SISTEM INFORMASI PENDAFTARAN PASIEN DI POLIKLINIK BERBASIS SMS

(Studi Kasus : Poliklinik Universitas Islam Indonesia)

LAPORAN TUGAS AKHIR

Diajukan Sebagai Salah Satu Syarat
Untuk Memperoleh Gelar Sarjana
Jurusan Teknik Informatika

oleh:

Erita Yuliastuti
03523220

JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INDUSTRI
UNIVERSITAS ISLAM INDONESIA
YOGYAKARTA

2007

LEMBAR PENGESAHAN PEