NOT NULL
posting	Varchar	30	NOT NULL

Sumber: Perancangan

4.2.3.2.10. Tabel `rsslokerka`

Tabel `rsslokerka` (Tabel 4.26) memiliki dua *field* yaitu *field* `idrssloker` sebagai *foreign key*, dan *field* `idkualifikasiakademik` *foreign key*. Semua *field* pada tabel ini tidak diperbolehkan kosong.

Tabel 4.26 *Data Object Description* tabel `rsslokerka`

Field	Tipe	Panjang	Keterangan
idrssloker	Integer		NOT NULL, FOREIGN KEY
idkualifikasiakademik	Integer		NOT NULL, FOREIGN KEY

Sumber: Perancangan

4.2.3.2.11. Tabel `smslokermanual`

Tabel `smslokermanual` (Tabel 4.27) memiliki dua *field* yaitu *field* `username` sebagai *foreign key* dan *field* `idlokermanual` sebagai *foreign key*. Semua *field* pada tabel ini tidak diperbolehkan kosong.

Tabel 4.27 *Data Object Description* tabel `smslokermanual`

Field	Tipe	Panjang	Keterangan
Username	Varchar	50	NOT NULL, FOREIGN KEY
Idlokermanual	Integer		NOT NULL, FOREIGN KEY

Sumber: Perancangan

4.2.3.2.12. Tabel `smslokerrss`

Tabel `smslokerrss` (Tabel 4.28) memiliki dua *field* yaitu *field* `username` sebagai *foreign key* dan *field* `idrssloker` sebagai *foreign key*. Semua *field* pada tabel ini tidak diperbolehkan kosong.

Tabel 4.28 *Data Object Description* tabel `smslokerrss`

Field	Tipe	Panjang	Keterangan
Username	varchar	50	NOT NULL, FOREIGN KEY
Idrssloker	integer		NOT NULL, FOREIGN KEY

Sumber: Perancangan

4.2.3.2.13. Tabel `tips`

Tabel `tips` (Tabel 4.29) memiliki empat *field* yaitu *field* `idtips` sebagai *primary key*, *field* `judul`, `tglposting` dan `isi` digunakan untuk menyimpan data artikel kategori tips dalam *database*. Semua *field* pada tabel ini tidak diperbolehkan kosong.

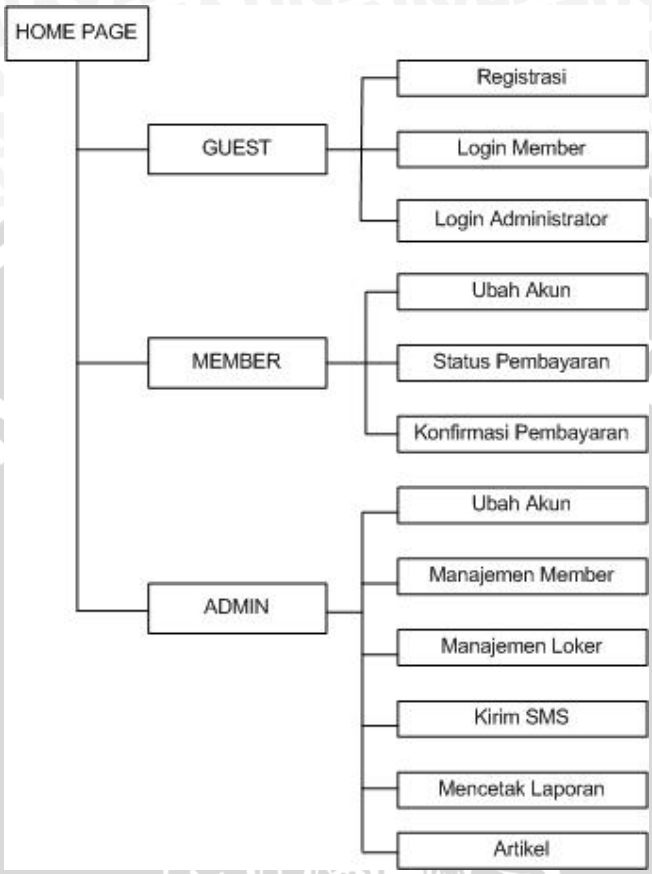
Tabel 4.29 *Data Object Description* tabel `tips`

Field	Tipe	Panjang	Keterangan
idtips	integer	6	NOT NULL, AUTO_INCREMENT, PRIMARY KEY
judul	varchar	100	NOT NULL
tglposting	varchar	30	NOT NULL
isi	long text		NOT NULL

Sumber: Perancangan

4.2.4. Perancangan Antarmuka

Proses perancangan antarmuka ini terdiri atas perancangan menu navigasi dan *layout* tampilan. Gambar 4.33 merupakan menu navigasi aplikasi ini.

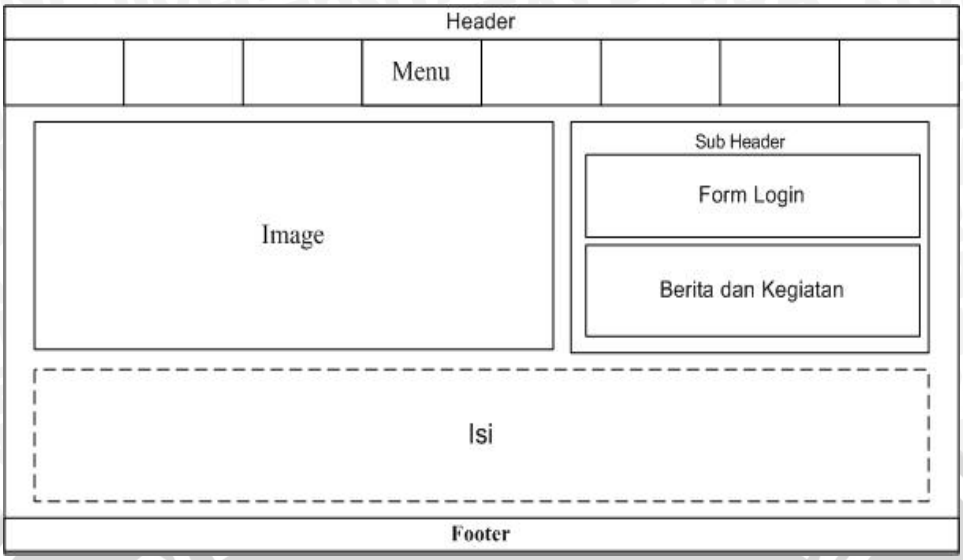


Gambar 4.33 Menu navigasi
Sumber: Perancangan

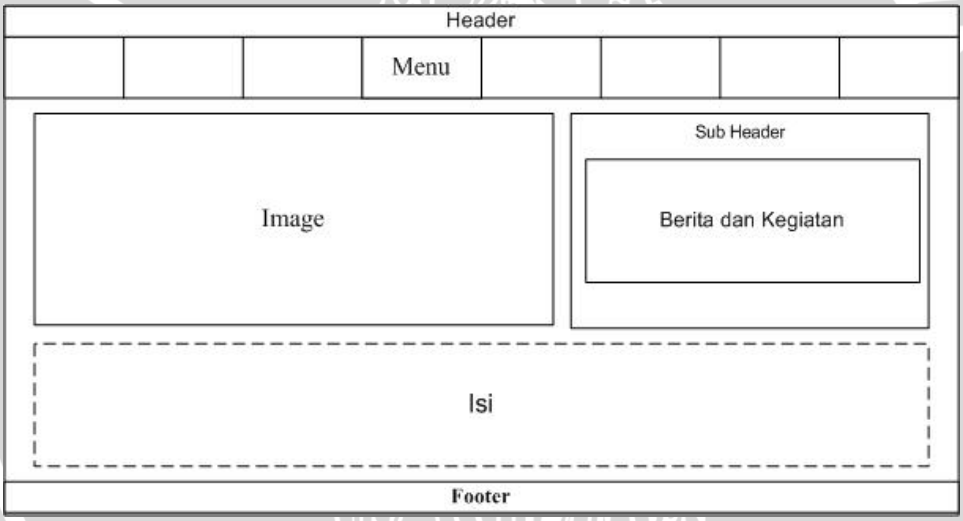
Layout aplikasi ini dibagi menjadi tiga bagian antara lain *layout* untuk Guest, Member, dan Administrator.

4.2.4.1. Perancangan Antarmuka Guest

Perancangan antarmuka Guest terdiri atas tiga bagian yaitu Registrasi, Login Member, dan Login Administrator. Pada halaman Guest, *layout* terdiri atas header, menu, image, sub header, isi dan footer, pada halaman depan untuk sub header terbagi lagi menjadi form *login* dan form kabar dan kegiatan, diperlihatkan pada Gambar 4.34 sedangkan untuk halaman menu yang lain, pada sub header hanya terdapat form kabar dan kegiatan saja, diperlihatkan pada Gambar 4.35



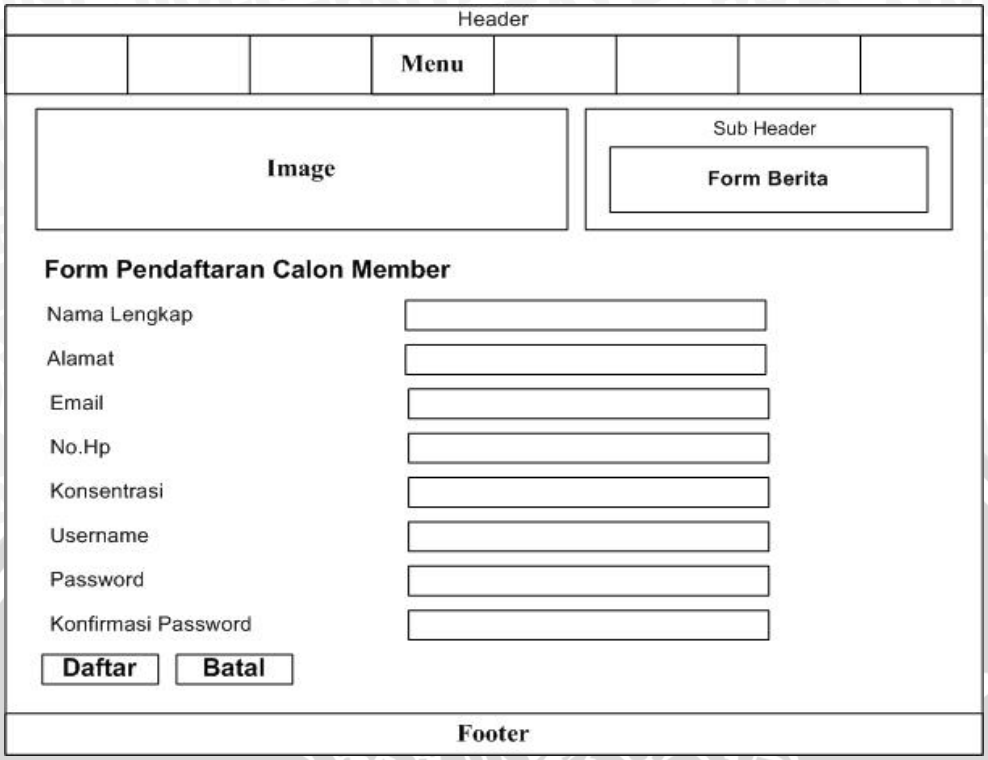
Gambar 4.34 Rancangan Antarmuka Guest pada halaman utama
Sumber: Perancangan



Gambar 4.35 Rancangan Antarmuka Guest selain halaman utama
Sumber: Perancangan

4.2.4.1.1 Registrasi Member

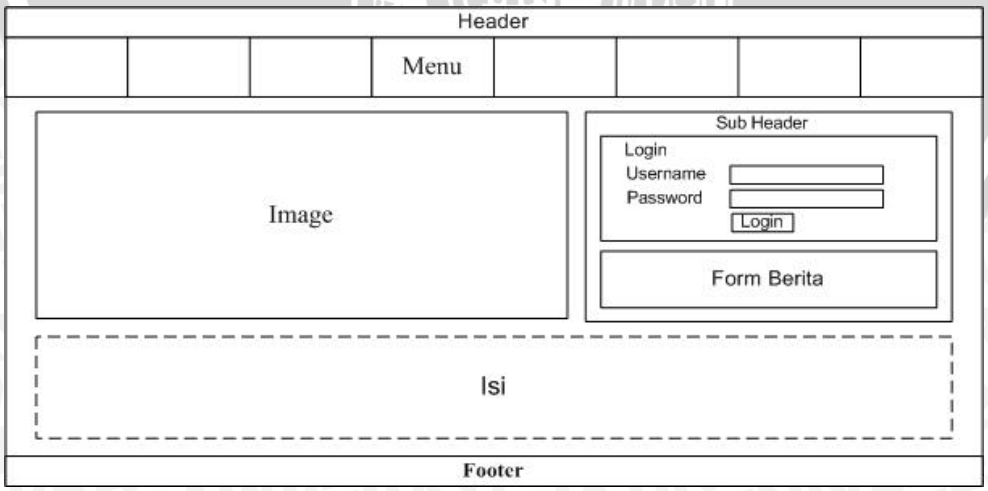
Pengguna harus mempunyai akun untuk mengakses sistem. Pengguna yang belum mempunyai akun dapat melakukan proses registrasi sehingga dapat melakukan *login* dengan akun baru tersebut. *Layout* halaman registrasi diperlihatkan pada Gambar 4.36



Gambar 4.36 Rancangan Antarmuka Registrasi Member
Sumber: Perancangan

4.2.4.1.2 Login Member

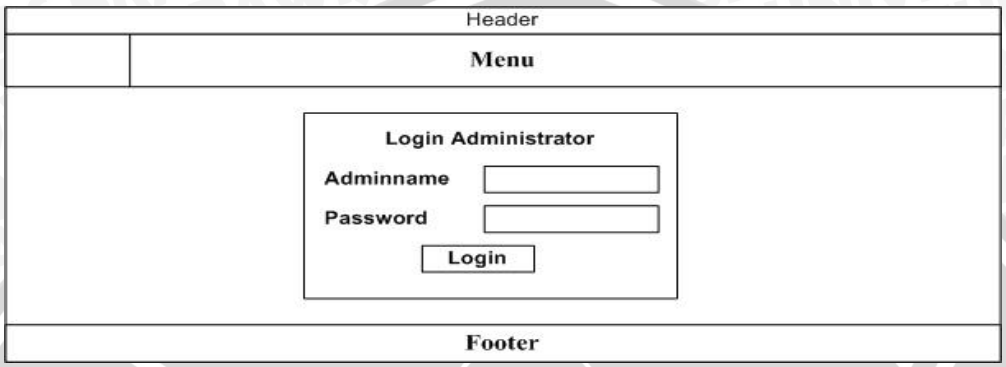
Untuk dapat mengakses sistem sebagai Member, pengguna harus melakukan proses *login* terlebih dahulu. Proses *login* memerlukan *username* dan *password* yang sesuai dengan *username* dan *password* Member. *Layout* halaman proses *login* diperlihatkan pada Gambar 4.37.



Gambar 4.37 Rancangan Antarmuka Login Member
Sumber: Perancangan

4.2.4.1.3 Login Administrator

Untuk dapat mengakses sistem sebagai Administrator, pengguna harus melakukan proses *login* terlebih dahulu. Proses *login* memerlukan *username* dan *password* yang sesuai dengan *username* dan *password* Administrator. *Layout* halaman proses *login* diperlihatkan pada Gambar 4.38

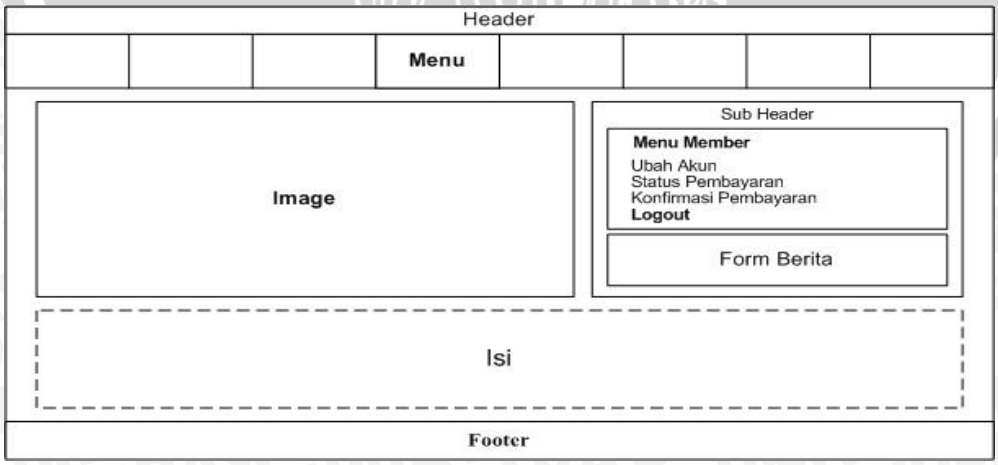


Gambar 4.38 Rancangan Antarmuka Login Administrator
Sumber: Perancangan

4.2.4.2. Perancangan Antarmuka Member

Perancangan antarmuka Member terdiri atas tiga bagian yaitu Ubah Akun, Konfirmasi Pembayaran, dan Status Pembayaran. Untuk mengakses halaman ini, pengguna harus *login* sebagai Member

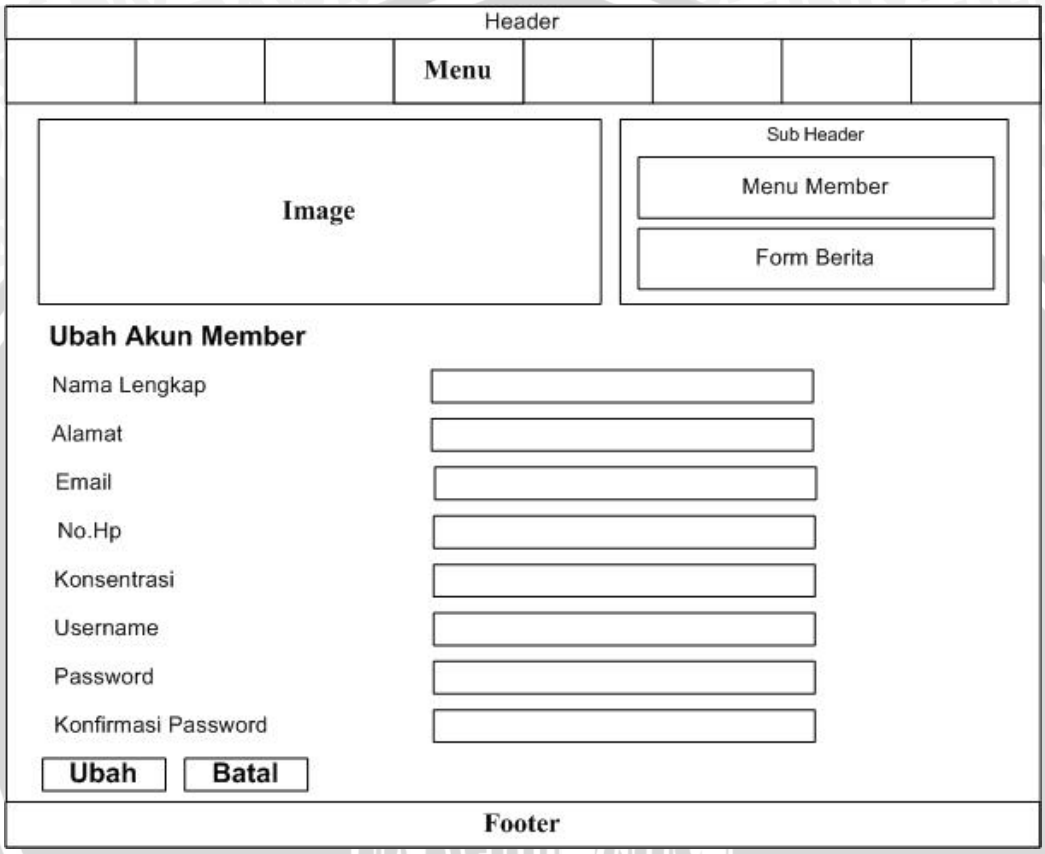
Untuk semua menu pada halaman member, sub header terbagi menjadi dua yaitu form menu member dan form berita. Form menu member menyediakan beberapa fitur antara lain fitur untuk mengubah akun, melihat status pembayaran, konfirmasi pembayaran dan *Logout*. *Layout* halaman utama Member diperlihatkan pada Gambar 4.39



Gambar 4.39 Rancangan Antarmuka Halaman Utama Member
Sumber: Perancangan

4.2.4.2.1 Ubah Akun

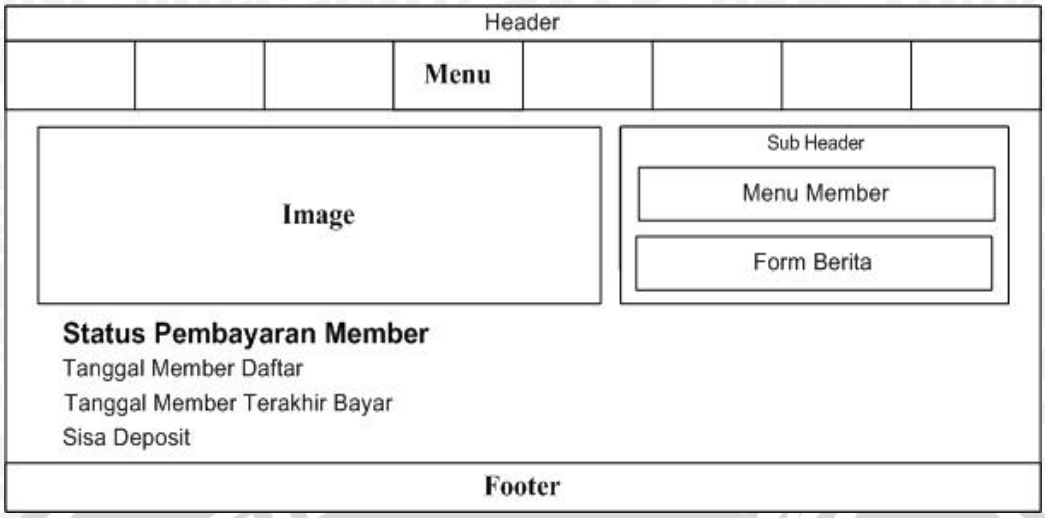
Pengguna yang ingin mengubah akunnya dapat mengakses halaman ini. Akun yang dapat diubah antara lain nama lengkap, alamat, email, no.hp, konsentrasi, username, password dan konfirmasi password. *Layout* halaman ini diperlihatkan pada Gambar 4.40



Gambar 4.40 Rancangan Antarmuka Mengubah Profil Member
Sumber: Perancangan

4.2.4.2.2 Status Pembayaran

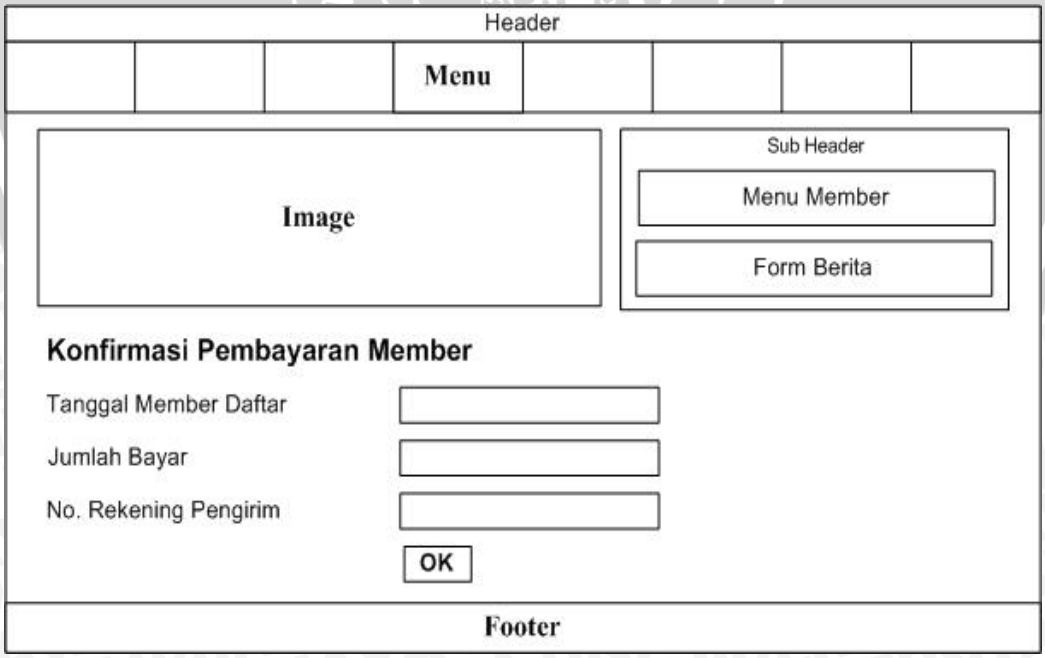
Pada halaman ini, pengguna dapat melihat status pembayaran dengan melihat sisa deposit. Apabila sisa deposit 0, maka pengguna harus melakukan pembayaran jika ingin tetap berlangganan. *Layout* halaman Status Pembayaran diperlihatkan pada Gambar 4.41.



Gambar 4.41 Rancangan Antarmuka Status Pembayaran Member
Sumber: Perancangan

4.2.4.2.3 Konfirmasi Pembayaran

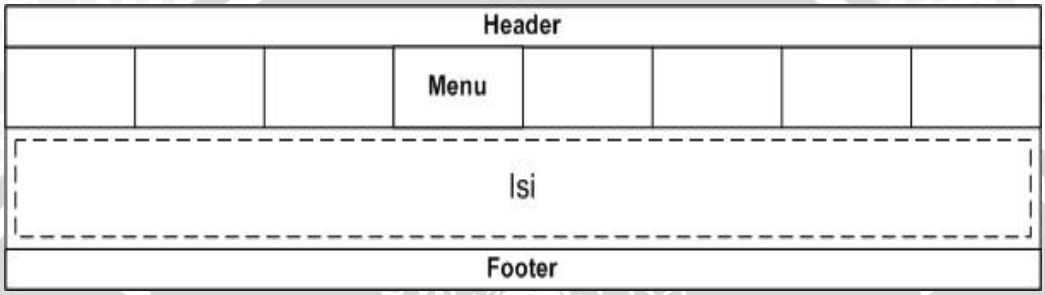
Pada halaman ini, pengguna dapat melakukan konfirmasi pembayaran dengan memasukkan data berupa tanggal member daftar, jumlah bayar dan no.rekening pengirim setelah pengguna melakukan pembayaran ke rekening Administrator. *Layout* halaman Konfirmasi Pembayaran diperlihatkan pada Gambar 4.42.



Gambar 4.42 Rancangan Antarmuka Konfirmasi Pembayaran
Sumber: Perancangan

4.2.4.3. Perancangan Antarmuka Administrator

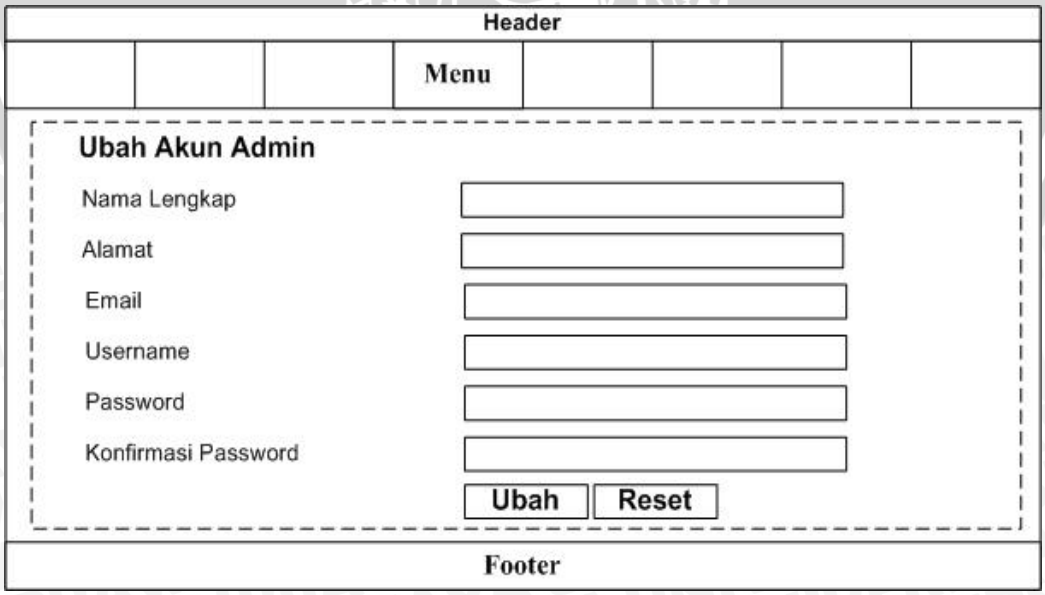
Perancangan antarmuka Administrator terdiri atas empat bagian yaitu Ubah Akun, Manajemen Member, Manajemen Loker dan Kirim SMS. Untuk mengakses halaman ini, pengguna harus *login* sebagai Administrator. Halaman Administrator terdiri dari header, menu, isi dan footer dengan isi menyesuaikan dengan menu yang dipilih. *Layout* halaman utama Administrator diperlihatkan pada Gambar 4.43



Gambar 4.43 Rancangan Antarmuka Halaman Utama Administrator
Sumber: Perancangan

4.2.4.3.1 Ubah Akun

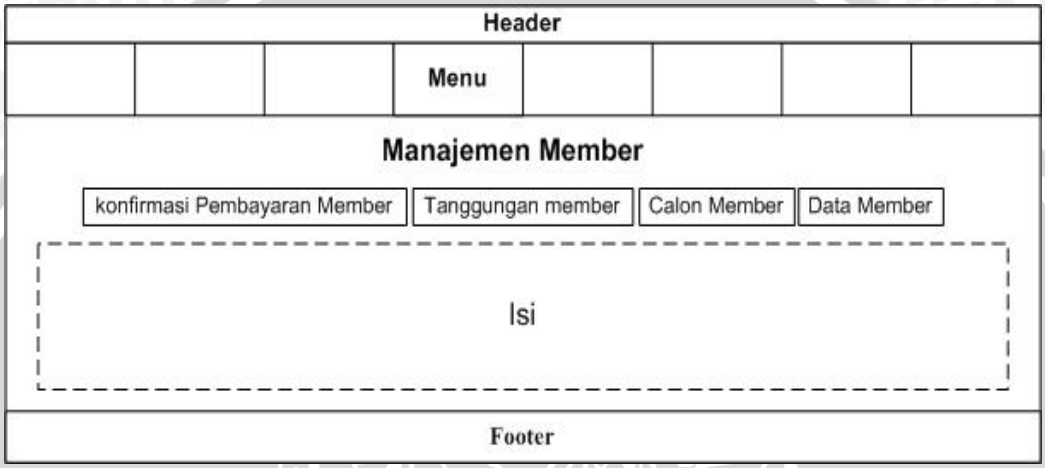
Administrator yang ingin mengubah akunnya dapat mengakses halaman ini. Akun yang dapat diubah antara lain nama Nama Lengkap, Alamat, *email*, *username*, *password* Konfirmasi *Password*. *Layout* halaman proses pengubahan akun Administrator diperlihatkan pada Gambar 4.44



Gambar 4.44 Rancangan Antarmuka Ubah Profil Administrator
Sumber: Perancangan

4.2.4.3.2 Manajemen Member

Halaman Manajemen Member menyediakan beberapa fitur antara lain fitur untuk Konfirmasi Pembayaran, Lihat Tanggungan Member, Data Calon Member, dan Data Member. Pada halaman ini, Administrator dapat melakukan proses manajemen Member antara lain mengkonfirmasi pembayaran member, penambahan, pengubahan, dan penghapusan Member. *Layout* halaman manajemen Member diperlihatkan pada Gambar 4.45



Gambar 4.45 Rancangan Antarmuka Manajemen Member
Sumber: Perancangan

Bagian isi akan menampilkan halaman sesuai dengan *link* yang dipilih. Adapun isi dari masing-masing fitur adalah sebagai berikut :

- **Konfirmasi Pembayaran Member**

  Pada halaman ini, Administrator dapat melihat konfirmasi pembayaran dari member dan mengkonfirmasinya guna meng*update* sisa deposit member. *Layout* isi dari halaman link ini ditunjukkan pada Gambar 4.46

Nama Member	Tgl Transfer	Jumlah Bayar	No.Rekening	Konfirmasi	Hapus
Nama Member	Tgl Transfer	Jumlah Bayar	No.Rekening	Konfirmasi	Hapus
Nama Member	Tgl Transfer	Jumlah Bayar	No.Rekening	Konfirmasi	Hapus

Gambar 4.46 Rancangan *layout* isi Melihat Konfirmasi Pembayaran Member
Sumber: Perancangan

- **Lihat Tanggungan Member**

  Pada halaman ini, Administrator dapat melihat data member yang memiliki tanggungan pembayaran, yaitu memiliki sisa deposit 0 dan belum melakukan pembayaran. *Layout* isi dari halaman link ini ditunjukkan pada Gambar 4.47

Username	Nama Lengkap	Email	No.HP	Alamat	Tgl Akhir
Username	Nama Lengkap	Email	No.HP	Alamat	Tgl Akhir

Gambar 4.47 Rancangan *layout* isi Melihat Tanggunan Pembayaran Member
Sumber: Perancangan

- **Calon Member**

  Pada halaman ini, Administrator dapat melihat data calon member, melihat detail profilnya dengan mengklik username, dan berhak untuk menerima atau menolak menjadi member. *Layout* isi dari halaman link ini ditunjukkan pada Gambar 4.48

Username	Nama Lengkap	Email	No. HP	Alamat	Tgl Daftar	Terima	Tolak
Username	Nama Lengkap	Email	No. HP	Alamat	Tgl Daftar	Terima	Tolak
Username	Nama Lengkap	Email	No. HP	Alamat	Tgl Daftar	Terima	Tolak

Gambar 4.48 Rancangan *layout* isi Melihat Data Calon Member
Sumber: Perancangan

- **Data Member**

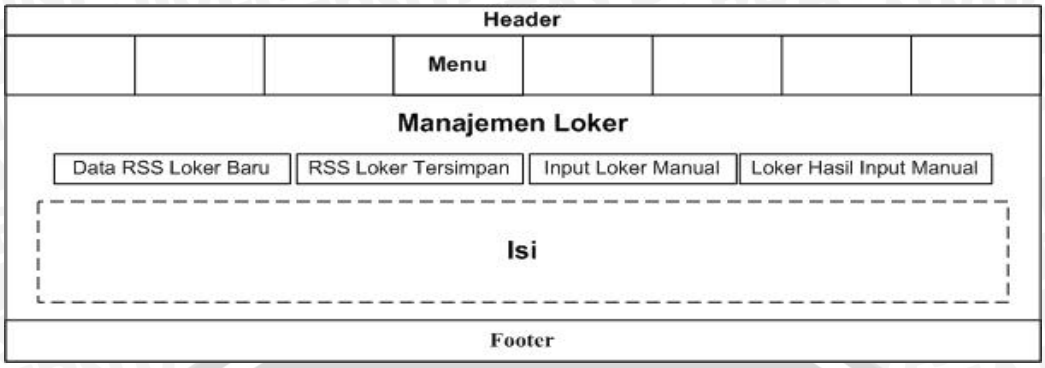
  Pada halaman ini, Administrator dapat melihat data seluruh member, melihat detail profilnya dengan mengklik username dan berhak untuk mengedit atau menghapus dari data member. *Layout* isi dari halaman link ini ditunjukkan pada Gambar 4.49

Username	Nama Lengkap	Email	No. HP	Alamat	Tgl Daftar	Edit	Hapus
Username	Nama Lengkap	Email	No. HP	Alamat	Tgl Daftar	Edit	Hapus
Username	Nama Lengkap	Email	No. HP	Alamat	Tgl Daftar	Edit	Hapus

Gambar 4.49 Rancangan *layout* isi Melihat Data Member
Sumber: Perancangan

4.2.4.3.3 Manajemen Loker (Lowongan Kerja)

Halaman Manajemen Loker menyediakan beberapa fitur antara lain fitur untuk Data RSS Loker Baru, Data RSS Loker Tersimpan, Input Loker Manual, Hasil Input Loker Manual. Pada halaman ini, Administrator dapat melakukan proses manajemen loker antara lain memilih dan menyeleksi informasi lowongan kerja dari RSS web lain serta mengelompokkannya berdasarkan kualifikasi akademik sebelum memasukkan infomasi tersebut ke dalam database, dapat melakukan penambahan, pengubahan, dan penghapusan informasi lowongan kerja, serta dapat memasukkan informasi lowongan kerja secara manual ke database. *Layout* halaman manajemen Loker diperlihatkan pada Gambar 4.50

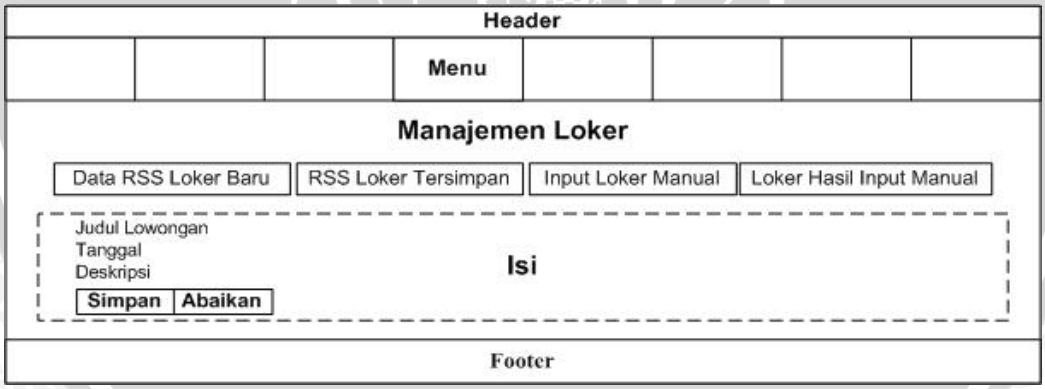


Gambar 4.50 Rancangan Antarmuka Manajemen Loker (Lowongan Kerja)
Sumber: Perancangan

Bagian isi akan menampilkan halaman sesuai dengan *link* yang dipilih. Adapun isi dari masing-masing fitur adalah sebagai berikut :

- **Data RSS Loker Baru**

  Halaman ini berisi informasi lowongan baru yang didapat dari membaca RSS web lain dan belum dikualifikasikan dalam spesifikasi akademik. Data yang tersimpan dalam loker ini harus diseleksi terlebih dahulu oleh Administrator sebelum disimpan di RSS Loker Tersimpan untuk ditampilkan kepada Guest atau Member. *Layout* isi dari halaman link ini ditunjukkan pada Gambar 4.51



Gambar 4.51 Rancangan Antarmuka Data RSS Loker Baru
Sumber: Perancangan

- **RSS Loker Tersimpan**

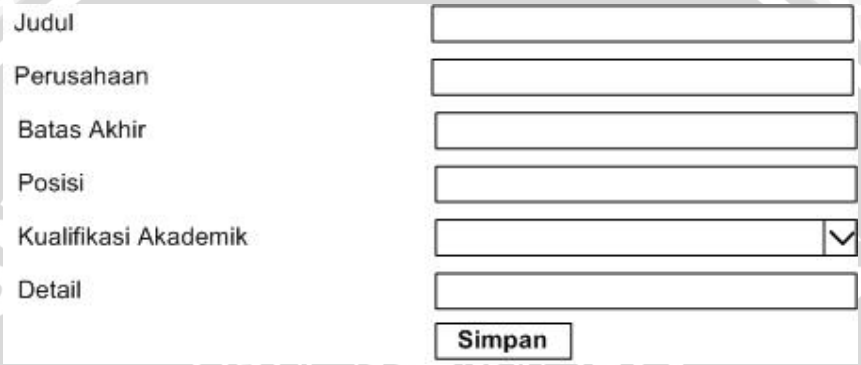
  Halaman ini berisi kumpulan informasi lowongan kerja yang telah diseleksi oleh Administrator dan telah diberi kualifikasi akademik. Informasi yang tersimpan pada loker inilah yang akan ditampilkan atau dikirimkan kepada Guest atau Member. *Layout* isi dari halaman link ini ditunjukkan pada Gambar 4.52

Judul	Tgl Posting	Perusahaan	Posisi	Batas Akhir	Kualifikasi Akademik	Ulasan	Edit	Hapus
Judul	Tgl Posting	Perusahaan	Posisi	Batas Akhir	Kualifikasi Akademik	Ulasan	Edit	Hapus
Judul	Tgl Posting	Perusahaan	Posisi	Batas Akhir	Kualifikasi Akademik	Ulasan	Edit	Hapus

Gambar 4.52 Rancangan *layout* isi Melihat RSS Loker yang telah Tersimpan
Sumber: Perancangan

- **Input Loker Manual**

  Halaman ini berisi *field-field* yang digunakan untuk memasukkan informasi lowongan kerja secara manual oleh Administrator. *Layout* isi dari halaman link ini ditunjukkan pada Gambar 4.53



Gambar 4.53 Rancangan *layout* isi Menyimpan RSS Loker
Sumber: Perancangan

- **Loker Hasil Input Manual**

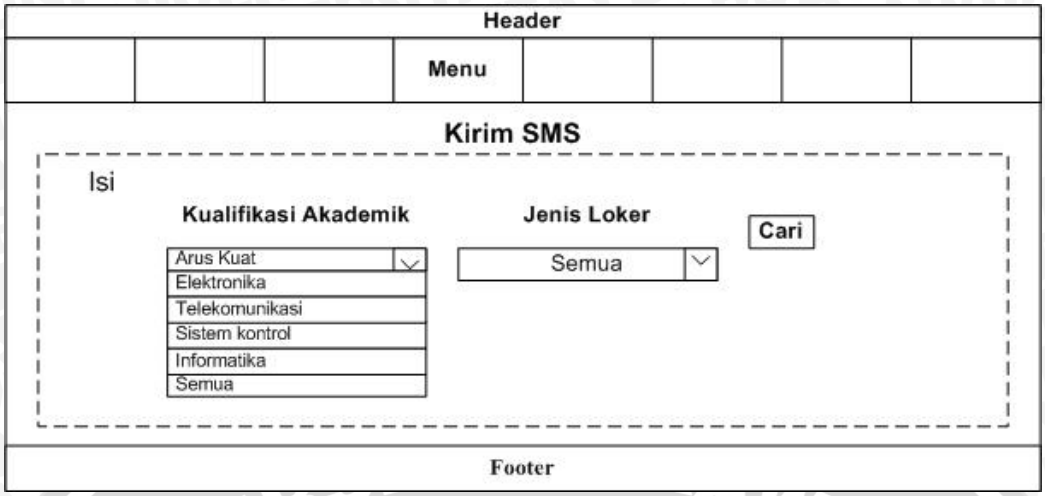
  Halaman ini berisi kumpulan informasi lowongan kerja yang diinputkan secara manual oleh Administrator. Informasi yang tersimpan pada loker ini juga akan ditampilkan atau dikirimkan kepada Guest atau Member. *Layout* isi dari halaman link ini ditunjukkan pada Gambar 4.54

Judul	Tgl Posting	Perusahaan	Posisi	Batas Akhir	Kualifikasi Akademik	Edit	Hapus
Judul	Tgl Posting	Perusahaan	Posisi	Batas Akhir	Kualifikasi Akademik	Edit	Hapus
Judul	Tgl Posting	Perusahaan	Posisi	Batas Akhir	Kualifikasi Akademik	Edit	Hapus

Gambar 4.54 Rancangan *layout* isi Melihat Loker Hasil Input Manual
Sumber: Perancangan

4.2.4.3.4 Kirim SMS

Halaman Kirim SMS ini digunakan untuk mengirimkan informasi lowongan kerja ke *Handphone* member. Pada halaman ini, Administrator dapat mencari Informasi lowongan kerja berdasarkan kualifikasi akademik dan jenis loker, Kemudian hasil pencarian akan ditampilkan dan Administrator dapat memilih mengirimkan informasi kerja per member atau ke semua member sekaligus. *Layout* halaman utama Kirim SMS diperlihatkan pada Gambar 4.55



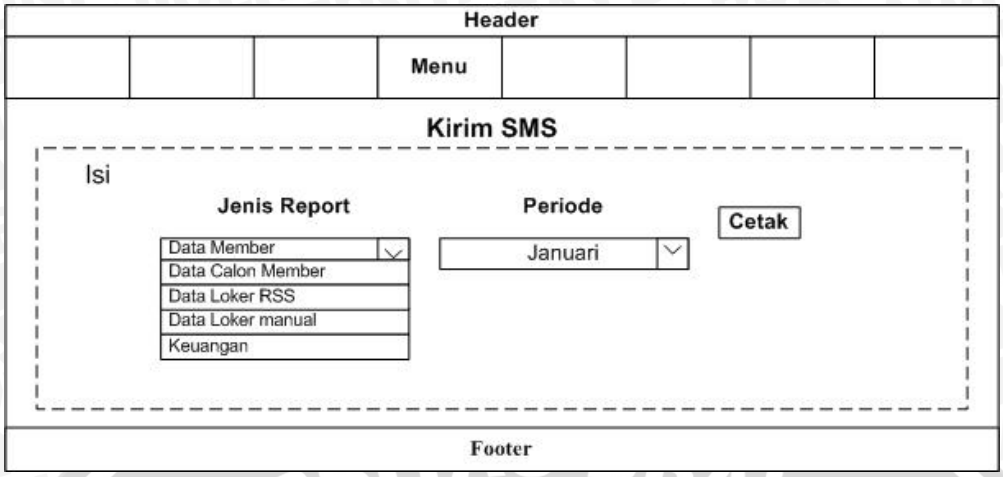
Gambar 4.55 Rancangan Antarmuka Halaman Utama Kirim SMS
Sumber: Perancangan

Setelah tombol cari ditekan, maka akan ditampilkan halaman sesuai kualifikasi akademik dan jenis loker yang dipilih. *Layout* ini diperlihatkan pada Gambar 4.56

Gambar 4.56 Rancangan Antarmuka Kirim SMS
Sumber: Perancangan

4.2.4.3.5 Mencetak Laporan

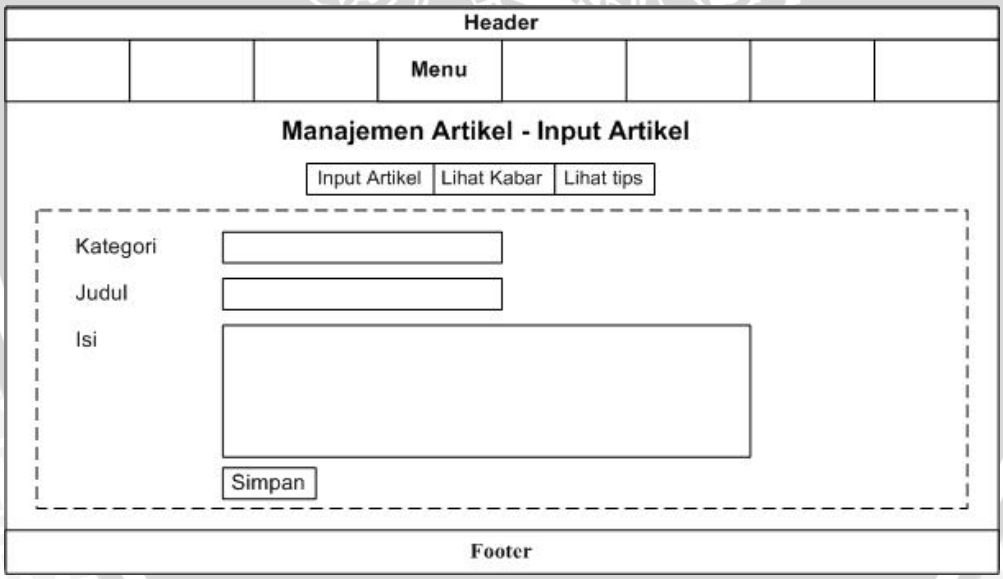
Halaman mencetak laporan ini digunakan untuk mencetak laporan baik berupa data member, data informasi lowongan kerja maupun keuangan. *Layout* halaman Mencetak laporan diperlihatkan pada Gambar 4.57



Gambar 4.57 Rancangan Antarmuka Mencetak Laporan
Sumber: Perancangan

4.2.4.3.6 Input Artikel

Halaman Artikel ini digunakan untuk memasukkan artikel baik kategori kabar maupun tips secara manual untuk ditampilkan pada halaman Member atau Guest. *Layout* halaman Input Artikel diperlihatkan pada Gambar 4.58



Gambar 4.58 Rancangan Antarmuka Artikel
Sumber: Perancangan

BAB V
IMPLEMENTASI

Bab ini membahas tentang implementasi perangkat lunak berdasarkan hasil yang telah didapatkan dari analisis kebutuhan dan proses perancangan perangkat lunak sebelumnya. Pembahasan terdiri dari penjelasan tentang spesifikasi lingkungan (spesifikasi perangkat keras dan perangkat lunak) di mana sistem diimplementasikan, implementasi tiap *class* pada *file* program, implementasi algoritma dan implementasi antarmuka aplikasi.

5.1 Spesifikasi Sistem

Dari hasil analisis kebutuhan dan perancangan yang diuraikan pada Bab 4, dilakukan implementasi agar sistem ini bisa berfungsi sesuai dengan kebutuhan. Spesifikasi sistem diimplementasikan pada spesifikasi perangkat keras dan perangkat lunak.

5.1.1 Spesifikasi Perangkat Keras (*Hardware*)

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

Tabel 5.1 Spesifikasi komputer untuk implementasi

Spesifikasi Komputer	
Merk Notebook	Lenovo 14002
Prosesor	Intel Pentium Dual CPU T2390 @ 1.86 GHz
Memori (RAM)	2 GB
Hardisk	160 GB, Hitachi
VGA Card	Onboard

Sumber: Implementasi

5.1.2 Spesifikasi Perangkat Lunak (*Software*)

Untuk membangun sistem ini dibutuhkan beberapa perangkat lunak yang dapat dilihat pada Tabel 5.2.

Tabel 5.2 Spesifikasi perangkat lunak untuk implementasi

Spesifikasi Perangkat Lunak	
Sistem Operasi	Windows XP Pro. SP. 2
Bahasa Pemrograman	Java Platform, Enterprise Edition
Lingkungan Pemrograman	Java Platform, Standard Edition 6 SDK Update 3 Java Platform, Enterprise Edition 5 SDK Update 3 MySQL 5.0.51b-community-nt Glass Fish
IDE *(Integrated Development Environment)*	NetBeans 6.5

Sumber: Implementasi

5.2 Implementasi Basis Data MySQL

Implementasi perancangan basis data `silokdb` dilakukan sesuai dengan *Entity Relationship Diagram*. Implementasi perancangan basis data menggunakan *query* SQL. *Query* SQL digunakan untuk mengimplementasikan rancangan basis data ke dalam sistem basis data MySQL. *Query* SQL yang digunakan dalam membentuk basis data `silokdb` ditunjukkan pada Gambar 5.1.

```
CREATE DATABASE silokdb;
```

Gambar 5.1 *Query* untuk membuat basis data `silokdb`
Sumber: Implementasi

Query SQL yang digunakan dalam membentuk tabel `adminakun` ditunjukkan pada Gambar 5.2.

```
CREATE TABLE adminakun(
        password VARCHAR(35)NOT NULL,
        namalengkap VARCHAR(50) NOT NULL,
        email VARCHAR(45) NOT NULL,
        alamat VARCHAR(80) NOT NULL,
        adminname VARCHAR(25) NOT NULL,
        PRIMARY KEY(adminname)
);
```

Gambar 5.2 *Query* untuk membuat tabel `adminakun`
Sumber: Implementasi

Query SQL yang digunakan dalam membentuk tabel `kabar` ditunjukkan pada Gambar 5.3.

```
CREATE TABLE kabar(
        idkabar INT(6) UNSIGNED NOT NULL AUTO_INCREMENT,
        judul VARCHAR(100) NOT NULL,
        tglposting VARCHAR(30) NOT NULL,
        isi LONGTEXT NOT NULL
        PRIMARY KEY(image_id)
);
```

Gambar 5.3 *Query* untuk membuat tabel `kabar`
Sumber: Implementasi

Query SQL yang digunakan dalam membentuk tabel `konfirmasibayar` ditunjukkan pada Gambar 5.4.

```
CREATE TABLE konfirmasibayar(
        idkonfirmasibayar INT UNSIGNED NOT NULL AUTO_INCREMENT,
        tglbayar VARCHAR(30),
        jumlah DOUBLE NOT NULL,
        norek VARCHAR(25),
        username VARCHAR(25),
        status VARCHAR(15),
        PRIMARY KEY(idkonfirmasibayar)
        FOREIGN KEY(username) REFERENCES memberakun(username)
        );
```

Gambar 5.4 *Query* untuk membuat tabel `konfirmasibayar`
Sumber: Implementasi

Query SQL yang digunakan dalam membentuk tabel `kualifikasiakademik` ditunjukkan pada Gambar 5.5.

```
CREATE TABLE kualifikasiakademik(
        idkualifikasiakademik INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
        nama VARCHAR(30) NOT NULL,
        PRIMARY KEY(idkualifikasiakademik)
);
```

Gambar 5.5 *Query* untuk membuat tabel `kualifikasiakademik`

Sumber: Implementasi

Query SQL yang digunakan dalam membentuk tabel `lokermanual` ditunjukkan pada Gambar 5.6.

```
CREATE TABLE lokermanual(
        idlokermanual INT(5) UNSIGNED NOT NULL AUTO_INCREMENT,
        perusahaan VARCHAR(50) NOT NULL,
        batasakhir VARCHAR(45) NOT NULL,
        posisi VARCHAR(55) NOT NULL
        tglposting VARCHAR(30) NOT NULL
        detail LONGTEXT NOT NULL
        judul VARCHAR(60) NOT NULL
        PRIMARY KEY(image_id)
);
```

Gambar 5.6 *Query* untuk membuat tabel `lokermanual`

Sumber: Implementasi

Query SQL yang digunakan dalam membentuk tabel `memberakun` ditunjukkan pada Gambar 5.7.

```
CREATE TABLE memberakun(
        Username VARCHAR(25) NOT NULL,
        password VARCHAR(35)INT(9) NOT NULL,
        namalengkap VARCHAR(50)INT(15) NOT NULL,
        email VARCHAR(45) NOT NULL,
        tempatlahir VARCHAR(20) NOT NULL,
        tanggallahir VARCHAR(15) NOT NULL,
        nohp VARCHAR(20) NOT NULL,
        levelpendidikan VARCHAR(5) NOT NULL,
        universitas VARCHAR(20) NOT NULL,
        fakultas VARCHAR(30) NOT NULL,
        jurusan VARCHAR(30) NOT NULL,
        idkualifikasiakademik INTEGER NOT NULL UNSIGNED,
        status VARCHAR(15) NOT NULL,
        alamat VARCHAR(80) NOT NULL,
        tgldaftar VARCHAR(30) NOT NULL,
        tglbayar VARCHAR(30) NOT NULL,
        deposit INT(6) NOT NULL UNSIGNED,
        PRIMARY KEY(username)
        FOREIGN KEY(idkualifikasiakademik) REFERENCES
kualifikasiakademik(idkualifikasiakademik)
```

Gambar 5.7 *Query* untuk membuat tabel `memberakun`

Sumber: Implementasi

Query SQL yang digunakan dalam membentuk tabel `rssbuff` ditunjukkan pada Gambar 5.8.

```
CREATE TABLE rssbuff(
        title VARCHAR(70) NOT NULL,
        date VARCHAR(45) NOT NULL,
        description VARCHAR(200) NOT NULL,
        link VARCHAR(60) NOT NULL,
        flag VARCHAR(15) NULL,
        PRIMARY KEY(title)
        PRIMARY KEY(link)
);
```

Gambar 5.8 *Query* untuk membuat tabel `rssbuff`
Sumber: Implementasi

Query SQL untuk membentuk tabel `rssloker` ditunjukkan pada Gambar 5.9.

```
CREATE TABLE rssloker(
        idrssloker INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
        title VARCHAR(70) NOT NULL,
        link VARCHAR(60) NOT NULL,
        date VARCHAR(45) NOT NULL,
        description VARCHAR(200) NOT NULL,
        perusahaan VARCHAR(60) NOT NULL,
        batasakhir VARCHAR(30) NOT NULL,
        posisi VARCHAR(55) NOT NULL,
        posting VARCHAR(30) NOT NULL,
        PRIMARY KEY(idrssloker)
);
```

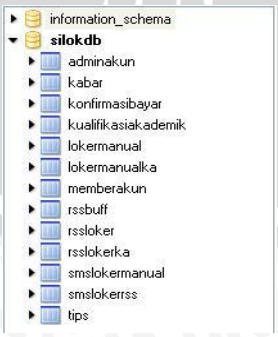
Gambar 5.9 *Query* untuk membuat tabel `rssloker`
Sumber: Implementasi

Query SQL yang digunakan dalam membentuk tabel `tips` ditunjukkan pada Gambar 5.10.

```
CREATE TABLE tips(
        idtips INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
        judul VARCHAR(100) NOT NULL,
        tglposting VARCHAR(30) NOT NULL,
        isi LONGTEXT NOT NULL,
        PRIMARY KEY(manages_id)
);
```

Gambar 5.10 *Query* untuk membuat tabel `tips`
Sumber: Implementasi

Basis data `skripsi` yang telah diimplementasikan pada DBMS MySQL Query Browser ditunjukkan dalam Gambar 5.11.



Gambar 5.11 DBMS MySQL Query Browser
Sumber: Implementasi

5.3 Implementasi Klas Pada File Program

Class-class yang telah didesain pada proses perancangan, masing-masing direalisasikan pada sebuah *file* program *.java. Tabel 5.3 menunjukkan pasangan antara *class* dengan *file* program yang digunakan untuk mengimplementasikannya.

Tabel 5.3 Implementasi klas pada file java

No.	Nama Paket	Nama *Class*	Nama *File* Program
1	com.myapp.struts.admin	AbaiRssAction	AbaiRssAction.java
2		AdminLoginAction	AdminLoginAction.java
3		AdminLogoutAction	AdminLogoutAction.java
4		BukaRssSimpanAction	BukaRssSimpanAction.java
5		HapusKabarAction	HapusKabarAction.java
6		HapusKonfirmasiBayarAction	HapusKonfirmasiBayarAction.java
7		HapusLokerManualAction	HapusLokerManualAction.java
8		HapusLokerRssAction	HapusLokerRssAction.java
9		HapusMemberAction	HapusMemberAction.java
10		HapusTipsAction	HapusTipsAction.java
11		LihatDetilKabarTipsAction	LihatDetilKabarTipsAction.java
12		LihatDetilLokerManEdit	LihatDetilLokerManEdit.java
13		LihatDetilLokerRssEdit	LihatDetilLokerRssEdit.java
14		LihatDetilMemberAction	LihatDetilMemberAction.java
15		PerubahanAdminAction	PerubahanAdminAction.java
16		SimpanArtikelAction	SimpanArtikelAction.java
17		SimpanEditMemberAction	SimpanEditMemberAction.java
18		SimpanPerubahanArtikelAction	SimpanPerubahanArtikelAction.java
19		TerimaMemberAction	TerimaMemberAction.java
20		TolakMemberAction	TolakMemberAction.java
21		UpdateDepositMemberAction	UpdateDepositMemberAction.java
22		DoPathAdmin	DoPathAdmin.java
23		KirimSmsPerMemberAction	KirimSmsPerMemberAction.java
24		LihatMemberByKaAction	LihatMemberByKaAction.java
25		PilihLokerAction	PilihLokerAction.java
26		SimpanEditLokerManualAction	SimpanEditLokerManualAction.java

No.	Nama Paket	Nama *Class*	Nama *File* Program
27	com.myapp.struts.admin	SimpanEditLokerRssAction	SimpanEditLokerRssAction.java
28		SimpanLokerManualAction	SimpanLokerManualAction.java
29		SimpanRssLokerAction	SimpanRssLokerAction.java
30		AdminLoginForm	AdminLoginForm.java
31		PilihLokerForm	PilihLokerForm.java
32		DataRss	DataRss.java
33		SimpanArtikelForm	SimpanArtikelForm.java
34		SimpanEditMemberForm	SimpanEditMemberForm.java
35		PerubahanAdminForm	PerubahanAdminForm.java
36		PengambilRss	PengambilRss.java
37		SimpanLokerManualForm	SimpanLokerManualForm.java
38		SimpanPerubahanArtikelForm	SimpanPerubahanArtikelForm.java
39		SimpanRssLokerForm	SimpanRssLokerForm.java
40		UpdateBatasAkhirMemberForm	UpdateBatasAkhirMemberForm.java
41		PengirimSmsViaSmpp	PengirimSmsViaSmpp.java
42		Deposit	Deposit.java
43		SmsLoker	SmsLoker.java
44		MemberReportBean	MemberReportBean.java
45		LokerReportBean	LokerReportBean.java
46		UangReportBean	UangReportBean.java
47	com.myapp.struts.guest	DoPath	DoPath.java
48		LihatDetilKabarAction	LihatDetilKabarAction.java
49		PendaftaranMemberAction	PendaftaranMemberAction.java
50		PendaftaranMemberForm	PendaftaranMemberForm.java
51		GuestLoginAction	GuestLoginAction.java
52		GuestLoginForm	GuestLoginForm.java
53	com.myapp.struts.member	KonfirmasiPembayaran	KonfirmasiPembayaran.java.java
54		MemberLogoutAction	MemberLogoutAction.java
55		PenyimpanPerubahanMemberAction	PenyimpanPerubahanMemberAction.java
56		StatusPembayaranAction	StatusPembayaranAction.java
57		SubmitKonfirmasiPembayaranAction	SubmitKonfirmasiPembayaranAction.java
58		PerubahanAkunMemberAction	PerubahanAkunMemberAction.java

No.	Nama Paket	Nama *Class*	Nama *File* Program
59		PenyimpanPerubahanMemberForm	PenyimpanPerubahanMemberForm.java
60		SubmitKonfirmasiPembayaranForm	SubmitKonfirmasiPembayaranForm.java
61	com.myapp.hibernate.data	AdminAkun	AdminAkun.java
62		MemberAkun	MemberAkun.java
63		KonfirmasiBayar	KonfirmasiBayar.java
64		LokerManualTersimpan	LokerManualTersimpan.java
65	com.myapp.hibernate.data	Kabar	Kabar.java
66		RssBuff	RssBuff.java
67		RssTersimpan	RssTersimpan.java
68		Tips	Tips.java
69		KualifikasiAkademik	KualifikasiAkademik.java
70		RssbuffId	RssbuffId.java
71		RssLoker	RssLoker.java
72		LokerManual	LokerManual.java
73	com.myapp.hibernate.helper	AdminakunHelper	AdminakunHelper.java
74		KabarHelper	KabarHelper.java
75		KonfirmasiBayarHelper	KonfirmasiBayarHelper.java
76		LokerManualHelper	LokerManualHelper.java
77		RssBuffHelper	RssBuffHelper.java
78		RsslokerHelper	RsslokerHelper.java
79		TipsHelper	TipsHelper.java
80		MemberakunHelper	MemberakunHelper.java
81		HibernateUtil	HibernateUtil.java
82	com.myapp.security	DesEncrypter	DesEncrypter.java

Sumber: Perancangan

5.4 Implementasi Algoritma

Pada sistem ini, terdapat beberapa algoritma proses utama (pokok) yang dilakukan di dalam system. Diantarnya adalah proses pengambilan data RSS, penambahan nilai deposit dari member dan proses pengiriman SMS terhadap member yang dipilih.

5.4.1 Implementasi Algoritma Mengambil Data RSS

Aplikasi memiliki fungsionalitas untuk mengambil data RSS dari web yang lain yang menyediakan informasi lowongan kerja. Fungsi tersebut diimplementasikan pada operasi `getRssList` yang ada di dalam klas `DataRss.java`. Algoritma unntuk mengumpulkan RSS tersebut ditunjukkan pada Algoritma 5.1

```
METHOD getRssList PARAMETER addr IS String: LinkedList<DataRss>
BEGIN

DECLARATION:
TYPE rssItemList IS LinkedList<RSSItem>
TYPE rssList IS LinkedList<DataRss>
TYPE hand IS RSSHandler
TYPE rssItem IS RSSItem
TYPE dr IS DataRss


DESCRIPTION:
    1.  CREATE OBJECT rssList AS LinkedList<DataRss>
    2.  CREATE OBJECT hand AS RSSHandler
    3.  TRY
    4.    CALL RSSParser.parseXmlFile(addr, hand, false)
    5.    rssItemList <- CALL hand.getRSSChannel().getItems()
    6.    FOR i <- 0 to rssItemList.size THEN
    7.       CREATE OBJECT rssItem AS RSSItem()
    8.       rssItem <- CALL rssItemList.get(i)
    9.       CREATE OBJECT dr AS DataRss
    10.      CALL dr.setAbout(CALL rssItem.getAboutAttribute())
    11.      CALL dr.setTitle(CALL rssItem.getTitle())
    12.      CALL dr.setLink(CALL rssItem.getLink())
    13.      CALL dr.setDate(CALL rssItem.getDate())
    14.      CALL dr.setDescription(CALL rssItem.getDescription())
    15.      CALL rssList.add(dr)
    16.   END FOR
    17. CATCH
    18.   PRINT “Error membaca file xml”
    19. END TRY

RETURN rssList
END updateDeposit
```

Algoritma 5.1 Algoritma Mengambil Data RSS
Sumber: Implementasi

Klas `RSSHandler`, `RSSItem` dan `RSSParser` merupakan klas-klas yang disediakan oleh *library* Java API (`rsslib4j`) yang menggunakan *SAX XML parsing*. Atribut dari RSS yang didapat dari proses pembacaan dengan menggunakan library ini masih global. Untuk menyesuaikan dengan kebutuhan aplikasi, dibuat klas `DataRss` yang merupakan klas dimana di dalamnya terdapat atribut dari RSS yang dibutuhkan oleh aplikasi saja.

5.4.2 Implementasi Algoritma *Update* Deposit Member

Perhitungan biaya layanan SMS ini dilakukan dengan menghitung setiap pengiriman SMS. Apabila terjadi pengiriman SMS terhadap member, maka deposit dari member yang bersangkutan akan berkurang. Jika member melakukan pembayaran, maka nilai deposit akan bertambah. Proses penambahan nilai deposit keika member melakukan pembayaran ditunjukkan pada Algoritma 5.2. Algoritma ini diimplementasikan pada fungsi `updateDeposit` dalam klas `Deposit.java`.

```
METHOD updateDeposit :boolean
BEGIN

DECLARATION:
TYPE idBayar IS int
TYPE namaMember IS String
TYPE tglBayar IS String
TYPE jumBayar IS double
TYPE mh IS MemberakunHelper
TYPE poin IS int
TYPE statusUpdateDeposit IS boolean
TYPE kh IS KonfirmasibayarHelper
TYPE statusUpdateKonfirmasiBayar IS Boolean
TYPE ret IS boolean

DESCRIPTION:
    CREATE OBJECT mh AS MemberakunHelper
    1.  poin <- this.jumBayar / 1000
    2.  poin <- poin + CALL mh.ambilDeposit(this.namaMember)
    3.  CREATE OBJECT mh AS MemberakunHelper
    4.  statusUpdateDeposit <- CALL mh.updateDeposit(this.namaMember, poin)
    5.  CREATE OBJECT mh AS MemberakunHelper
    6.  CALL mh.updateTglBayarItemByUname(this.namaMember, this.tglBayar)
    7.  IF statusUpdateDeposit THEN
    8.    CREATE OBJECT kh AS KonfirmasibayarHelper
    9.    statusUpdateKonfirmasiBayar <- CALL
        kh.updateStatusItemById(this.idBayar)
    10.   IF statusUpdateKonfirmasiBayar THEN
    11.     ret <- true
    12.   ELSE
    13.     ret <- false
    14.   END IF
    15. ELSE
    16.   ret <- false
    17. END IF


RETURN ret
END updateDeposit
```

Algoritma 5.2 Algoritma *Update* Deposit Member
Sumber: Implementasi

5.4.3 Implementasi Algoritma Mengirim SMS Per Member

Algoritma proses pengiriman SMS terhadap suatu member, ditunjukkan pada Algoritma 5.3.

```
METHOD kirimSmsPerMember PARAMETER idLoker IS String, jenisLoker IS String, member
IS List<String> : boolean
BEGIN
DECLARATION:
TYPE ret IS Boolean
TYPE isiReport IS String
TYPE memberSukses IS String
TYPE memberGagal IS String
TYPE lnkLoker IS String
TYPE idLok IS int
TYPE p IS PengirimSmsViaSmpp
TYPE namaMember IS String
TYPE mh IS MemberakunHelper
TYPE m IS Memberakun
TYPE pesan IS String
TYPE noHp IS String
TYPE rh IS RsslokerHelper
TYPE rssLoker IS RssTersimpan
DESCRIPTION:
1.   ret <- false
2.   memberSukses  <- “<ul>”
3.   memberGagal <- “<ul>”
4.   lnkLoker <- “”
5.   idLok <- CALL Integer.ParseInt(idLoker)
6.   CREATE OBJECT p AS PengirimSmsViaSmpp
7.   IF (CALL p.bukaSession()) THEN
8.      FOR i <- 0 to i<member.size THEN
9.         namaMember <- CALL member.get(i)
10.        CREATE OBJECT mh AS MemberakunHelper
11.        m <- CALL mh.ambilMember(namaMember)
12.        noHp <- CALL m.getNohp()
13.        CREATE OBJECT rh AS RsslokerHelper
14.        rssLoker <- CALL rh.getLokerRssById(idLok)
15.        pesan <- CALL rh.getTitle+CALL rh.getLink+CALL rh.getPosisi+CALL
           rh.getPerusahaan+CALL rh.getBatasAkhir
16.        IF jenisLoker=”rss” THEN
17.           IF (CALL p.kirimSms(noHp,pesan)) THEN
18.             CREATE OBJECT mh AS MemberakunHelper
19.             IF (CALL mh.setRsslokers(namaMember,idLok)) THEN
20.                memberSukses <- memberSukses+ "<li><a
                   href=\"lihatDetilMember.do?hal=3&u="+namaMember+"&lnk=subhal=3\">"
                   +namaMember+"</a></li>"
21.             ELSE
22.                memberSukses <- memberSukses+ "<li><a
    href=\"lihatDetilMember.do?hal=3&u="+namaMember+"&lnk=subhal=3\">"+namaMember+"<
    /a>(Gagal update database)</li>";
23.             END IF
24.          ELSE
25.            memberGagal <- memberGagal+ "<li><a
    href=\"lihatDetilMember.do?hal=3&u="+namaMember+"&lnk=subhal=3\">"+namaMember+"<
    /a></li>"
26.          END IF
27.       ELSE
28.             IF (CALL p.kirimSms(noHp,pesan)) THEN
29.             CREATE OBJECT mh AS MemberakunHelper
30.             IF (CALL mh.setLokermanuals(namaMember,idLok)) THEN
31.                memberSukses <- memberSukses+ "<li><a
    href=\"lihatDetilMember.do?hal=3&u="+namaMember+"&lnk=subhal=3\">"+namaMember+"<
    /a></li>"
32.             ELSE
33.                memberSukses <- memberSukses+ "<li><a
    href=\"lihatDetilMember.do?hal=3&u="+namaMember+"&lnk=subhal=3\">"+namaMember+"<
    /a>(Gagal update database)</li>";
34.             END IF
35.          ELSE
```

```
36.            memberGagal <- memberGagal+ "<li><a
                href=\"lihatDetilMember.do?hal=3&u="+namaMember+"&lnk=subhal=3\">
                "+namaMember+"</a></li>"
37.          END IF
38.        END IF
39.     END FOR
40.  ELSE
41.     isiReport <- "Gagal terhubung ke SMPP server";
42.     ret <- false;
43.  END IF
RETURN ret
END kirimSmsPerMember
```

Algoritma 5.3 Algoritma Mengirim SMS Per Member
Sumber: Implementasi

5.4.4 Implementasi Algoritma Membuka *Session* Koneksi ke SMPP Server Simulator

Untuk mengirimkan pesan ke server SMPP, yang pertama dilakukan adalah membuka *session* koneksi dengan server SMPP. Algoritma untuk membuka koneksi dengan server SMPP diberikan pada fungsi **`bukaSession ()`** yang ada pada klas `PengirimSmsViaSmpp.java`. Algoritma untuk membuka koneksi ke server SMPP ditunjukkan pada Algoritma 5.4

```
METHOD bukaSession PARAMETER host IS STRING, port IS INTEGER :BOOLEAN
BEGIN

DECLARATION:
TYPE session IS SMPPSession
TYPE ret IS BOOLEAN

DESCRIPTION:
    1.  CREATE OBJECT session AS SMPPSession
    2.  ret <- false
    3.  TRY
    4.       CALL session.connectAndBind(host, port)
    5.       ret <- true
    6.  CATCH
    7.       ret <- false
    8.  END TRY

RETURN ret
END bukaSession
```

Algoritma 5.4 Algoritma Mengambil Data RSS
Sumber: Implementasi

5.5 Implementasi Antarmuka Aplikasi

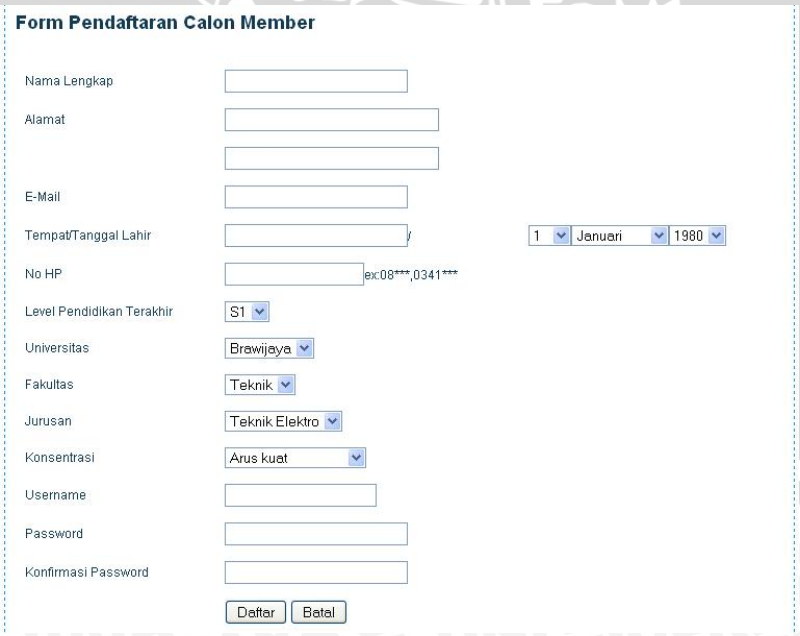
Pada aplikasi ini, antarmuka dibuat sesuai dengan kebutuhan fungsional yang harus disediakan oleh sistem. Tujuan utama pengembangan antarmuka adalah untuk memberikan kemudahan bagi pengguna (*user*) dalam menggunakan aplikasi yang telah dibangun. Implementasi antarmuka aplikasi ini terdiri atas implementasi antarmuka Guest, Member, dan Administrator.

5.5.1 Implementasi Antarmuka Guest

Guest adalah pengguna yang belum terautentifikasi sebagai Member atau Administrator. Fasilitas yang dapat diakses Guest antara lain registrasi, *login* sebagai Member, dan *login* sebagai Administrator.

5.5.1.1 Implementasi Antarmuka Registrasi

Antarmuka ini akan ditampilkan apabila pengguna ingin menjadi member dan ingin berlangganan informasi lowongan kerja melalui *handphone*. Pengguna harus mengisi beberapa *field* antara lain Nama Lengkap, Alamat, Email, Tempat/Tanggal Lahir, No.HP, Level Pendidikan Terakhir, Universitas, Fakultas, Jurusan, Konsentrasi, Username, Password, Konfirmasi Password. Jika pengguna mengisi *field-field* tersebut dengan benar dan memenuhi persyaratan sebagai member maka akan dibuatkan akun baru untuk pengguna.



Gambar 5.12 Antarmuka Registrasi
Sumber: Implementasi

Gambar 5.12 merupakan gambar hasil implementasi antarmuka untuk registrasi. Penjelasan masing-masing bagian dari antarmuka ini diuraikan pada Tabel 5.4.

Tabel 5.4 Penjelasan tombol-tombol pada antarmuka Registrasi

No	Label Tombol	Penjelasan
1	Daftar	Tombol ini digunakan untuk melakukan proses pembuatan akun baru yang nantinya akan di *approve* oleh Administrator. Sebelum menekan tombol ini, pengguna harus mengisi *field-field* dengan benar.
2	Batal	Tombol ini digunakan untuk membatalkan proses pendaftaran

Sumber: Implementasi

5.5.1.2 Implementasi Antarmuka Login Member

Antarmuka *login* ini akan ditampilkan ketika pengguna ingin menggunakan aplikasi sebagai Member. Antarmuka ini mempunyai dua buah *field* yaitu Username dan Password.

Gambar 5.13 Antarmuka *Login* Member

Sumber: Implementasi

Gambar 5.13 merupakan gambar hasil implementasi antarmuka untuk *login* Member. Penjelasan masing-masing bagian dari antarmuka ini diuraikan pada Tabel 5.5.

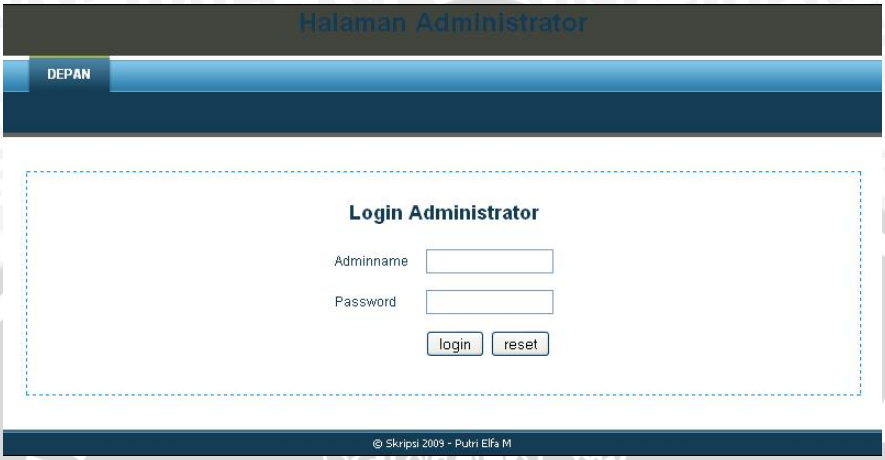
Tabel 5.5 Penjelasan tombol-tombol pada antarmuka *Login* Member

No	Label Tombol	Penjelasan
1	Login	Tombol ini digunakan untuk melakukan autentifikasi pengguna. Jika pengguna mengisi *field-field* dengan benar maka akan ditampilkan antarmuka utama Member.

Sumber: Implementasi

5.5.1.3 Implementasi Antarmuka Login Administrator

Antarmuka *login* ini akan ditampilkan ketika pengguna ingin menggunakan aplikasi sebagai Administrator. Antarmuka ini mempunyai dua buah *field* yaitu Adminname dan Password.



Gambar 5.14 Antarmuka *Login* Administrator

Sumber: Implementasi

Gambar 5.14 merupakan gambar hasil implementasi antarmuka untuk *login.* Penjelasan masing-masing bagian dari antarmuka ini diuraikan pada Tabel 5.6.

Tabel 5.6 Penjelasan tombol-tombol pada antarmuka Login Administrator

No	Label Tombol	Penjelasan
1	Login	Tombol ini digunakan untuk melakukan autentifikasi pengguna. Jika pengguna mengisi *field-field* dengan benar maka akan ditampilkan antarmuka halaman utama Administrator.
2.	Reset	Tombol ini digunakan untuk mengembalikan nilai *field*-field pada *form* seperti semula.

Sumber: Implementasi

5.5.2 Implementasi Antarmuka Member

Jika username dan password yang diisi oleh pengguna di antarmuka Login Member sesuai dengan yang ada pada database, maka pengguna dapat menggunakan fasilitas aplikasi sebagai Member. Antarmuka aplikasi yang dapat diakses Member antara lain Ubah Akun, Status Pembayaran, dan Konfirmasi Pembayaran

Gambar 5.15 merupakan gambar hasil implementasi antarmuka halaman utama member. Penjelasan masing-masing bagian dari antarmuka ini diuraikan pada Tabel 5.7.

Gambar 5.15 Antarmuka Halaman Utama Member

Sumber: Implementasi

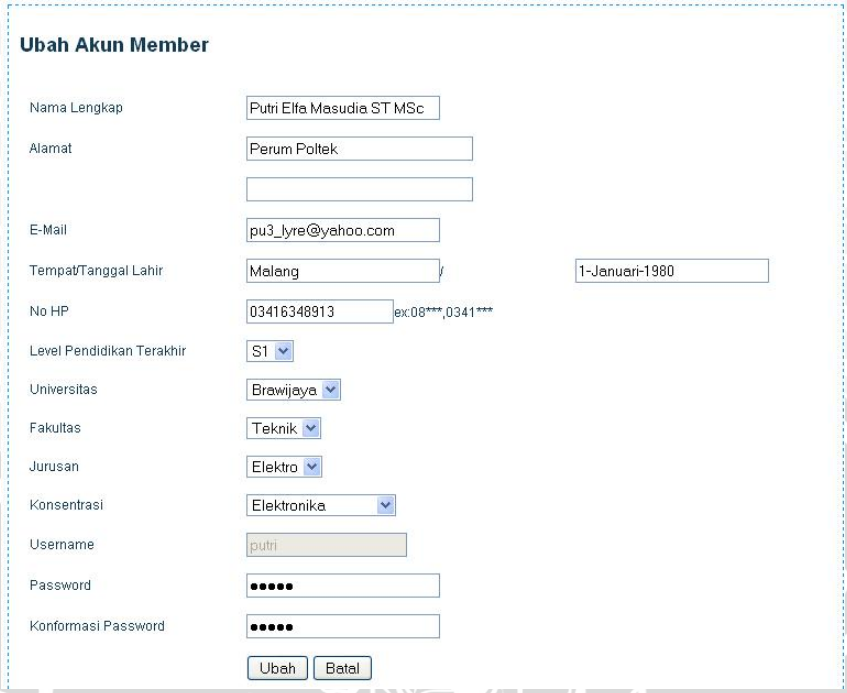
Tabel 5.7 Penjelasan *link* pada antarmuka Ubah Akun

No	Label	Penjelasan
1	Ubah Akun	*Link* ini digunakan untuk melakukan proses pengubahan akun. Sebelum menekan tombol ini, pengguna harus mengisi *field-field* dengan benar.
2	Status Pembayaran	*Link* ini digunakan untuk melihat sisa deposit Member
3	Konfirmasi Pembayaran	*Link* ini digunakan untuk melakukan konfirmasi pembayaran setelah member membayar melalui bank
4	Logout	*Link* ini digunakan pengguna untuk keluar dari antarmuka Member.

Sumber: Implementasi

5.5.2.1 Implementasi Antarmuka Ubah Akun Member

Antarmuka ini akan ditampilkan ketika pengguna ingin mengubah akunnya. *Field-field* yang terdapat pada antarmuka ini antara lain Nama Lengkap, Alamat, Email, Tempat/Tanggal Lahir, No.HP, Level Pendidikan Terakhir, Universitas, Fakultas, Jurusan, Konsentrasi, Username, Password, Konfirmasi Password.



Gambar 5.16 Antarmuka Ubah Akun Member

Sumber: Implementasi

Gambar 5.16 merupakan gambar hasil implementasi antarmuka untuk mengubah akun Member. Penjelasan masing-masing bagian dari antarmuka ini diuraikan pada Tabel 5.8.

Tabel 5.8 Penjelasan tombol-tombol pada antarmuka Ubah Akun

No	Label Tombol	Penjelasan
1	Ubah	Tombol ini digunakan untuk melakukan proses pengubahan akun. Sebelum menekan tombol ini, pengguna harus mengisi *field-field* dengan benar.
2	Batal	Tombol ini digunakan untuk membatalkan proses pengubahan akun.

Sumber: Implementasi

5.5.2.2 Implementasi Antarmuka Status Pembayaran

Antarmuka ini akan ditampilkan jika pengguna ingin melihat status pembayaran member dengan melihat sisa deposit. Pada antarmuka ini ditampilkan tanggal member daftar, tanggal member terakhir bayar dan sisa deposit. Implementasi Antarmuka Status Pembayaran ditunjukkan dalam Gambar 5.17.

Gambar 5.17 Antarmuka Status Pembayaran

Sumber: Implementasi

5.5.2.3 Implementasi Antarmuka Konfirmasi Pembayaran

Antarmuka ini akan ditampilkan jika pengguna ingin melakukan konfirmasi pembayaran setelah melakukan pembayaran di bank. *Field-field* yang terdapat pada antarmuka ini antara lain Tanggal Bayar, Jumlah Bayar, No. Rekening Pengirim.

Gambar 5.18 Antarmuka Konfirmasi Pembayaran

Sumber: Implementasi

Gambar 5.18 merupakan gambar hasil implementasi antarmuka untuk konfirmasi pembayaran member. Penjelasan masing-masing bagian dari antarmuka ini diuraikan pada Tabel 5.9.

Tabel 5.9 Penjelasan tombol-tombol pada antarmuka Konfirmasi Pembayaran

No	Label Tombol	Penjelasan
1	Ok	Tombol ini digunakan untuk melakukan proses pengubahan akun. Sebelum menekan tombol ini, pengguna harus mengisi *field-field* dengan benar.

Sumber: Implementasi

5.5.3 Implementasi Antarmuka Administrator

Jika pengguna mengisi *field-field* antarmuka Admin Login dengan benar maka pengguna dapat menggunakan fasilitas aplikasi sebagai Administrator. Antarmuka aplikasi yang dapat diakses Administrator antara lain Ubah Akun, Member, Loker, Kirim SMS, Report, Artikel.

Gambar 5.19 Antarmuka Halaman Utama Administrator

Sumber: Implementasi

Gambar 5.19 merupakan gambar hasil implementasi antarmuka halaman utama Administrator. Penjelasan masing-masing bagian dari antarmuka ini diuraikan pada Tabel 5.10.

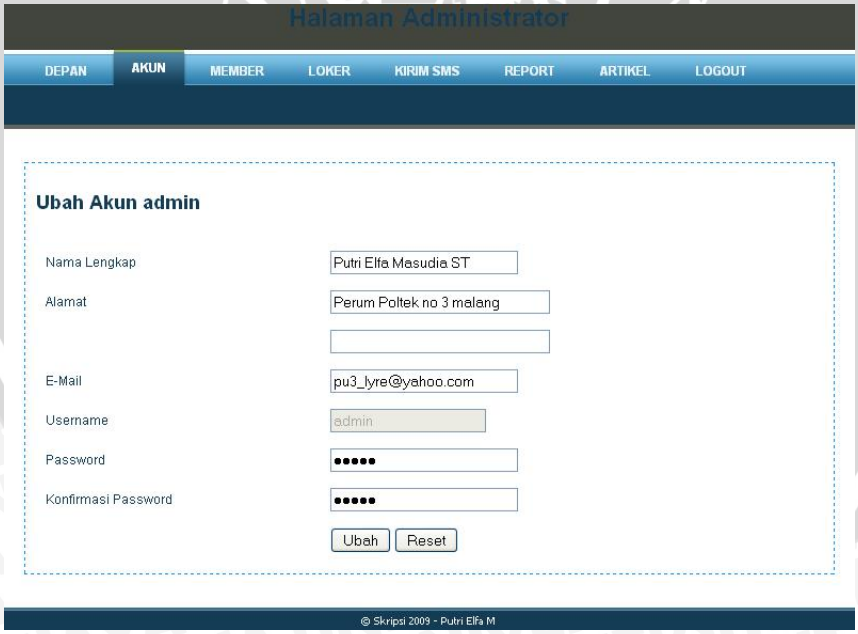
Tabel 5.10 Penjelasan tombol-tombol pada antarmuka halaman utama Administrator

No	Label Menu	Penjelasan
1	Akun	Menu ini digunakan untuk mengubah akun Administrator.
2	Member	Menu ini digunakan mengatur data Member.
3	Loker	Menu ini digunakan untuk mengatur data informasi lowongan kerja sebelum dimasukkan ke database dan ditampilkan di halaman Member atau Guest
4	Kirim SMS	Menu ini digunakan untuk mengirimkan informasi lowongan kerja ke *handphone* member
5	Report	Menu ini digunakan untuk mencetak data Member, Loker dan Keuangan per periode
6	Artikel	Menu ini digunakan untuk memasukkan artikel secara manual dan menampilkanannya ke halaman Member atau Guest
7	logout	Menu ini digunakan pengguna untuk keluar dari antarmuka Administrator.

Sumber: Implementasi

5.5.3.1 Implementasi Antarmuka Ubah Akun Administrator

Antarmuka ini akan ditampilkan jika Administrator ingin mengubah akunnya. *Field-field* yang terdapat pada antarmuka ini antara lain Nama Lengkap, Alamat, Email, Username, Password, Konfirmasi Password.



Gambar 5.20 Antarmuka Ubah Akun Administrator

Sumber: Implementasi

Gambar 5.20. merupakan gambar hasil implementasi antarmuka untuk mengubah akun Administrator. Penjelasan masing-masing bagian dari antarmuka ini diuraikan pada Tabel 5.11.

Tabel 5.11 Penjelasan tombol-tombol pada antarmuka Ubah Akun Administrator

No	Label Menu	Penjelasan
1	Ubah	Tombol ini digunakan untuk mengubah akun Administrator.
2	Reset	Tombol ini digunakan untuk mengembalikan nilai *field*-field pada *form* seperti semula.

Sumber: Implementasi

5.5.3.2 Implementasi Antarmuka Manajemen Member

Antarmuka Manajemen Member menyediakan beberapa menu antara lain menu untuk Konfirmasi Pembayaran Member, Lihat Tanggungan Member, Data Calon Member, dan Data Member. Pada antarmuka ini, Administrator dapat melakukan proses manajemen Member antara lain mengkonfirmasi pembayaran member, penambahan, pengubahan, dan penghapusan Member.

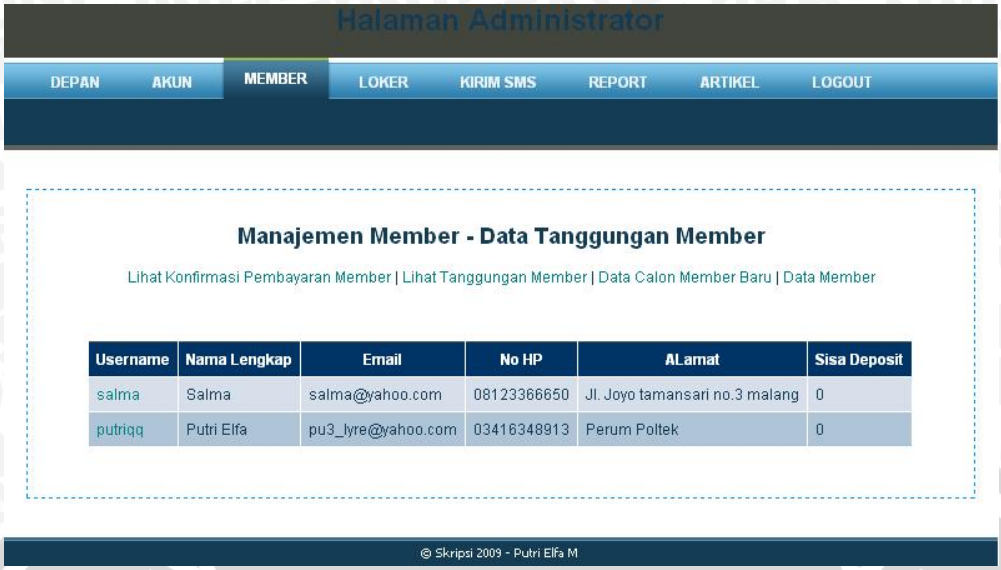
Gambar 5.21 Antarmuka Manajemen Member-Konfirmasi Pembayaran
Sumber: Implementasi

Gambar 5.21. merupakan gambar hasil implementasi antarmuka konfirmasi pembayaran member. Antarmuka ini digunakan untuk mengkonfirmasi pembayaran Member. Penjelasan masing-masing bagian dari antarmuka ini diuraikan pada Tabel 5.12.

Tabel 5.12 Penjelasan *link* pada antarmuka Konfirmasi Pembayaran Member

No	Label	Penjelasan
1	Konfirmasi	*Link* ini digunakan untuk mengkonfirmasi pembayaran Member
2	Hapus	*Link* ini digunakan untuk melakukan proses penghapusan konfirmasi pembayaran Member

Sumber: Implementasi



Gambar 5.22 Antarmuka Manajemen Member-Lihat Tanggungan Member

Sumber: Implementasi

Gambar 5.22. merupakan gambar hasil implementasi antarmuka lihat tanggungan member. Antarmuka ini digunakan untuk melihat siapa saja Member yang belum membayar. Penjelasan masing-masing bagian dari antarmuka ini diuraikan pada Tabel 5.13.

Tabel 5.13 Penjelasan *link* pada antarmuka Lihat Tanggungan Member

No	Label	Penjelasan
1	Username	*Link* ini digunakan untuk melihat profil Member secara detail

Sumber: Implementasi

Gambar 5.23 Antarmuka Manajemen Member-Data Calon Member

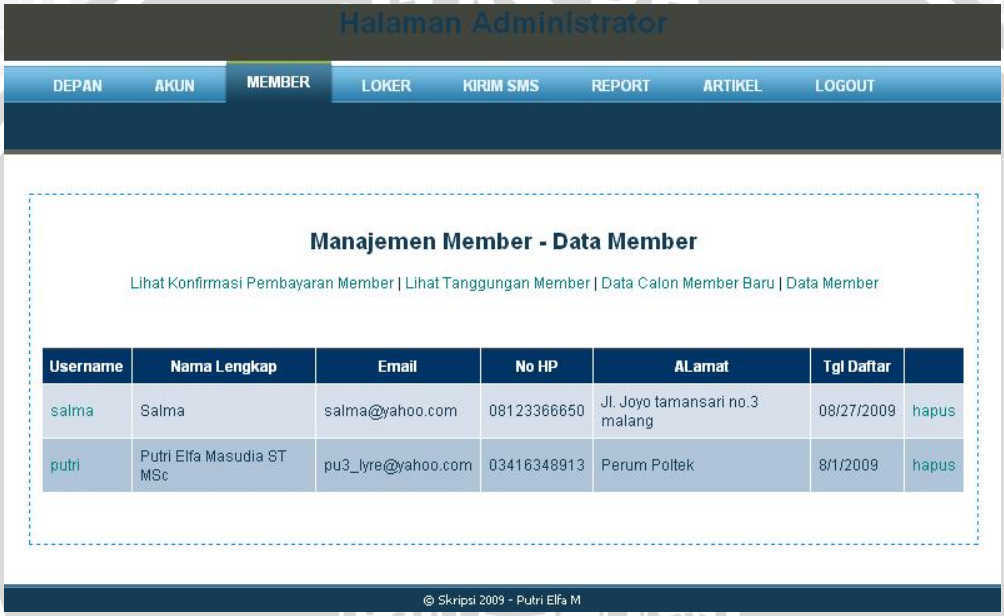
Sumber: Implementasi

Gambar 5.23. merupakan gambar hasil implementasi antarmuka data calon member. Antarmuka ini digunakan untuk melihat dan menyeleksi data calon member yang akan di *approve*. Penjelasan masing-masing bagian dari antarmuka ini diuraikan pada Tabel 514.

Tabel 5.14 Penjelasan *link* pada antarmuka Data Calon Member

No	Label	Penjelasan
1	Username	*Link* ini digunakan untuk melihat profil Member secara detail
2	Terima	*Link* ini digunakan untuk menerima calon Member menjadi Member
3	Tolak	*Link* ini digunakan untuk menolak calon Member

Sumber: Implementasi



Gambar 5.24 Antarmuka Manajemen Member-Data Member
Sumber: Implementasi

Gambar 5.24. merupakan gambar hasil implementasi antarmuka data member. Antarmuka ini digunakan untuk menampilkan data member. Penjelasan masing-masing bagian dari antarmuka ini diuraikan pada Tabel 5.15.

Tabel 5.15 Penjelasan *Link* pada antarmuka Data Member

No	Label	Penjelasan
1	username	*Link* ini digunakan untuk melihat profil Member secara detail
2	Hapus	*Link* ini digunakan untuk melakukan proses penghapusan data Member

Sumber: Implementasi

5.5.3.3 Implementasi Antarmuka Manajemen Loker

Antarmuka ini akan ditampilkan jika Administrator ingin mengatur data informasi lowongan kerja. Antarmuka ini terdiri dari beberapa menu yaitu Data

RSS Loker Baru, Data RSS Loker Tersimpan, Input Loker Manual, dan Hasil Input Loker Manual.

Gambar 5.25 Antarmuka Manajemen Loker-Rss Loker Baru
Sumber: Implementasi

Gambar 5.25. merupakan gambar hasil implementasi antarmuka untuk rss loker baru. Penjelasan masing-masing bagian dari antarmuka ini diuraikan pada Tabel 5.16.

Tabel 5.16 Penjelasan *Link* pada antarmuka Data RSS Loker Baru

No	Label	Penjelasan
1	Judul	*Link* ini digunakan untuk menuju *URL* website penyedia Rss
2	Simpan	*Link* ini digunakan untuk menyeleksi data Rss baru yang akan dimasukkan ke data Rss loker tersimpan.
3	Abaikan	*Link* ini digunakan untuk mengabaikan data Rss baru

Sumber: Implementasi

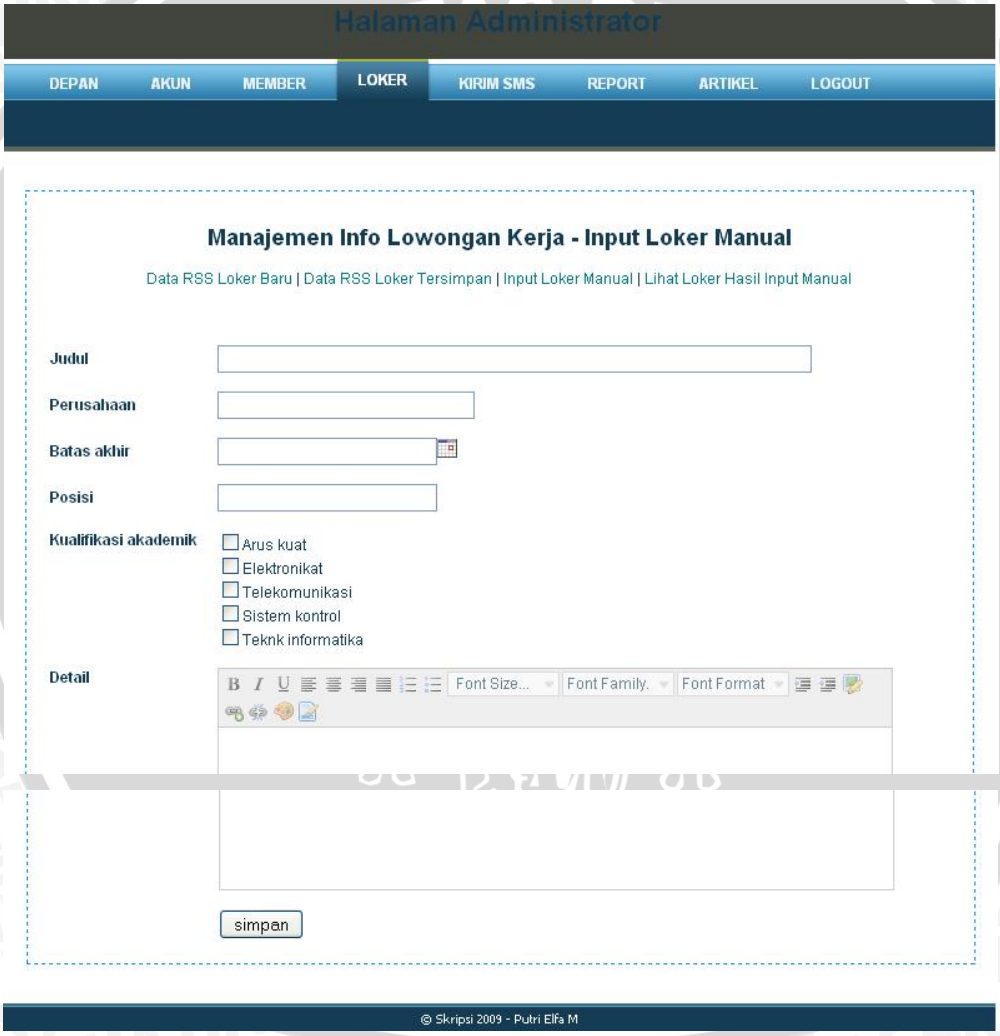
Gambar 5.26 Antarmuka Manajemen Loker-Rss Loker Tersimpan
Sumber: Implementasi

Gambar 5.27. merupakan gambar hasil implementasi antarmuka untuk data rss loker tersimpan. Penjelasan masing-masing bagian dari antarmuka ini diuraikan pada Tabel 5.17.

Tabel 5.17 Penjelasan *Link* pada antarmuka Data RSS Loker Tersimpan

No	Label	Penjelasan
1	Judul	*Link* ini digunakan untuk menuju *URL* website penyedia Rss
2	Edit	*Link* ini digunakan untuk melakukan proses pengeditan informasi lowongan kerja secara manual
3	Hapus	*Link* ini digunakan untuk melakukan proses penghapusan data

Sumber: Implementasi



Gambar 5.27 Antarmuka Manajemen Loker-Input Loker Manual

Sumber: Implementasi

Gambar 5.27. merupakan gambar hasil implementasi antarmuka untuk input loker manual. Penjelasan masing-masing bagian dari antarmuka ini diuraikan pada Tabel 5.18.

Tabel 5.18 Penjelasan Tombol pada antarmuka Input Loker Manual

No	Label	Penjelasan
1	Simpan	*Link* ini digunakan untuk menyimpan informasi lowongan kerja yang dimasukkan secara manual oleh Administrator

Sumber: Implementasi

Gambar 5.28 Antarmuka Manajemen Loker-Input Loker Manual
Sumber: Implementasi

Gambar 5.28. merupakan gambar hasil implementasi antarmuka untuk manajemen hasil input loker manual. Penjelasan masing-masing bagian dari antarmuka ini diuraikan pada Tabel 5.19.

Tabel 5.19 Penjelasan *Link* pada antarmuka Hasil Input Loker Manual

No	Label	Penjelasan
1	Judul	*Link* ini digunakan untuk melihat informasi lowongan kerja secara detail
2	Edit	*Link* ini digunakan untuk melakukan proses pengeditan informasi lowongan kerja
3	Hapus	*Link* ini digunakan untuk melakukan proses penghapusan data

Sumber: Implementasi

5.5.3.4 Implementasi Antarmuka Kirim SMS

Antarmuka ini akan ditampilkan jika Administrator ingin mengirimkan informasi lowongan kerja kepada Member. Pada antarmuka ini terdapat tombol pencarian Informasi lowongan kerja berdasarkan kualifikasi akademik dan jenis loker

Gambar 5.29 Antarmuka Kirim SMS

Sumber: Implementasi

Gambar 5.29. merupakan gambar hasil implementasi antarmuka untuk Kirim SMS. Penjelasan masing-masing bagian dari antarmuka ini diuraikan pada Tabel 5.20.

Tabel 5.20 Penjelasan tombol-tombol pada antarmuka Kirim SMS

No	Label	Penjelasan
1	Cari	Tombol ini digunakan untuk mencari informasi lowongan kerja berdasarkan kualifikasi akademik dan jenis loker, yang kemudian data tersebut akan ditampilkan berdasarkan jenis loker

Sumber: Implementasi

5.5.3.5 Implementasi Antarmuka Report

Antarmuka ini akan ditampilkan jika Administrator ingin mencetak laporan per periode. Pada antarmuka ini terdapat tombol pencetakan jenis laporan per periode.

Gambar 5.30 Antarmuka Report

Sumber: Implementasi

Gambar 5.30. merupakan gambar hasil implementasi antarmuka untuk Report. Penjelasan masing-masing bagian dari antarmuka ini diuraikan pada Tabel 5.21.

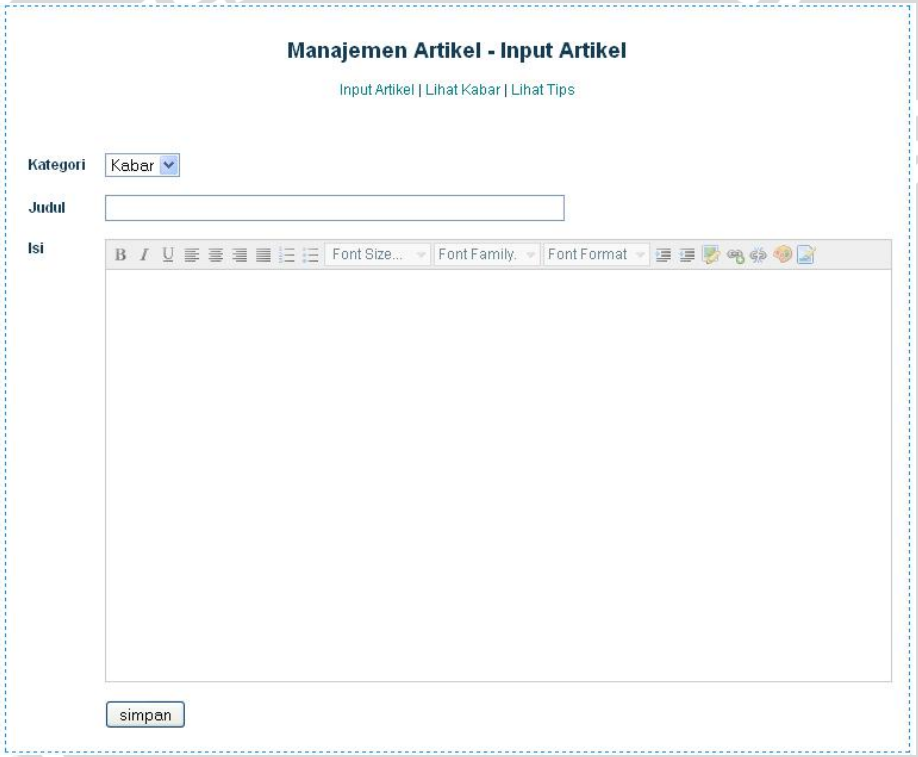
Tabel 5.21 Penjelasan tombol-tombol pada antarmuka Kirim SMS

No	Label	Penjelasan
1	Cetak	Tombol ini digunakan untuk mencetak laporan berdasarkan periode

Sumber: Implementasi

5.5.3.6 Implementasi Antarmuka Artikel

Antarmuka ini akan ditampilkan jika Administrator ingin memasukkan artikel secara manual yang akan ditampilkan pada Member atau Guest. *Field-field* yang terdapat pada antarmuka ini antara lain Kategori, Judul, Isi



Gambar 5.31 Antarmuka Artikel
Sumber: Implementasi

Gambar 5.31. merupakan gambar hasil implementasi antarmuka artikel. Penjelasan masing-masing bagian dari antarmuka ini diuraikan pada Tabel 5.22.

Tabel 5.22 Penjelasan tombol-tombol pada antarmuka Artikel

No	Label	Penjelasan
1	Simpan	*Link* ini digunakan untuk menyimpan artikel yang telah dimasukkan secara manual oleh Administrator

Sumber: Implementasi

BAB VI
PENGUJIAN DAN ANALISIS

Bab ini membahas proses pengujian dan analisis terhadap sistem yang telah dibangun. Dalam proses pengujian digunakan lima buah strategi pengujian, yaitu pengujian *E-R Diagram*, pengujian koneksi *database*, pengujian unit, pengujian integrasi, dan pengujian validasi. Pada pengujian unit dan pengujian integrasi, akan digunakan teknik pengujian *white box (white box testing)*, sedangkan pada pengujian validasi akan digunakan teknik pengujian *black box (black box testing)*. Proses analisis dilakukan untuk mengetahui unjuk kerja dari sistem perangkat lunak yang sedang dibangun.

6.1 Pengujian

Proses pengujian dilakukan melalui lima tahapan (strategi) yaitu pengujian *E-R Diagram*, pengujian koneksi *database*, pengujian unit, pengujian integrasi, dan pengujian validasi.

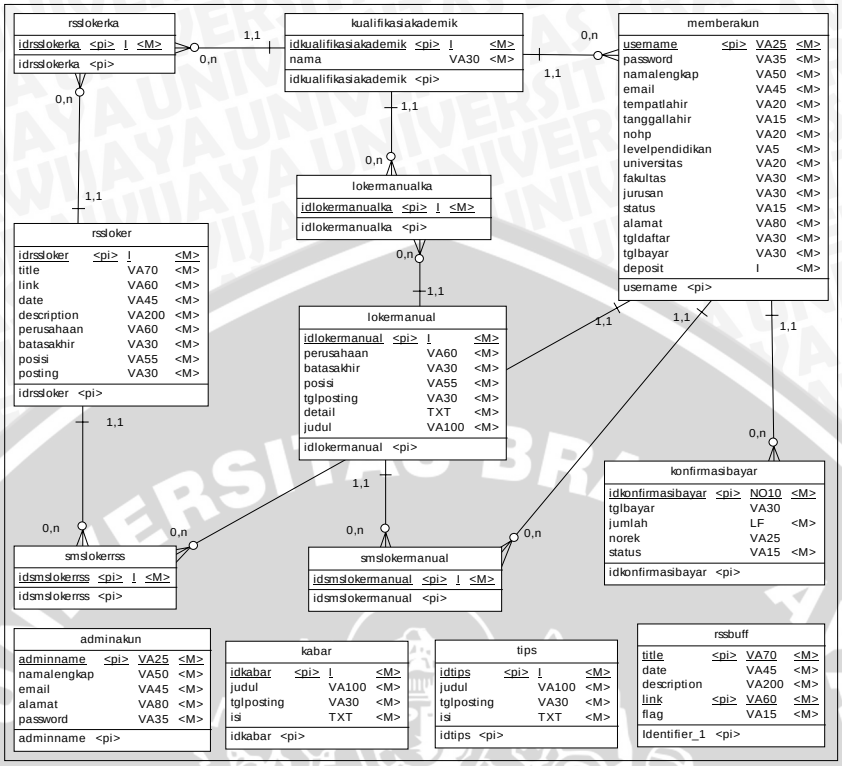
6.1.1 Pengujian *E-R Diagram*

Pengujian ini meliputi pembuatan tabel `memberakun`, `kualifikasiakademik`, `rssloker`, `lokermanual`, `rsslokerka`, `lokermanualka`, `konfirmasibayar`, `smslokerrss`, `smslokermanual`, `adminakun`, `kabar`, `tips` dan `rssbuff` pada basis data `silokdbtester` menggunakan *software* Sybase PowerDesigner 10.

Nama Kasus Uji	:	Kasus Uji *E-R Diagram*
Tujuan Pengujian	:	Pengujian dilakukan untuk memastikan bahwa tabel pada basis data `silokdb` hasil perancangan sesuai dengan tabel hasil implementasi.
Spesifikasi *Hardware*	:	▪ Prosesor : Intel PentiumIII Processor 731MHz ▪ Memori (RAM) : 256 MB ▪ Hardisk : Maxtor 2F0L0F0, kapasitas 30 GB ▪ VGA Card : NVIDIA Vanta
Spesifikasi *Software*	:	▪ Windows XP Profesional version 2002 Service Pack 1 ▪ MySQL 5.0.24a-community-nt ▪ SQL *Shell*

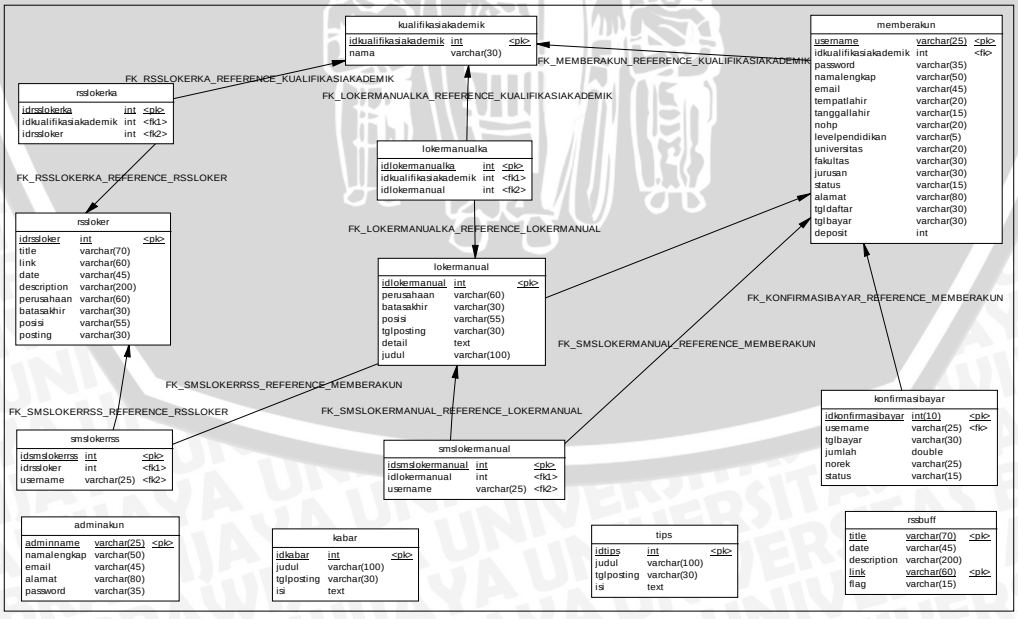
Prosedur Uji :

1. Sebuah *window* Command Prompt dijalankan dari:
   `Start | Run... | Open: cmd.exe`
2. *Server* basis data MySQL dijalankan sebagai *service* dengan memberikan perintah:
   `C:\Documents and Settings\putri>net start mysql`
3. Memasuki SQL *Shell* dengan perintah berikut:
   `C:\Program Files\MySQL\MySQL Server 5.0\bin>mysql –u root -p`
4. Tabel-tabel yang terdapat pada basis data `silokdbtester` ditampilkan dengan menggunakan perintah SQL berikut:
   `mysql>show tables;`
5. Membuka *software* Sybase PowerDesigner 10 dengan cara berikut:
   `Start | All Programs | Sybase | PowerDesigner 10 | PowerDesigner`
6. Menggambarkan hasil normalisasi dari diagram ER pada area kerja *Conceptual Data Model* (CDM).
7. Memeriksa diagram ER tersebut dengan cara menekan tombol *Check Model* pada *toolbar*. Hasil pemeriksaan ini disebut dengan CDM *Object* yang ditunjukkan dalam Gambar 6.1.



Gambar 6.1 *Conceptual Data Model Object*
Sumber: Pengujian

8. Untuk mengubah diagram ER dari CDM *Object* menjadi *Physical Data Model* (PDM) *Object*, dilakukan proses *generate* dengan menu `Tools | Generate Physical Data Model`. PDM *Object* ditunjukkan dalam Gambar 6.2.



Gambar 6.2 *Physical Data Model*
Sumber: Pengujian

9. Setelah berubah menjadi PDM *Object*, dilakukan *generate* ke *database* MySQL dengan menu `Tools | Generate Database`.

Hasil yang diharapkan : Tabel pada basis data `silokdbtester` hasil *generate* dari rancangan dengan menggunakan PowerDesigner 10 sesuai dengan tabel hasil implementasi pada basis data `silokdb`.

Hasil Pengujian : Tabel pada basis data `silokdbtester` hasil perancangan sesuai dengan tabel pada basis data `silokdb` yang telah diimplementasikan.

```
Database Generation
Generation: Check model starting...
Generation: Check model successful.
Sorting objects...
Sort completed.
Script Generation...
Dropping Indexes...
-> Index: RELATIONSHIP_10_FK (RELATIONSHIP_10_FK)
-> Index: RELATIONSHIP_3_FK (RELATIONSHIP_3_FK)
-> Index: RELATIONSHIP_4_FK (RELATIONSHIP_4_FK)
-> Index: RELATIONSHIP_5_FK (RELATIONSHIP_5_FK)
-> Index: RELATIONSHIP_1_FK (RELATIONSHIP_1_FK)
-> Index: RELATIONSHIP_2_FK (RELATIONSHIP_2_FK)
-> Index: RELATIONSHIP_8_FK (RELATIONSHIP_8_FK)
-> Index: RELATIONSHIP_9_FK (RELATIONSHIP_9_FK)
-> Index: RELATIONSHIP_6_FK (RELATIONSHIP_6_FK)
-> Index: RELATIONSHIP_7_FK (RELATIONSHIP_7_FK)
Dropping Tables...
-> Table: adminakun (ADMINAKUN)
-> Table: kabar (KABAR)
-> Table: konfirmasibayar (KONFIRMASIBAYAR)
-> Table: kualifikasiakademik (KUALIFIKASIAKADEMIK)
-> Table: lokermanual (LOKERMANUAL)
-> Table: lokermanualka (LOKERMANUALKA)
-> Table: memberakun (MEMBERAKUN)
-> Table: rssbuff (RSSBUFF)
-> Table: rssloker (RSSLOKER)
-> Table: rsslokerka (RSSLOKERKA)
-> Table: smslokermanual (SMSLOKERMANUAL)
```

```
-> Table: smslokerrss (SMSLOKERRSS)
-> Table: tips (TIPS)
Creating Tables...
-> Table: adminakun (ADMINAKUN)
-> Creating indexes of the table adminakun (ADMINAKUN)...
-> Table: kabar (KABAR)
-> Creating indexes of the table kabar (KABAR)...
-> Table: konfirmasibayar (KONFIRMASIBAYAR)
-> Creating indexes of the table konfirmasibayar (KONFIRMASIBAYAR)...
-> Index: RELATIONSHIP_10_FK (RELATIONSHIP_10_FK)
-> Table: kualifikasiakademik (KUALIFIKASIAKADEMIK)
-> Creating indexes of the table kualifikasiakademik (KUALIFIKASIAKADEMIK)...
-> Table: lokermanual (LOKERMANUAL)
-> Creating indexes of the table lokermanual (LOKERMANUAL)...
-> Table: lokermanualka (LOKERMANUALKA)
-> Creating indexes of the table lokermanualka (LOKERMANUALKA)...
-> Index: RELATIONSHIP_3_FK (RELATIONSHIP_3_FK)
-> Index: RELATIONSHIP_4_FK (RELATIONSHIP_4_FK)
-> Table: memberakun (MEMBERAKUN)
-> Creating indexes of the table memberakun (MEMBERAKUN)...
-> Index: RELATIONSHIP_5_FK (RELATIONSHIP_5_FK)
-> Table: rssbuff (RSSBUFF)
-> Creating indexes of the table rssbuff (RSSBUFF)...
-> Table: rssloker (RSSLOKER)
-> Creating indexes of the table rssloker (RSSLOKER)...
-> Table: rsslokerka (RSSLOKERKA)
-> Creating indexes of the table rsslokerka (RSSLOKERKA)...
-> Index: RELATIONSHIP_1_FK (RELATIONSHIP_1_FK)
-> Index: RELATIONSHIP_2_FK (RELATIONSHIP_2_FK)
-> Table: smslokermanual (SMSLOKERMANUAL)
-> Creating indexes of the table smslokermanual (SMSLOKERMANUAL)...
-> Index: RELATIONSHIP_8_FK (RELATIONSHIP_8_FK)
-> Index: RELATIONSHIP_9_FK (RELATIONSHIP_9_FK)
-> Table: smslokerrss (SMSLOKERRSS)
-> Creating indexes of the table smslokerrss (SMSLOKERRSS)...
-> Index: RELATIONSHIP_6_FK (RELATIONSHIP_6_FK)
-> Index: RELATIONSHIP_7_FK (RELATIONSHIP_7_FK)
-> Table: tips (TIPS)
-> Creating indexes of the table tips (TIPS)...
Creating References...
-> Reference: konfirmasibayar_reference_memberakun
(KONFIRMASIBAYAR_REFERENCE_MEMBERAKUN)
-> Reference: lokermanualka_reference_kualifikasiakademik
(LOKERMANUALKA_REFERENCE_KUALIFIKASIAKADEMIK)
-> Reference: lokermanualka_reference_lokermanual
(LOKERMANUALKA_REFERENCE_LOKERMANUAL)
-> Reference: memberakun_reference_kualifikasiakademik
(MEMBERAKUN_REFERENCE_KUALIFIKASIAKADEMIK)
-> Reference: rsslokerka_reference_kualifikasiakademik
(RSSLOKERKA_REFERENCE_KUALIFIKASIAKADEMIK)
-> Reference: rsslokerka_reference_rssloker (RSSLOKERKA_REFERENCE_RSSLOKER)
-> Reference: smslokermanual_reference_lokermanual
(SMSLOKERMANUAL_REFERENCE_LOKERMANUAL)
-> Reference: smslokermanual_reference_memberakun
(SMSLOKERMANUAL_REFERENCE_MEMBERAKUN)
-> Reference: smslokerrss_reference_memberakun
(SMSLOKERRSS_REFERENCE_MEMBERAKUN)
-> Reference: smslokerrss_reference_rssloker (SMSLOKERRSS_REFERENCE_RSSLOKER)
Script Generation completed
Generation successful

Usage:
  (1) Start command prompt
  (2) Go to the directory
D:\DOKUMEN\PUTRI\skripsi\^^SKRIPSIKU_1\BAB6\pengujian-db\
  (3) Start the SQL interpreter: mysql.exe
  (4) Run the database creation script: mysql> source fromPDM.sql
```

Gambar 6.3 *Script Generate Database* dari basis data `silokdbtester`
Sumber: Pengujian

```
C:\Program Files\MySQL\MySQL Server 5.0\bin>mysql -u root -p
Enter password: *********
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 4
Server version: 5.0.51b-community-nt MySQL Community Edition (GPL)

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> use silokdbtester;
Database changed
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mydb               |
| mysql              |
| silokdb            |
| silokdbtester      |
| test               |
| testdb             |
+--------------------+
7 rows in set (0.03 sec)
mysql>
```

Gambar 6.4 Basis data `silokdbtester` hasil proses *Generate Database* pada Sybase PowerDesigner 10
Sumber: Pengujian

```
mysql> use silokdbtester;
Database changed
mysql> show tables;
+-------------------------+
| Tables_in_silokdbtester |
+-------------------------+
| adminakun               |
| kabar                   |
| konfirmasibayar         |
| kualifikasiakademik     |
| lokermanual             |
| lokermanualka           |
| memberakun              |
| rssbuff                 |
| rssloker                |
| rsslokerka              |
| smslokermanual          |
| smslokerrss             |
| tips                    |
+-------------------------+
13 rows in set (0.00 sec)
```

Gambar 6.5 Tabel pada basis data `silokdbtester` hasil proses *Generate Database* pada Sybase PowerDesigner 10
Sumber: Pengujian

```
mysql> use silokdb;
Database changed
mysql> show tables;
+-------------------------+
| Tables_in_silokdb       |
+-------------------------+
| adminakun               |
| kabar                   |
| konfirmasibayar         |
| kualifikasiakademik     |
| lokermanual             |
| lokermanualka           |
| memberakun              |
| rssbuff                 |
| rssloker                |
| rsslokerka              |
| smslokermanual          |
| smslokerrss             |
| tips                    |
+-------------------------+
13 rows in set (0.00 sec)
```

Gambar 6.6 Tabel pada basis data `silokdb` hasil implementasi
Sumber: Pengujian

Analisis : Dari hasil pengujian, terlihat bahwa tabel `memberakun`, `kualifikasiakademik`, `rssloker`, `lokermanual`, `rsslokerka`, `lokermanualka`, `konfirmasibayar`, `smslokerrss`, `smslokermanual`, `adminakun`, `kabar`, `tips` dan `rssbuff` berhasil di-*generate* menjadi tabel-tabel yang berelasi di dalam basis data `silokdbtester`. Tabel pada basis data `silokdbtester` hasil perancangan sesuai dengan tabel hasil implementasi pada basis data `silokdb`. Hal ini menunjukkan bahwa desain database yang dibuat sudah benar (*valid*)

6.1.2 Pengujian Koneksi Basis Data dan *Web Server*

Nama Kasus Uji : Kasus Uji Koneksi *Database*

Tujuan Pengujian : Pengujian ini dilakukan untuk memastikan bahwa komputer *client* dapat melakukan koneksi dengan *database* MySQL yang berada pada komputer *server*.

Spesifikasi *Hardware* : PC *Server* Aplikasi:

- Prosesor Intel ® Pentium ® Dual CPU T2390 @ 1.86 GHz, 2 GB RAM
- Sistem operasi Microsoft Windows XP Professional *Version* 2002 SP2
- Alamat IP pada perangkat Ethernet: 192.168.1.3

PC *Client*:

- Prosesor AMD Turion™ 64 X2 Mobile Technology TL-52 1.6 GHz, 2 GB RAM
- Sistem operasi Microsoft Windows Vista™ Home Premium
- Alamat IP pada perangkat Ethernet: 192.168.1.4

Spesifikasi *Software*

PC *Server* Aplikasi:

- *Database Server* MySQL 5.0.51b (`mysqld-nt.exe`)
- *Application Server* Glassfish v 2
- J2SE Development Kit 6.0 Update 11
- J2SE Runtime Environment 6.0 Update 11
- SQL *Shell* (`mysql.exe`)

PC *Client*:

- *Web Browser* Mozilla Firefox versi 3.0.4

Prosedur Uji : PC *Server* Aplikasi:

Sebuah *window* Command Prompt dijalankan dari:

```
Start | Run... | Open: cmd.exe
```

Server database MySQL dijalankan sebagai *service* dengan memberikan perintah:

```
C:\>net start mysql
```

Aplikasi yang sedang berjalan dan koneksi yang sedang

aktif ditampilkan dengan memberikan perintah:

```
C:\>netstat –an
```

PC *Client*:

1. Membuka aplikasi *web* skripsi v.1 (`http://192.168.1.3:8080/ProjectSkripsi/`).
2. Melakukan proses *login* sebagai Member, atau *login* sebagai Administrator.
3. Aplikasi yang sedang berjalan dan koneksi yang sedang aktif ditampilkan dengan memberikan perintah:

   ```
   C:\>netstat –an
   ```

PC *Server* Aplikasi:

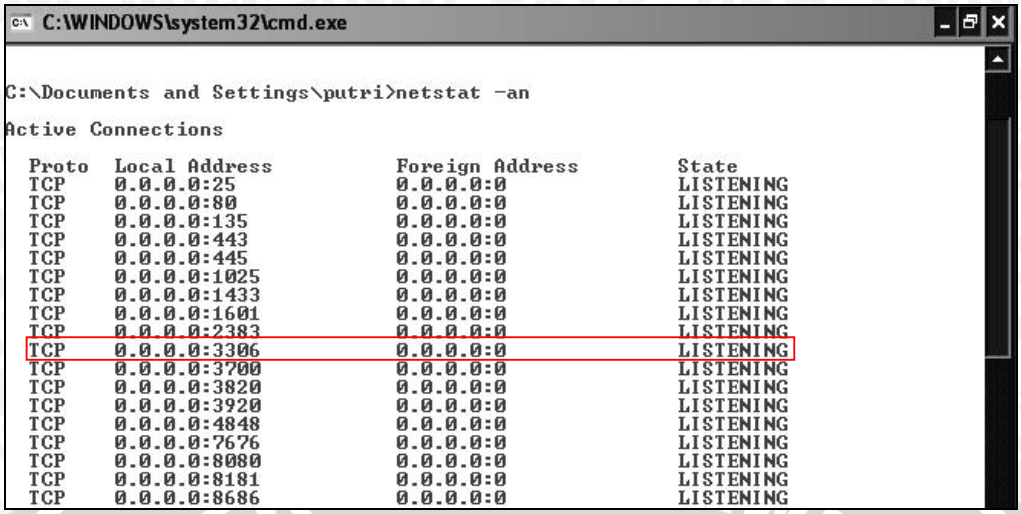
1. Aplikasi yang sedang berjalan dan koneksi yang sedang aktif ditampilkan kembali dengan memberikan perintah:

   ```
   C:\>netstat –an
   ```

Hasil yang diharapkan : Komputer *client* dapat melakukan koneksi dengan *web server* yang berada pada komputer *server* dan aplikasi yang telah di deploy di *server* bisa terhubung ke *database server*.

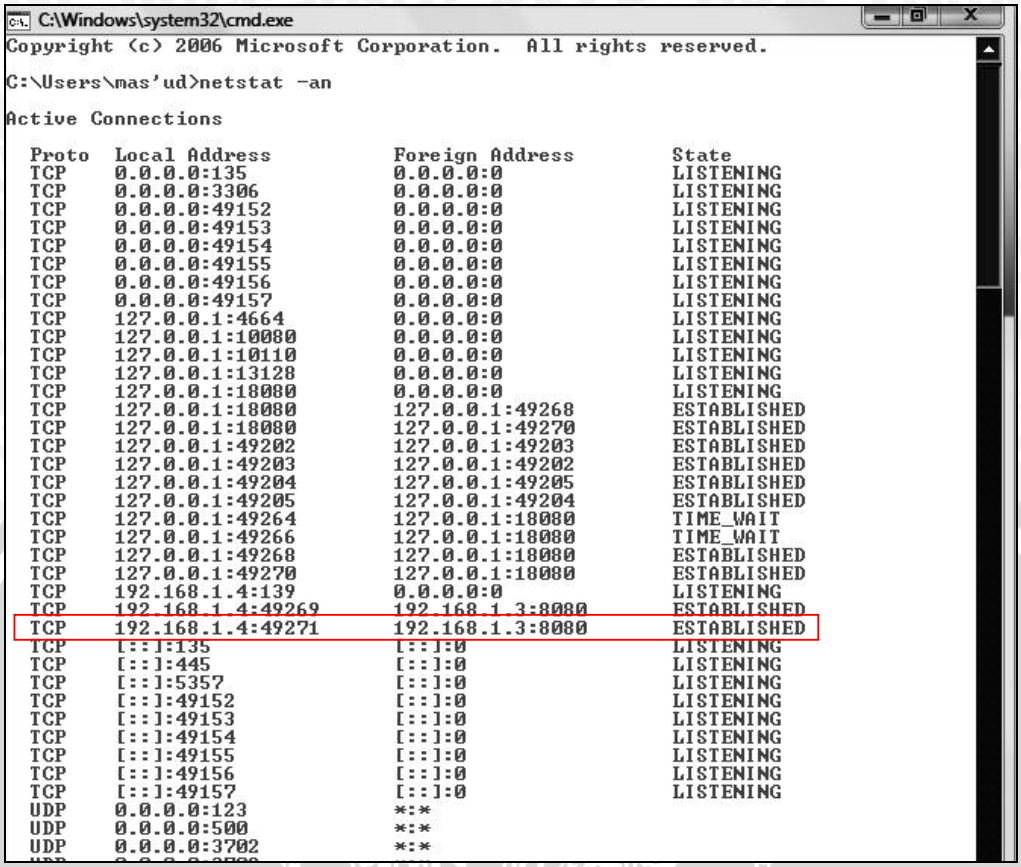
Hasil Pengujian : Komputer *client* dapat melakukan koneksi dengan *web server* yang berada pada komputer *server* dan aplikasi yang telah di deploy di *server* telah bisa terhubung ke *database server*.

Hasil dari penggunaan perintah `netstat –an` pada komputer *server* aplikasi sebelum ada koneksi dengan komputer *client* ditunjukkan dalam Gambar 6.7. Perintah tersebut digunakan untuk menampilkan koneksi yang sedang aktif. Dari gambar tersebut terlihat bahwa *database server* MySQL (`mysqld-nt.exe`) memiliki kondisi (*state*) `LISTENING` pada alamat lokal `0.0.0.0:3306`. Hal tersebut berarti bahwa *database server* MySQL telah siap untuk menerima sebuah koneksi *database* pada *port* TCP 3306.



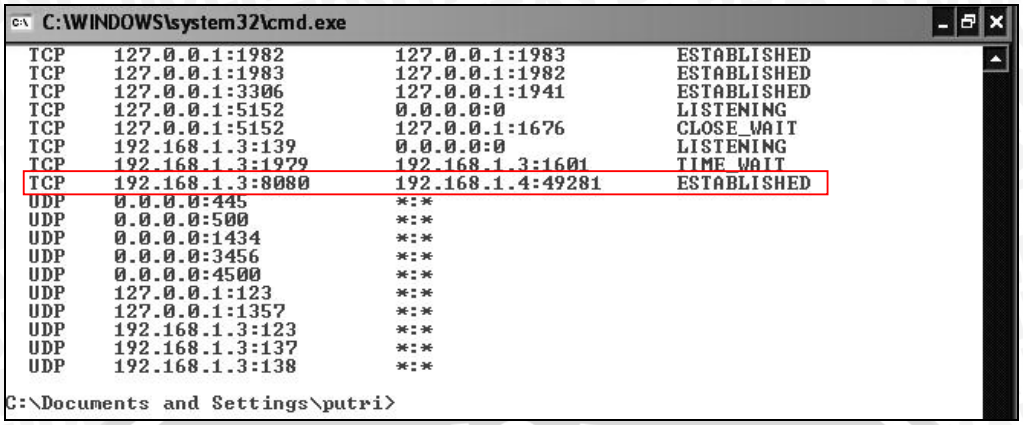
Gambar 6.7 Koneksi yang sedang aktif pada komputer *server* sebelum aplikasi dijalankan pada komputer *client*
Sumber: Pengujian

Hasil dari penggunaan perintah `netstat -an` pada komputer *client* setelah membuka aplikasi ini ditunjukkan dalam Gambar 6.8. Perintah tersebut digunakan untuk menampilkan koneksi yang sedang aktif. Dari gambar tersebut terlihat bahwa terdapat koneksi antara komputer *server* (alamat IP: `192.168.1.3`) dengan komputer *client* (alamat IP: `192.168.1.4`) ini ditunjukkan dengan adanya koneksi antara alamat IP `192.168.1.4:49271` ke alamat IP `192.168.1.3:8080` dengan kondisi (*state*): `ESTABLISHED`.



Gambar 6.8 Koneksi yang sedang aktif pada komputer *client* setelah aplikasi dijalankan dan dihubungkan dengan *database server* MySQL
Sumber: Pengujian

Untuk menampilkan koneksi yang sedang aktif pada komputer *server* digunakan perintah `netstat -an`. Gambar 6.9 menunjukkan koneksi yang sedang aktif pada komputer *server* setelah aplikasi dibuka pada komputer *client*. Dari gambar tersebut terlihat bahwa terdapat koneksi antara komputer *server* (alamat IP: `192.168.1.3`) dengan komputer *client* (alamat IP: `192.168.1.4`) ini ditunjukkan dengan adanya koneksi antara alamat IP `192.168.1.4:49281` ke alamat IP `192.168.1.3:8080` dengan kondisi (*state*): `ESTABLISHED`.



Gambar 6.9 Koneksi yang sedang aktif pada komputer *server* setelah aplikasi dijalankan pada komputer *client*
Sumber: Pengujian

Analisis : Aplikasi dapat dibuka dan dijalankan pada jaringan komputer yang menggunakan protokol TCP/IP. Permintaan proses dari *client* ke *server* dilakukan melalui protokol HTTP. Aplikasi ini juga memiliki koneksi ke *database server* yang di deploy dalam komputer *server*.

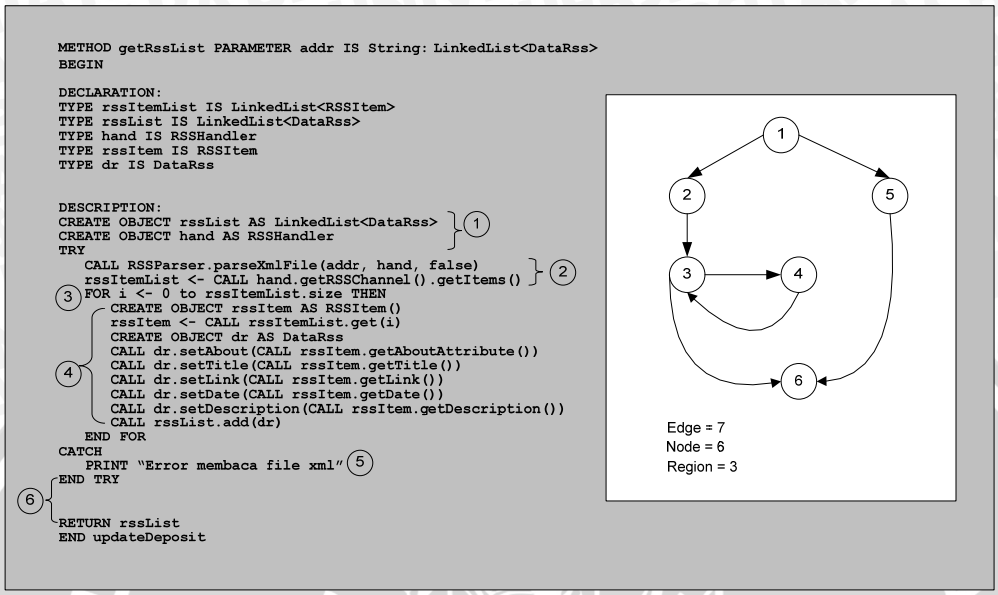
6.1.3 Pengujian Unit

Dalam sistem yang dibangun dengan pemrograman berorientasi objek, pengujian unit diterapkan untuk suatu metode (operasi) dari suatu *class*. Pada pengujian unit ini, digunakan teknik pengujian *white box (white box testing)* dengan teknik *basis path testing*. Pada teknik *basis path testing*, proses pengujian dilakukan dengan memodelkan algoritma pada suatu *flow graph*, menentukan jumlah kompleksitas siklomatis *(cyclomatic complexity)*, menentukan sebuah basis set dari jalur independen dan memberikan kasus uji *(test case)* pada setiap basis set yang telah ditentukan. Di dalam penulisan laporan skripsi ini, hanya dicantumkan hasil pengujian unit untuk algoritma dari beberapa metode (operasi) saja (tidak untuk keseluruhan metode).

6.1.3.1 Pengujian Unit untuk Operasi `getRssList(String addr)`

Operasi `getRssList(String addr)` merupakan implementasi algoritma untuk mengambil kumpulan RSS dari suatu alamat website. Operasi ini terdapat

dalam *class* `DataRss`. Gambar 6.10 memperlihatkan proses pemodelan dalam *flow graph* pada operasi `getRssList(String addr)`.



Gambar 6.10 Pemodelan operasi `getRssList(String addr)` ke dalam *flow graph*
Sumber : Pengujian

Dari hasil pemodelan ke dalam *flow graph* pada Gambar 6.10 yang telah dilakukan terhadap operasi `getRssList(String addr)`, ditentukan jumlah kompleksitas siklomatis *(cyclomatic complexity)* melalui persamaan V(G) = E – N + 2, dimana V(G) merupakan jumlah kompleksitas siklomatis, E merupakan sisi (garis penghubung antar *node*) dan N merupakan jumlah simpul *(node).*

$$V(G) = E - N + 2$$
$$= 7 - 6 + 2$$
$$= 3$$

Dari nilai *cyclomatic complexity* yang telah dihasilkan dari perhitungan yaitu V(G) = 3 ditentukan dua basis set dari jalur independen yaitu:

Jalur 1 : 1-2-3-4-3-..

Jalur 2 : 1-2-3-6

Jalur 3 : 1-5-6

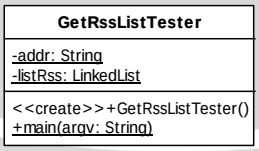
Penentuan kasus uji *(test case)* untuk masing-masing jalur dan hasil eksekusi untuk masing-masing kasus uji adalah seperti pada Tabel 6.1

Tabel 6.1 *Test case* untuk pengujian unit operasi `getRssList(String addr)`

Jalur	Kasus Uji	Hasil yang diharapkan	Hasil yang didapatkan
1	Parameter `addr` diisi dengan url rss yang valid(**D:\DOKUMEN\PUTRI\skripsi\^^SKRIPSIKU_1\project_skripsi\main_pro_v_1.1\ProjectSkripsi\feeds.xml**) dan terdapat data rss di dalamnya	Method `getRssList(String addr)`mengembalikan list data RSS (terdapat 3 buah item RSS) ke klas dummy pemanggilnya (`GetRssListTester`)	Method `getRssList(String addr)`berhasil mengembalikan 3 item data RSS ke klas `GetRssListTester`
2	Parameter `addr` diisi dengan alamat yang valid, tetapi tidak ada data rss di dalamnya (**D:\DOKUMEN\PUTRI\skripsi\^^SKRIPSIKU_1\project_skripsi\main_pro_v_1.1\ProjectSkripsi\feeds-empty.xml**)	Method `getRssList(String addr)`mengembalikan list kosong data RSS (terdapat 0 buah item RSS) ke klas dummy pemanggilnya (`GetRssListTester`)	Method `getRssList(String addr)`berhasil mengembalikan list kosong data RSS ke klas `GetRssListTester`
3	Parameter `addr` diisi dengan url yang tidak valid	Method `getRssList(String addr)`mengembalikan list kosong data RSS (terdapat 0 buah item RSS) ke klas dummy pemanggilnya (`GetRssListTester`) dan muncul exception karena url tidak valid	Method `getRssList(String addr)`berhasil mengembalikan list kosong data RSS ke klas `GetRssListTester` serta muncul exception akibat url yang tida valid

Sumber : Pengujian

Untuk proses pengujian tersebut digunakan sebuah *class dummy* yaitu *class* `GetRssListTester`. Atribut dan operasi yang terdapat pada *class* `GetRssListTester` ditunjukkan dalam Gambar 6.11.

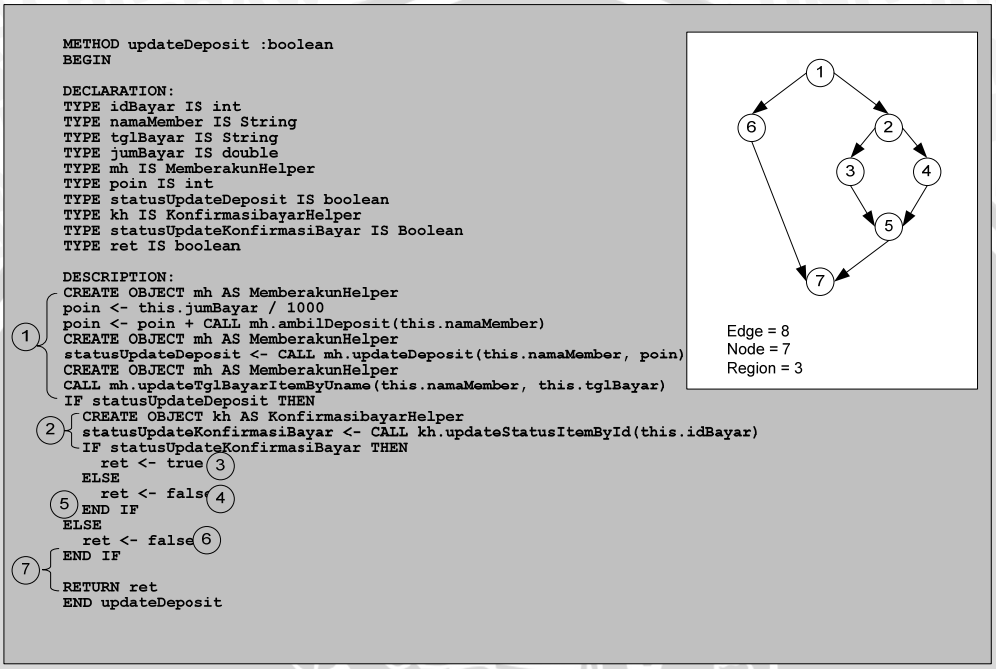


Gambar 6.11 Diagram *class dummy* `GetRssListTester`

Sumber: Pengujian

6.1.3.2 Pengujian Unit untuk Operasi `updateDeposit()`

Operasi `updateDeposit()` merupakan implementasi dari algoritma untuk mengubah nilai deposit dari member. Operasi ini terdapat dalam *class* `Deposit`. Gambar 6.12 memperlihatkan proses pemodelan dalam *flow graph* pada operasi `updateDeposit()`.



Gambar 6.12 Pemodelan operasi `updateDeposit()` ke dalam *flow graph*
Sumber : Pengujian

Dari hasil pemodelan ke dalam *flow graph* pada Gambar 6.11 yang telah dilakukan terhadap operasi `updateDeposit()`, ditentukan jumlah kompleksitas siklomatis *(cyclomatic complexity)* melalui persamaan V(G) = E – N + 2, dimana V(G) merupakan jumlah kompleksitas siklomatis, E merupakan sisi (garis penghubung antar *node*) dan N merupakan jumlah simpul *(node).*

V(G) = E – N + 2
= 8 – 7 + 2
= 3

Dari nilai *cyclomatic complexity* yang telah dihasilkan dari perhitungan yaitu V(G) = 3 ditentukan dua basis set dari jalur independen yaitu:

Jalur 1 : 1-2-3-5-7

Jalur 2 : 1-2-4-5-7

Jalur 3 : 1-6-7

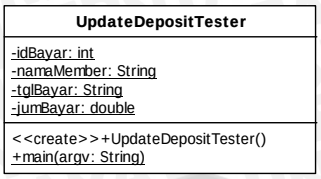
Penentuan kasus uji *(test case)* untuk masing-masing jalur dan hasil eksekusi untuk masing-masing kasus uji adalah seperti pada Tabel 6.2.

Tabel 6.2 *Test case* untuk pengujian unit operasi `updateDeposit()`

Jalur	Kasus Uji	Hasil yang diharapkan	Hasil yang didapatkan
1	Memberikan nilai idbayar = 30, tglBayar="8/27/2009", namaMember="putri", jumBayar=10000 (semua nilai valid, sesuai dengan data yang ada di database)	Method `updateDeposit()`mengembalikan "true" ke klas pemanggilnya (`UpdateDepositTester`). Hal ini berarti deposit member dengan nama "putri" bertambah 10 poin (menjadi 188), dan data transaksi konfirmasi bayar dengan id=30, dari status awal baru menjadi tersimpan.	Method `updateDeposit()`berhasil mengembalikan "true" ke klas pemanggilnya (`UpdateDepositTester`). Deposit member dengan nama "putri" telah bertambah 10 poin (menjadi 188), dan status data transaksi konfirmasi bayar dengan id=30 berubah menjadi tersimpan.
2	Memberikan nilai idbayar = 32, tglBayar="8/27/2009", namaMember="putri", jumBayar=10000 (nilai idBayar tidak valid karena tidak ada transaksi konfirmasi bayar yang tersimpan di database dengan id=32)	Method `updateDeposit()`mengembalikan "false" ke klas pemanggilnya (`UpdateDepositTester`). Tidak ada perubahan data transaksi konfirmasi bayar	Method `updateDeposit()berhasil` mengembalikan "false" ke klas pemanggilnya (`UpdateDepositTester`) dan Tidak ada perubahan data transaksi konfirmasi bayar
3	Memberikan nilai idbayar = 30, tglBayar="8/27/2009", namaMember="elfa", jumBayar=10000 (nilai namaMember tidak valid karena tidak ada data member yang tersimpan di database dengan nama="elfa")	Method `updateDeposit()`mengembalikan "false" ke klas pemanggilnya (`UpdateDepositTester`). Tidak ada perubahan data deposit dari member	Method `updateDeposit()berhasil` mengembalikan "false" ke klas pemanggilnya (`UpdateDepositTester`). Dan Tidak ada perubahan data deposit dari member

Sumber : Pengujian

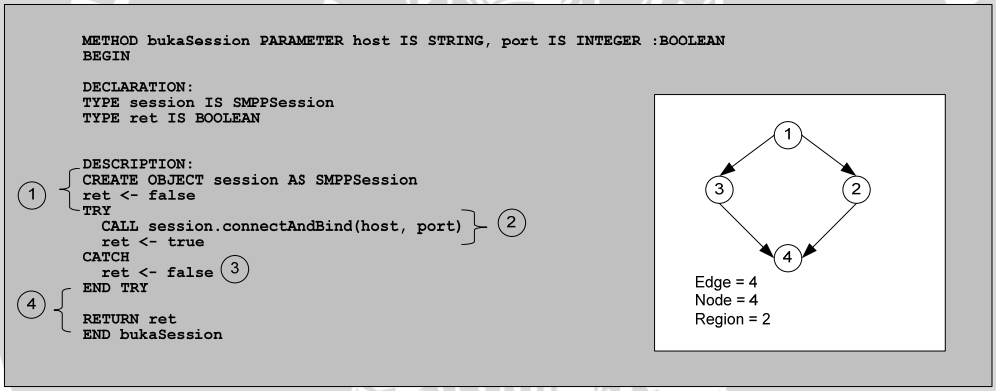
Untuk proses pengujian tersebut digunakan sebuah *class dummy* yaitu *class* `UpdateDepositTester`. Atribut dan operasi yang terdapat pada *class* `UpdateDepositTester` ditunjukkan dalam Gambar 6.13.



Gambar 6.13 Diagram *class dummy* `UpdateDepositTester`
Sumber: Pengujian

6.1.3.3 Pengujian Unit untuk Operasi `bukaSession(String host, int port)`

Operasi `bukaSession(String host, int port)` merupakan implementasi algoritma proses pembukaan konseksi dengan server SMPP. Operasi ini terdapat dalam *class* `PengirimSmsViaSmpp`. Gambar 6.14 memperlihatkan proses pemodelan dalam *flow graph* pada operasi `bukaSession(String host, int port)`.



Gambar 6.14 Pemodelan operasi **`bukaSession(String host, int port)`**ke dalam *flow graph*
Sumber : Pengujian

Dari hasil pemodelan ke dalam *flow graph* pada Gambar 6.14 yang telah dilakukan terhadap operasi **`bukaSession(String host, int port)`**, ditentukan jumlah kompleksitas siklomatis *(cyclomatic complexity)* melalui persamaan V(G) = E – N + 2, dimana V(G) merupakan jumlah kompleksitas siklomatis, E merupakan sisi (garis penghubung antar *node*) dan N merupakan jumlah simpul *(node)*.

$$V(G) = E - N + 2$$
$$= 4 - 4 + 2$$
$$= 2$$

Dari nilai *cyclomatic complexity* yang telah dihasilkan dari perhitungan yaitu V(G) = 2 ditentukan satu basis set dari jalur independen yaitu:

Jalur 1 : 1-2-4

Jalur 2 : 1-3-4

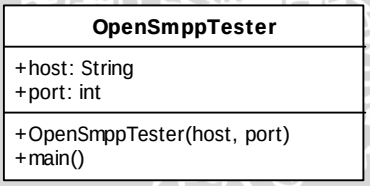
Penentuan kasus uji *(test case)* untuk masing-masing jalur dan hasil eksekusi untuk masing-masing kasus uji adalah seperti pada Tabel 6.3

Tabel 6.3 *Test case* untuk pengujian unit operasi **`bukaSession(String host, int port)`**

Jalur	Kasus Uji	Hasil yang diharapkan	Hasil yang didapatkan
1	Jalankan SMPP Server pada localhost, host = "localhost", port=8056	Method `bukaSession()` mengembalikan nilai "true" ke klas pemanggil (`OpenSmppTester`).	Method `bukaSession()` `berhasil` mengembalikan nilai "true" ke klas pemanggil (`OpenSmppTester`).
2	SMPP Server tidak dijalankan, host = "localhost", port=8056. Atau Jalankan SMPP Server pada localhost, host = "open,com", port=8056	Method `bukaSession()` mengembalikan nilai "false" ke klas pemanggil (`OpenSmppTester`).	Method `bukaSession()` mengembalikan nilai "false" ke klas pemanggil (`OpenSmppTester`).

Sumber : Pengujian

Untuk proses pengujian tersebut digunakan sebuah *class dummy* yaitu *class* `SmsLokerTester`. Atribut dan operasi yang terdapat pada *class* `OpenSmppTester` ditunjukkan dalam Gambar 6.15.



Gambar 6.15 Diagram *class dummy* `OpenSmppTester`

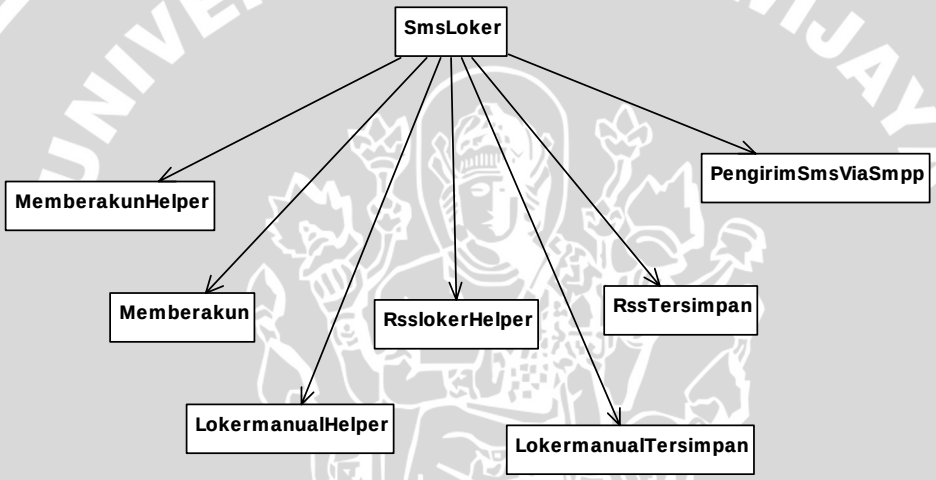
Sumber: Pengujian

6.1.4 Pengujian Integrasi

Pengujian integrasi diterapkan pada proses yang mengintegrasikan fungsionalitas dari beberapa *class* untuk melakukan sebuah operasi tertentu. Pada pengujian integrasi yang dijadikan sebagai obyek uji adalah *class-class* yang menggabungkan kinerja dari *class-class* yang lain. *Class-class* yang mengintegrasikan beberapa *class* yang lain tersebut akan berperan juga sebagai

test driver yang mengendalikan proses pengujian sehingga bisa memperlihatkan status valid atau tidaknya hasil integrasi.

Dalam laporan tugas akhir ini pengujian integrasi dilakukan pada method `kirimSmsPerMember(String idLoker, String jenisLoker, List<String> member)`yang ada dalam klas SmsLoker. Dalam operasi yang dilakukan pada fungsi `kirimSmsPerMember(String idLoker, String jenisLoker, List<String> member)`, dilibatkan beberapa klas yaitu klas `MemberakunHelper`, `Memberakun`, `RsslokerHelper`, `RssTersimpan`, `LokermanualHelper`, `LokermanualTersimpan` dan klas `PengirimSmsViaSmpp`.



Gambar 6.16 Relasi *class* `SmsLoker`
Sumber: Pengujian

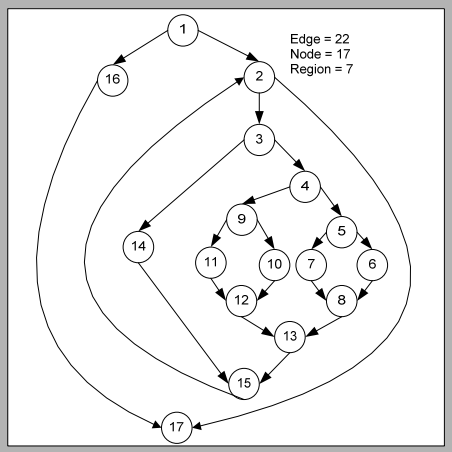
Teknik *Basis Path Testing* yang telah digunakan dalam tahap pengujian unit, diterapkan juga sebagai teknik dalam tahap pengujian integrasi. Operasi `kirimSmsPerMember(String idLoker, String jenisLoker, List<String> member)` merupakan implementasi algoritma proses pengiriman SMS informasi lowongan kerja kepada member. Operasi ini terdapat dalam *class* `SmsLoker`. Gambar 6.17 memperlihatkan proses pemodelan dalam *flow graph* pada operasi `kirimSmsPerMember(String idLoker, String jenisLoker, List<String> member)`.

```
METHOD kirimSmsPerMember PARAMETER idLoker IS String, jenisLoker IS String, member IS List<String> : boolean
BEGIN

DECLARATION:
TYPE ret IS Boolean
TYPE isiReport IS String
TYPE memberSukses IS String
TYPE memberGagal IS String
TYPE lokerGagal IS String
TYPE lnkLoker IS String
TYPE idLok IS int
TYPE p IS PengirimSmsViaSmpp
TYPE namaMember IS String
TYPE mh IS MemberakunHelper
TYPE m IS Memberakun
TYPE pesan IS String
TYPE noHp IS String
TYPE rh IS RsslokerHelper
TYPE rssLoker IS RssTersimpan
TYPE lh IS LokermanualHelper
TYPE manLoker IS LokermanualTersimpan

DESCRIPTION:
ret <- false
memberSukses  <- "<ul>"
memberGagal <- "<ul>"
lokerGagal <- "<ul>"
lnkLoker <- ""
idLok <- CALL Integer.ParseInt(idLoker)
CREATE OBJECT p AS PengirimSmsViaSmpp
IF (CALL p.bukaSession()) THEN
  FOR i <- 0 to i<member.size THEN
    namaMember <- CALL member.get(i)
    CREATE OBJECT mh AS MemberakunHelper
    m <- CALL mh.ambilMember(namaMember)
    IF m != null THEN
      noHp <- CALL m.getNohp()
      IF jenisLoker="rss" THEN
        CREATE OBJECT rh AS RsslokerHelper
        rssLoker <- CALL rh.getLokerRssById(idLok)
        IF rssLoker != null THEN
          pesan <- CALL rssLoker.getTitle+CALL rssLoker.getLink+CALL rssLoker.getPosisi+CALL rssLoker.getPerusahaan+
                   CALL rssLoker.getBatasAkhir
          CALL p.kirimSms(noHp,pesan)
          CREATE OBJECT mh AS MemberakunHelper
          CALL mh.setRsslokers(namaMember,idLok)
          memberSukses <- memberSukses+ "<li><a href=\"lihatDetilMember.do?hal=3&u="+namaMember+"&lnk=subhal=3\
          ">"+namaMember+"</a></li>"
        ELSE
          lokerGagal <- lokerGagal+"<li>"+idLok+" jenis loker = RSS Loker</li>"
        END IF
      ELSE
        CREATE OBJECT lh AS LokermanualHelper
        manLoker <- CALL lh.getLokerManualById(idLok)
        IF manLoker != null THEN
          pesan <- CALL manLoker.getJudul+CALL manLoker.getPerusahaan+CALL manLoker.getBatasakhir+
                   CALL manLoker.getPosisi
          CALL p.kirimSms(noHp,pesan)
          CREATE OBJECT mh AS MemberakunHelper
          CALL mh.setLokermanulas(namaMember,idLok)
          memberSukses <- memberSukses+ "<li><a href=\"lihatDetilMember.do?hal=3&u="+namaMember+"&lnk=subhal=3\
          ">"+namaMember+"</a></li>"
        ELSE
          lokerGagal <- lokerGagal+"<li>"+idLok+" jenis loker = Manual Loker</li>"
        END IF
      END IF
    ELSE
      memberGagal <- memberGagal+"<li>"+namaMember+"</li>"
    END IF
  END FOR
  CALL p.tutupSession()
  memberSukses=memberSukses+"</ul>"
  memberGagal=memberGagal+"</ul>"
  lokerGagal=lokerGagal+"</ul>"
  isiReport <- "Daftar user yang berhasil dikirimi loker ("+lnkLoker+idLoker+"</a>) via SMS : "+memberSukses
  isiReport <- isiReport + "<br>Daftar user yang gagal dikirimi : "+memberGagal
  isiReport <- isiReport + "<br>Daftar loker yang gagal dikirimkan : "+lokerGagal
  ret <- true
ELSE
  isiReport <- "Gagal terhubung ke SMPP server";
  ret <- false;
END IF

RETURN ret
END kirimSmsPerMember
```



Gambar 6.17 Pemodelan operasi kirimSmsPerMember(String idLoker, String jenisLoker, List<String> member) ke dalam *flow graph*
Sumber : Pengujian

Dari hasil pemodelan ke dalam *flow graph* pada Gambar 6.13 yang telah dilakukan terhadap operasi `kirimSmsPerMember(String idLoker, String jenisLoker, List<String> member)`, ditentukan jumlah kompleksitas siklomatis *(cyclomatic complexity)* melalui persamaan V(G) = E – N + 2, dimana V(G) merupakan jumlah kompleksitas siklomatis, E merupakan sisi (garis penghubung antar *node*) dan N merupakan jumlah simpul *(node).*

V(G) = E – N + 2

= 22 – 17 + 2

= 7

Dari nilai *cyclomatic complexity* yang telah dihasilkan dari perhitungan yaitu V(G) = 8 ditentukan satu basis set dari jalur independen yaitu:

Jalur 1 : 1-2-17

Jalur 2 : 1-16-17

Jalur 3 : 1-2-3-4-5-6-8-13-15-2-…

Jalur 4 : 1-2-3-4-5-7-8-13-15-2-…

Jalur 5 : 1-2-3-4-9-10-12-13-15-2-…

Jalur 6 : 1-2-3-4-9-11-12-13-15-2-…

Jalur 7 : 1-2-3-14-15-2-…

Penentuan kasus uji *(test case)* untuk masing-masing jalur dan hasil eksekusi untuk masing-masing kasus uji adalah seperti pada Tabel 6.4.

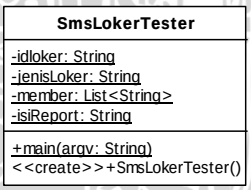
Tabel 6.4 *Test case* untuk pengujian unit operasi `kirimSmsPerMember(String idLoker, String jenisLoker, List<String> member)`

Jalur	Kasus Uji	Hasil yang diharapkan	Hasil yang didapatkan
1	Jalankan SMPP Server, idLoker = “21”, jenisLoker=”man”, member=”putri”	Method `kirimSmsPerMember` mengembalikan nilai “true” ke klas pemanggil (`SmsLokerTester`). Pada atribut memberSukses=”putri”, memberGagal=””, lokerGagal=””	Method `kirimSmsPerMember` berhasil mengembalikan nilai “true” ke klas pemanggil (`SmsLokerTester`). Atribut memberSukses=”putri”, memberGagal=””, lokerGagal=””
2	SMPP Server tidak dijalankan, idLoker = “21”, jenisLoker=”man”, member=”putri”	Method `kirimSmsPerMember` mengembalikan nilai “false” ke klas pemanggil (`SmsLokerTester`).	Method `kirimSmsPerMember` berhasil mengembalikan nilai “false” ke klas pemanggil (`SmsLokerTester`).
3	Jalankan SMPP Server, idLoker = “21”, jenisLoker=”rss”, member=”putri”	Method `kirimSmsPerMember` mengembalikan nilai “true” ke klas pemanggil (`SmsLokerTester`). Pada atribut memberSukses=”putri”, memberGagal=””, lokerGagal=””	Method `kirimSmsPerMember` berhasil mengembalikan nilai “true” ke klas pemanggil (`SmsLokerTester`). Atribut memberSukses=”putri”, memberGagal=””, lokerGagal=””
4	Jalankan SMPP Server, idLoker = “0”, jenisLoker=”rss”, member=”putri”	Method `kirimSmsPerMember` mengembalikan nilai “true” ke klas pemanggil (`SmsLokerTester`). Pada atribut memberSukses=”putri”,	Method `kirimSmsPerMember` berhasil mengembalikan nilai “true” ke klas pemanggil (`SmsLokerTester`). Atribut memberSukses=”putri”,

		memberGagal="", lokerGagal="0"	memberGagal="", lokerGagal="0"
5	Jalankan SMPP Server, idLoker = "21", jenisLoker="man", member="putri"	Method `kirimSmsPerMember` mengembalikan nilai "true" ke klas pemanggil (`SmsLokerTester`). Pada atribut memberSukses="", memberGagal="putri".	Method `kirimSmsPerMember` berhasil mengembalikan nilai "true" ke klas pemanggil (`SmsLokerTester`). Atribut memberSukses="", memberGagal="putri".
6	Jalankan SMPP Server, idLoker = "0", jenisLoker="man", member="putri"	Method `kirimSmsPerMember` mengembalikan nilai "true" ke klas pemanggil (`SmsLokerTester`). Pada atribut memberSukses="putri", memberGagal="", lokerGagal="0"	Method `kirimSmsPerMember` berhasil mengembalikan nilai "true" ke klas pemanggil (`SmsLokerTester`). Atribut memberSukses="putri", memberGagal="", lokerGagal="0"
7	Jalankan SMPP Server, idLoker = "21", jenisLoker="man", member="elfa"	Method `kirimSmsPerMember` mengembalikan nilai "true" ke klas pemanggil (`SmsLokerTester`). Pada atribut memberSukses="", memberGagal="putri".	Method `kirimSmsPerMember` berhasil mengembalikan nilai "true" ke klas pemanggil (`SmsLokerTester`). Atribut memberSukses="", memberGagal="putri".

Sumber : Pengujian

Untuk proses pengujian tersebut digunakan sebuah *class dummy* yaitu *class* `SmsLokerTester`. Atribut dan operasi yang terdapat pada *class* `SmsLokerTester` ditunjukkan dalam Gambar 6.18.



Gambar 6.18 Diagram *class dummy* `SmsLokerTester`
Sumber: Pengujian

6.1.5 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. Dalam melakukan pengujian validasi digunakan metode pengujian *Black Box*, karena tidak memerlukan untuk

berkonsentrasi terhadap alur jalannya algoritma program dan lebih ditekankan untuk menemukan konformitas antara kinerja sistem dengan daftar kebutuhan. Obyek dari pengujian validasi adalah item-item kebutuhan yang telah dirumuskan pada daftar kebutuhan pada Tabel 4.2. Nama kasus uji dan obyek pengujian ditunjukkan pada Sub Bab 6.1.5.1 dan hasil pengujiannya akan ditunjukkan pada Sub Bab 6.1.5.2.

6.1.5.1 Kasus Uji Validasi

Untuk mengetahui kesesuaian antara kebutuhan dengan kinerja sistem, pada setiap kebutuhan *(requirement)* dilakukan proses pengujian dengan masing-masing kasus uji.

a. Kasus Uji Registrasi

Nama Kasus Uji	:	Kasus Uji Registrasi
Objek Uji	:	Kebutuhan no. 1
Tujuan Pengujian	:	Pengujian dilakukan untuk memastikan bahwa aplikasi dapat memenuhi kebutuhan fungsional untuk melakukan registasi menjadi Member.
Prosedur Uji	:	1. Membuka halaman utama web. 2. Masuk ke menu ”Daftar”. 3. Memasukkan isian dalam field ”Nama Lengkap”, ”Alamat”, ”Email”, ”Tempat/Tanggal Lahir”, ”No HP”, ”Level Pendidikan Terakhir”, ”Universitas”, ”Fakultas”, ”Jurusan”, ”Konsentrasi”, ”Username”, ”Password”, dan ”Konfirmasi Password”. 4. Menekan tombol ”Daftar”.
Hasil yang diharapkan	:	Aplikasi dapat menyimpan data member yang telah dimasukkan, dan memberikan notifikasi terhadap member yang telah memasukkan datanya, bahwa data tersebut telah berhasil disimpan.

b. Kasus Uji Login Member

Nama Kasus Uji	:	Kasus Uji Login Member

Objek Uji	:	Kebutuhan no. 2
Tujuan Pengujian	:	Pengujian dilakukan untuk memastikan bahwa aplikasi dapat memenuhi kebutuhan fungsional untuk *login* sebagai Member sehingga pengguna dapat menggunakan fasilitas untuk Member.
Prosedur Uji	:	1. Membuka halaman utama (home) dari web. 2. Mengisi *field* "Username" dan "Passsword" pada tempat Login Member. 3. Menekan tombol "login".
Hasil yang diharapkan	:	Aplikasi dapat melakukan validasi nilai *username* dan *password* yang diisi pengguna dan menampilkan menu yang bisa digunakan oleh Member.

c. Kasus Uji Login Administrator

Nama Kasus Uji	:	Kasus Uji Login Administrator
Objek Uji	:	Kebutuhan no. 3
Tujuan Pengujian	:	Pengujian dilakukan untuk memastikan bahwa aplikasi dapat memenuhi kebutuhan fungsional untuk login sebagai Administrator sehingga dapat menggunakan fasilitas untuk Administrator.
Prosedur Uji	:	1. Membuka antarmuka login untuk Administrator (`http://192.168.1.3:8080/ProjectSkripsi/admin/`). 2. Mengisi *field* "Adminname", dan "Passsword" kemudian menekan tombol "login".
Hasil yang diharapkan	:	Aplikasi dapat melakukan validasi Adminname dan Password yang dimasukkan, dan menampilkan halaman Administrator.

d. Kasus Uji Penampilan Informasi Lowongan Kerja

Nama Kasus Uji	:	Kasus Uji Penampilan Informasi Lowongan Kerja
Objek Uji	:	Kebutuhan no. 4
Tujuan Pengujian	:	Pengujian dilakukan untuk memastikan bahwa aplikasi dapat memenuhi kebutuhan fungsional

untuk menampilkan informasi lowongan kerja.

Prosedur Uji : 1. Masuk ke dalam halaman utama web.
2. Memilih menu “Lowongan”.
3. Memilih “Daftar Loker RSS Baru” atau “Daftar Loker Hasil Input Manual”.

Hasil yang diharapkan : Aplikasi dapat menampilkan berbagai informasi lowongan kerja yang berasal dari data RSS (jika dipilih “Daftar Loker RSS Baru”) atau dari hasil input manual (jika dipilih “Daftar Loker Hasil Input Manual”).

e. Kasus Uji Pengubahan Profil Member

Nama Kasus Uji : Kasus Uji Pengubahan Profil Member

Objek Uji : Kebutuhan no. 5

Tujuan Pengujian : Pengujian dilakukan untuk memastikan bahwa aplikasi dapat memenuhi kebutuhan fungsional untuk mengubah profil Member.

Prosedur Uji : 1. Login sebagai Member.
2. Masuk ke dalam menu member ”Ubah Akun”
3. Memasukkan field yang ingin diubah (Nama Lenegkap, Alamat, Email, Tempat/Tanggal Lahir, No. HP, Level Pendidikan, Universitas, Fakultas, Jurusan Konsentrasi dan Password).
4. Menekan tombol ”Ubah”

Hasil yang diharapkan : Aplikasi dapat menyimpan perubahan data profil Member dan memberikan notifikasi kepada Member bahwa profil sudah berhasil diubah.

f. Kasus Uji Pengecekan Deposit Member

Nama Kasus Uji : Kasus Uji Pengecekan Deposit Member

Objek Uji : Kebutuhan no. 6

Tujuan Pengujian : Pengujian dilakukan untuk memastikan bahwa aplikasi dapat memenuhi kebutuhan fungsional

untuk melihat sisa deposit dari Member.

Prosedur Uji : 1. Login sebagai Member.
2. Meilih menu “Status Pembayaran”.

Hasil yang diharapkan : Aplikasi dapat menampilkan informasi sisa deposit dari Member dan tanggal terakhir Member melakukan pembayaran.

g. Kasus Uji Konfirmasi Pembayaran

Nama Kasus Uji : Kasus Uji Konfirmasi Pembayaran

Objek Uji : Kebutuhan no. 7

Tujuan Pengujian : Pengujian dilakukan untuk memastikan bahwa aplikasi dapat memenuhi kebutuhan fungsional kepada Member untuk memberikan/memasukkan konfirmasi kepada Administrator, bahwa Member telah melakukan pembayaran.

Prosedur Uji : 1. Login sebagai Member.
2. Pada menu Member, pilih ”Konfirmasi Pembayaran”.
3. Isi field ”Tanggal Pembayaran”, ”Jumlah Pembayaran” dan ”No Rekening Pengirim”.
4. Tekan tombol ”OK”.

Hasil yang diharapkan : Aplikasi dapat menyimpan data konfirmasi pembayaran yang telah dimasukkan oleh Member ke dalam database.

h. Kasus Uji Pengubahan Profil Administrator

Nama Kasus Uji : Kasus Uji Pengubahan Profil Administrator

Objek Uji : Kebutuhan no. 8

Tujuan Pengujian : Pengujian dilakukan untuk memastikan bahwa aplikasi dapat memenuhi kebutuhan fungsional untuk mengubah profil Administrator.

Prosedur Uji : 1. Login sebagai Administrator.
2. Menekan menu “Akun” pada antarmuka utama

Administrator.

3. Mengisi *field* “Nama Lengkap”, “Alamat”, “Email”, “Password”, dan “Konfirmasi Password” kemudian menekan tombol “Ubah”.

Hasil yang diharapkan : Aplikasi dapat menyimpan perubahan akun Administrator dan menampilkan informasi bahwa perubahan akun berhasil dilakukan.

i. Kasus Uji Pengaturan Data Member

Nama Kasus Uji : Kasus Uji Pengaturan Data Member

Objek Uji : Kebutuhan no. 9

Tujuan Pengujian : Pengujian dilakukan untuk memastikan bahwa aplikasi dapat memenuhi kebutuhan fungsional untuk mengatur data Member, diantaranya adalah untuk melihat konfirmasi bayar dari Member, melihat data tanggungan Member, data calon Member dan data Member.

Prosedur Uji :

1. Login sebagai Administrator.
2. Menekan menu “Member” pada antarmuka utama Administrator.
3. Memilih menu ”Lihat Konfirmasi Pembayaran Member”, ”Lihat Tanggungan Member”, ”Data Calon Member Baru” atau ”Data Member”.

Hasil yang diharapkan : Aplikasi dapat menampilkan informasi konfirmasi pembayaran dari Member, data tanggungan Member, data calon Member atau data Member.

j. Kasus Uji Pengaturan Data Informasi Lowongan Kerja

Nama Kasus Uji : Kasus Uji Pengaturan Data Informasi Lowongan Kerja

Objek Uji : Kebutuhan no. 10

Tujuan Pengujian : Pengujian dilakukan untuk memastikan bahwa aplikasi dapat memenuhi kebutuhan fungsional

untuk mengatur data lowongan pekerjaan, diantaranya untuk melihat data RSS terbaru, data RSS yang sudah tersimpan, memasukkan info lowongan kerja secara manual dan melihat info lowongan kerja hasil input manual.

Prosedur Uji : 1. Login sebagai Administrator.
2. Menekan menu “Loker” pada antarmuka utama Administrator.
3. Memilih menu “Data RSS Baru” atau “Data RSS Loker Tersimpan”, “Input Loker Manual”, “Lihat Loker Hasil Input Manual”.

Hasil yang diharapkan : Aplikasi dapat menampilkan data RSS lowongan kerja terbaru, atau menampilkan data RSS lowongan kerja yang telah tersimpan, memasukkan data lowongan kerja dan melihat data lowongan kerja hasil input manual.

k. Kasus Uji Pengiriman Informasi Lowongan Kerja

Nama Kasus Uji : Kasus Uji Pengiriman Informasi Lowongan Kerja

Objek Uji : Kebutuhan no. 11

Tujuan Pengujian : Pengujian dilakukan untuk memastikan bahwa aplikasi dapat memenuhi kebutuhan fungsional untuk mengirimkan informasi lowongan kerja kepada Member melalui SMS.

Prosedur Uji : 1. Login sebagai Administrator.
2. Menekan menu “Kirim SMS” pada antarmuka utama Administrator.
3. Memilih “Kualifikasi Akademik” dan “Jenis Loker”, kemudian menekan tombol “Cari”.
4. Memilih link “Kirim Per member” atau “Kirim Semua”
5. Jika memilih “Kirim Per Member”, kemudian pilih Member yang akan dikirimi SMS dan

kemudian kirim

Hasil yang diharapkan : Aplikasi dapat mengirimkan info lowongan kerja ke Member ke SMPP Server.

l. Kasus Uji Manajemen Artikel

Nama Kasus Uji : Kasus Uji Manajemen Artikel

Objek Uji : Kebutuhan no. 12

Tujuan Pengujian : Pengujian dilakukan untuk memastikan bahwa aplikasi dapat memenuhi kebutuhan fungsional untuk mengatur data artikel berupa kabar dan tips.

Prosedur Uji :
1. Login sebagai Administrator.
2. Menekan menu “Artikel” pada antarmuka utama Administrator.
3. Memilih “Input Artikel”, “Lihat Kabar” atau “Lihat Tips”
4. Jika memilih “Input Artikel”, masukkan field “Kategori”, “Judul” dan “Isi”, kemudian tekan tombol “Simpan”.

Hasil yang diharapkan : Aplikasi dapat menyimpan dan menampilkan data artikel (kabar dan tips).

m. Kasus Uji Pencetakan Laporan

Nama Kasus Uji : Kasus Uji Pencetakan Laporan

Objek Uji : Kebutuhan no. 13

Tujuan Pengujian : Pengujian dilakukan untuk memastikan bahwa aplikasi dapat memenuhi kebutuhan fungsional untuk mencetak laporan *(report)*.

Prosedur Uji :
1. Login sebagai Administrator.
2. Pilih ”Jenis Report” dan ”Periode”.

Hasil yang diharapkan : Aplikasi dapat menghasilkan *report* dalam format PDF yang bisa disimpan atau dicetak.

n. Kasus Uji Logout

Nama Kasus Uji : Kasus Uji Logout

Objek Uji	:	Kebutuhan no. 14
Tujuan Pengujian	:	Pengujian dilakukan untuk memastikan bahwa aplikasi dapat memenuhi kebutuhan fungsional untuk logout bagi Member ataupun bagi Administrator.
Prosedur Uji	:	1. Untuk Member, login sebagai Member kemudian pilih menu ”Logout”. 2. Untuk Administrator, login sebagai Administrator dan pilih menu “Logout”.
Hasil yang diharapkan	:	Aplikasi dapat melakukan logout dari status Member dan Administrator.

6.1.5.2 Hasil Pengujian Validasi

Dari kasus uji yang telah dilaksanakan sesuai dengan prosedur pengujian pada sub pokok bahasan 6.1.5.1, didapatkan hasil seperti ditunjukkan pada Tabel 6.5.

Tabel 6.5 Hasil pengujian validasi

No	Kasus Uji	Hasil yang didapatkan	Status
1	Kasus Uji Registrasi	Aplikasi dapat menyimpan data member yang telah dimasukkan, dan memberikan notifikasi terhadap member yang telah memasukkan datanya, bahwa data tersebut telah berhasil disimpan.	Valid
2	Kasus Uji Login Member	Aplikasi dapat melakukan validasi nilai *username* dan *password* yang diisi pengguna dan menampilkan menu yang bisa digunakan oleh Member.	Valid
3	Kasus Uji Login Administrator	Aplikasi dapat melakukan validasi Adminname dan Password yang dimasukkan, dan menampilkan halaman Administrator.	Valid
4	Kasus Uji Penampilan Informasi Lowongan Kerja	Aplikasi dapat menampilkan berbagai informasi lowongan kerja yang berasal dari data RSS (jika dipilih “Daftar Loker RSS Baru”) atau dari hasil input manual (jika dipilih “Daftar Loker Hasil Input Manual”).	Valid
5	Kasus Uji Pengubahan Profil Member	Aplikasi dapat menyimpan perubahan data profil Member dan memberikan notifikasi kepada Member bahwa profil sudah berhasil diubah.	Valid

6	**Kasus Uji Pengecekan Deposit Member**	**Aplikasi dapat menampilkan informasi sisa deposit dari Member dan tanggal terakhir Member melakukan pembayaran.**	**Valid**
7	**Kasus Uji Konfirmasi Pembayaran**	**Aplikasi dapat menyimpan data konfirmasi pembayaran yang telah dimasukkan oleh Member ke dalam database.**	**Valid**
8	**Kasus Uji PengubahanProfile Administrator**	**Aplikasi dapat menyimpan perubahan akun Administrator dan menampilkan informasi bahwa perubahan akun berhasil dilakukan.**	**Valid**
9	**Kasus Uji Pengaturan Data Member**	**Aplikasi dapat menampilkan informasi konfirmasi pembayaran dari Member, data tanggungan Member, data calon Member atau data Member.**	**Valid**
10	**Kasus Uji Pengaturan Data Informasi Lowongan Kerja**	**Aplikasi dapat menampilkan data RSS lowongan kerja terbaru, atau menampilkan data RSS lowongan kerja yang telah tersimpan, memasukkan data lowongan kerja dan melihat data lowongan kerja hasil input manual.**	**Valid**
11	**Kasus Uji Pengiriman Informasi Lowongan Kerja**	**Aplikasi dapat mengirimkan info lowongan kerja ke Member ke SMPP Server.**	**Valid**
12	**Kasus Uji Manajemen Artikel**	**Aplikasi dapat menyimpan dan menampilkan data artikel (kabar dan tips).**	**Valid**
13	**Kasus Uji Pencetakan Laporan**	**Aplikasi dapat menghasilkan *report* dalam format PDF yang bisa disimpan atau dicetak.**	**Valid**
14	**Kasus Uji *Logout***	**Aplikasi dapat melakukan logout dari status Member dan Administrator.**	**Valid**

Sumber : Pengujian

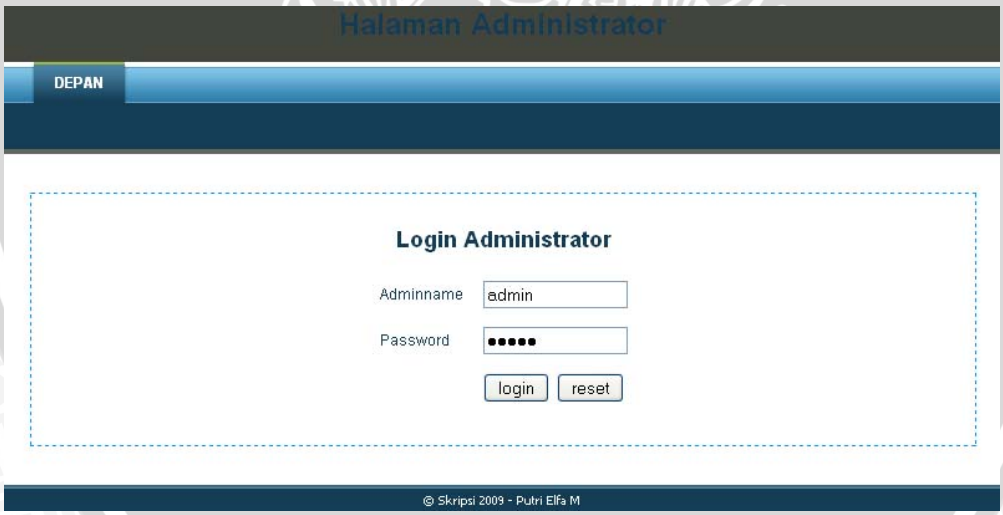
6.1.6 Pengujian Proses Login

Proses pengujian login dilakukan untuk memastikan tidak ada *user* yang login dua kali dalam waktu yang sama. Dalam proses pengembangan telah dirancang sistem hanya bisa menerima login dari *user* hanya sekali dalam satu waktu. Ketika status login dari suatu *user* masih aktif dan belum *logout*, maka *user* tersebut tidak bisa login, meskipun berasal dari komputer yang berbeda.

Pada proses pengujian login ini digunakan duua buah komputer dengan alamat IP masing-masing adalah 192.168.1.3 dan 192.168.1.4. Sedangkan aplikasi dideploy pada komputer 192.168.1.3.

a. Pengujian Login Admin

Pengujian dilakukan dengan melakukan proses login dengan *user* `admin` dua kali dalam satu waktu. Pada komputer dengan alamat IP 192.168.1.3 dilakukan proses login dengan *user* `admin`. Proses login ditunjukkan pada Gambar 6.19

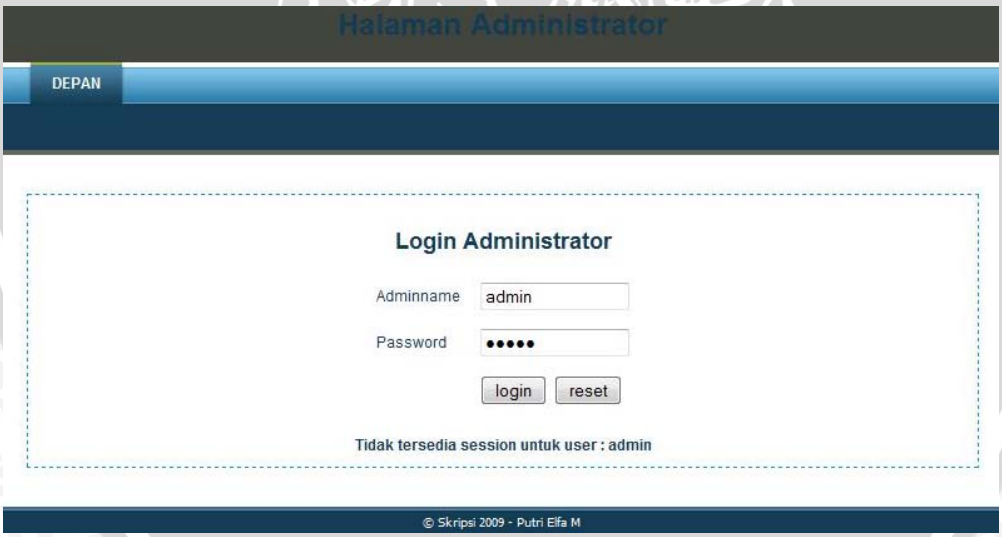


Gambar 6.19 Proses Login `Admin`
Sumber: Pengujian

Proses login berhasil dan berhasil masuk ke halaman administrator, seperti ditunjukkan pada Gambar 6.20

Gambar 6.20 Halaman Administrator
Sumber: Pengujian

Prosedur pengujian login selanjutnya adalah melakukan login dengan user `admin` yang dilakukan melalui komputer yang berbeda yaitu komputer 192.168.1.4. Proses login ini berusaha dilakukan sedangkan dalam waktu yang sama user `admin` masih berada pada status login pada komputer 192.168.1.3. Hasil dari proses login yang dilakukan pada komputer 192.168.1.4 ditunjukkan pada Gambar 6.21



Gambar 6.21 Halaman Login Administrator
Sumber: Pengujian

Dari Gambar 6.21 terlihat bahwa proses login yang dilakukan dari komputer 192.168.1.4 tidak bisa dilakukan.

b. Pengujian Login Member

Pada proses pengujian ini digunakan dua buah member yaitu `putri` dan `salma`. Pada komputer 192.168.1.3 dilakukan proses login member dengan akun member `putri`. Pada Gambar 6.22 ditunjukkan proses login dengan akun `putri` berhasil dilakukan.

Gambar 6.22 Halaman Member `putri`
Sumber: Pengujian

Di komputer 192.168.1.4 dilakukan proses login dengan menggunakan akun member yang sama yaitu putri, sedangkan pada saat yang sama member putri masih berada pada status login pada komputer 192.168.1.3. Gambar 6.23 menunjukkan hasil proses login dari komputer 192.168.1.4 dengan menggunakan member putri

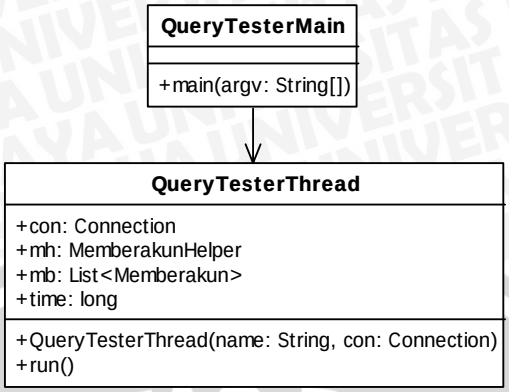
Gambar 6.23 Halaman Login Member
Sumber: Pengujian

Kemudian dilakukan proses login dengan menggunakan akun member yang berbeda yaitu `salma`. Proses login dilakukan dari komputer 192.168.1.4, dan hasilnya berhasil seperti yang ditunjukkan pada Gambar 6.24

Gambar 6.24 Halaman Member `salma`
Sumber: Pengujian

6.1.7 Pengujian Waktu Proses *Query*

Pengujian ini dilakukan untuk mengetahui waktu yang dibutuhkan untuk melakukan proses olah query pada *database*. Pada proses pengujian ini digunakan klas *dummy* `QueryTesterThread` yang merupakan subklas dari klas `Thread`. Klas `QueryTesterThread` digunakan untuk mengimplementasikan *multithread*, dimana tiap *thread* melakukan proses *query* terhadap *database*. Tabel yang digunakan untuk proses pengujian ini adalah tabel `memberakun`. Diagram klas untuk `QueryTesterThread` ditunjukkan pada Gambar 6.25



Gambar 6.25 Diagram Klas `QueryTesterMain` dan `QueryTesterThread`
Sumber: Pengujian

Pada proses pengujian proses *query* ini, *database* diletakkan pada *server* dengan spesifikasi *hardware* seperti diberikan pada Tabel 6.6

Tabel 6.6 Spesifikasi *database server* untuk pengujian *query*

Spesifikasi Komputer	
Prosesor	Intel(R) Pentium(R) Dual CPU T2390 @ 1.86 GHz
Memori (RAM)	1,99 GB
Hardisk	Hitachi , kapasitas 160 GB

Sumber: Pengujian

Variabel yang digunakan dalam proses pengujian ini adalah jumlah proses atau *thread* yang melakukan proses *query* secara bersamaan serta waktu yang dibutuhkan untuk proses *query*. Hasil proses uji ditampilkan pada Tabel 6.7

Tabel 6.7 Data percobaan waktu proses *query*

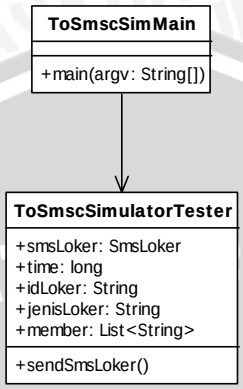
Jumlah *Thread*	**Waktu Rata-Rata Proses *Query* (*milisecond*)**
5	2615,6
10	2998,5
40	3735,6
100	6312,1
500	8966,9
1000	16.577,9
1500	23.247,9

Sumber: Pengujian

6.1.8 Pengujian Waktu Pengiriman SMS ke SMSC Simulator

Pengujian ini dilakukan untuk mengetahui waktu yang dibutuhkan untuk melakukan *submit* pesan SMS ke SMSC Simulator. Pada proses pengujian ini

digunakan klas *dummy* `ToSmscSimulatorTester` dan klas `ToSmscSimMain`. Diagram klas untuk `ToSmscSimulatorTester` dan klas `ToSmscSimMain` ditunjukkan pada Gambar 6.26



Gambar 6.26 Diagram Klas `ToSmscSimulatorTester` dan `ToSmscSimMain`
Sumber: Pengujian

Data hasil pengujian waktu yang diperlukan untuk *submit* pesan ke SMSC Simulator diberikan pada Tabel 6.8

Tabel 6.8 Data percobaan waktu *sumbit* SMS ke SMSC Simulator

Jumlah SMS	Waktu (*milisecond*)
10	6562
50	9078
100	11.891
150	15.328
200	19.250
250	22.359

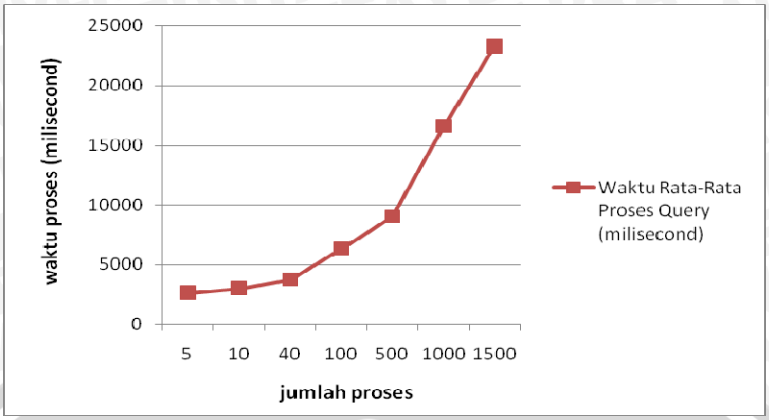
Sumber: Pengujian

6.2 Analisis Waktu Proses

Analisis dilakukan pada setiap data yang didapatkan dari proses pengujian. Analisis pada waktu proses ditujukan untuk mengetahui waktu proses dan faktor-faktor yang berpengaruh terhadapnya.

6.2.1. Analisis Waktu Proses Query Database

Data hasil pengujian yang dicantumkan pada Tabel 6.7 merupakan data waktu yang dibutuhkan oleh aplikasi untuk melakukan proses *query*. Grafik hasil pengujian waktu proses *query* ditunjukkan pada Gambar 6.27

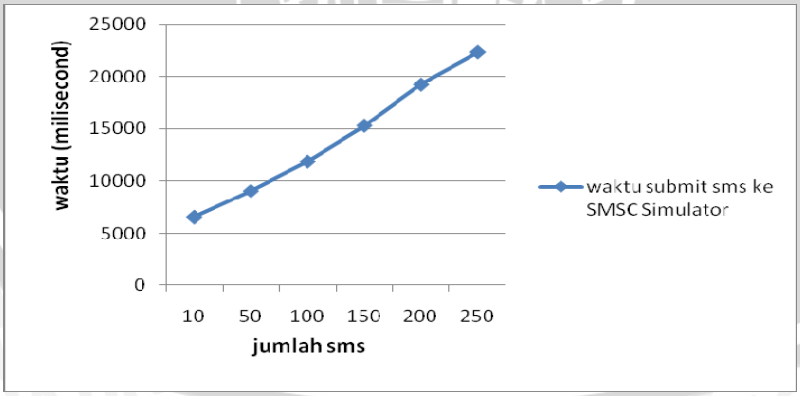


Gambar 6.27 Grafik Waktu Proses *Query*
Sumber: Pengujian

Dari grafik dalam Gambar 6.27 dapat dilihat sebuah pola peningkatan waktu yang dibutuhkan untuk melakukan proses *query* terhadap suatu tabel di database. Hal ini disebabkan karena dengan semakin banyaknya *request* proses ke suatu *database server*, maka *database server* membutuhkan *effort* yang lebih besar untuk melayani permintaan tersebut**.** Dan hal ini berdampak pada waktu yang dibutuhkan untuk melayani permintaan tersebut.

6.2.2. Analisis Waktu Proses Pengiriman SMS ke SMSC Simulator

Dari data hasil pengujian yang dicantumkan pada Tabel 6.8, jika digambarkan dalam bentuk grafik akan tampak seperti Gambar 6.28



Gambar 6.28 Grafik waktu proses *submit* SMS ke SMSC Simulator
Sumber: Pengujian

Dari grafik dalam Gambar 6.28 Terlihat bahwa waktu untuk melakukan proses *submit* SMS ke SMSC Simulator akan semakin meningkat seiring dengan bertambahnya jumlah SMS yang akan dikirimkan (waktu *submit* SMS ke SMSC Simulator sebanding dengan jumlah SMS).

BAB VII
PENUTUP

7.1 Kesimpulan

Berdasarkan hasil perancangan, implementasi, dan pengujian yang telah dilakukan maka dapat diambil kesimpulan sebagai berikut:

1. Hasil pengujian *E-R Diagram* menunjukkan bahwa tabel pada basis data hasil implementasi pada aplikasi ini sesuai dengan tabel pada basis data hasil perancangan.
2. Hasil pengujian koneksi basis data dan *web server* menunjukkan bahwa koneksi *server* basis data MySQL dan *web server Glassfish* yang terletak dalam sebuah jaringan LAN *(Local Area Network)* dapat dilakukan sehingga hubungan *client server* dapat berjalan.
3. Hasil pengujian unit dan pengujian integrasi menunjukkan bahwa sistem dapat melakukan proses manajemen member dan manajemen lowongan kerja dengan baik dan sesuai dengan hasil perancangan. Adapun proses manajemen tersebut adalah mengambil data RSS dari *website* lain, meng-*update* deposit member dan mengirimkan informasi lowongaan kerja lewat sms.
4. Berdasarkan analisis waktu proses query database menunjukkan bahwa semakin banyaknya *request* proses ke suatu *database server*, maka *database server* membutuhkan *effort* yang lebih besar untuk melayani permintaan tersebut, hal ini berdampak pada waktu yang dibutuhkan untuk melayani permintaan tersebut.
5. Berdasarkan analisis waktu proses pengiriman sms ke smsc simulator menunjukkan bahwa waktu untuk melakukan proses *submit* SMS ke SMSC Simulator akan semakin meningkat seiring dengan bertambahnya jumlah SMS yang akan dikirimkan (waktu *submit* SMS ke SMSC Simulator sebanding dengan jumlah SMS).

7.2 Saran

Mengacu dari kelemahan yang terdapat pada sistem yang telah dikembangkan dalam skripsi ini, beberapa saran yang dapat diberikan diantaranya:

1. Agar sistem lebih menarik untuk digunakan, maka desain antarmuka dapat dibuat lebih menarik dan variatif.
2. Pada sistem ini administrator harus bekerja keras dengan memasukkan informasi lowongan kerja yang tidak ada fasilitas rss-nya secara manual. Agar aplikasi menjadi lebih handal maka dapat ditambahkan fitur agar perusahaan dapat bergabung menjadi member untuk menginformasikan kebutuhan lowongan kerja sehingga meringankan pekerjaan administrator
3. Sistem ini masih belum memiliki tingkat keamanan yang baik, oleh karena itu keamanan sistem dapat lebih ditingkatkan.
4. Untuk pengembangan selanjutnya diharapkan sistem ini bisa benar-benar terealisasi secara nyata, sistem dapat terkoneksi secara langsung ke SMSC (tidak menggunakan *SMSC simulator*) sehingga informasi lowongan kerja dapat dikirimkan ke *handphone* member

DAFTAR PUSTAKA

[AYU-09] Ayuliana, 2009. **Testing Dan Implementasi.** http://ayuliana_st.staff.gunadarma.ac.id

[BOD-05] Bodic, Le Gwenael**.** 2005. **Mobile Messaging Technologies and Services**. England. John Willey and Sons, Ltd.

[CHO-05] Chopra, Vivek. Li, Sing. Jones, Rupert. 2005. **Beginning JavaServer Pages**. Canada. Wiley Publishing

[CON-04] Converse, Tim. Park, Joyce. Morgan, Clark. 2004. **PHP5 and mySQL Bible**. Kanada. Willey Publishing, Inc.

[DHA-03] Dharwiyanti, Sri,Wahono, Romi Satria. 2003, "Pengantar UML", akses dari :http://www.ilmukomputer.org/wp-content/uploads/2006/08/yanti-uml.zip, tanggal akses: 28 Oktober 2008

[DYK-05] Dykes, Lucinda. Tittel, Ed. 2005. **XML for Dummies**. Indianapolis. Willey Publishing, Inc

[FIN-05] Finkelstein, Ellen. 2005.**Syndicating Web Sites with RSS Feeds for Dummies**. Indianapolis. Willey Publishing, Inc

[KEY-05] Keyes, Jessica. 2005 **Software Engineering Handbook**. Boca Raton. Auerbach Publishing, Inc

[NUG-04] Nugroho, Bunafit. 2004. **PHP dan MySQL dengan Editor Dreamweaver MX**. Yogyakarta. Penerbit Andi

[PER-04] Perry, W. Bruce. 2004. **Java Servlet and JSP Cookbook**. 2004. Amerika. O'Rielly Media, Inc

[PIL-05] Pilone, Dan. Pitman, Neil. 2005. **UML 2.0 in a Nutshell**. United State Of America. O'Reilly Media Publishing, Inc.

[POW-05] Powers, Shelley. 2005. **What are Syndication Feeds**. Amerika. O'Rielly Media, Inc.

[PRE-02] Preeman, S. Roger, 2002, "Rekayasa Perangkat Lunak: Pendekatan Praktisi (buku 2). 2002. Penerbit Andi.

[SMI-04] Smith, Clint. Collins, Daniel. 2004. **3G Wireless Network**. Singapore. McGraw-Hill

[SMP-07] SMPP Developer Forum. 2007. **Short Message Peer to Peer Protocol Spesification v3.4.** http://smsforum.net/

[SSS-04] Intronik 2004, **ServerSMS-SMPP**. http://www.intronik.com/sms_sw_smpp.htm

[SPI-03] Spielman, Sue. 2003. **The Struts framework**. United States Of America. Morgan Kaufmann Publishers.

[WIJ-07] Wijono, Hartati Sri. Suharto, Herry B., Wijono, Soesilo Matius. 2007. **Pemrograman Java Servlet dan JSP dengan Netbeans**. Yogyakarta. Penerbit Andi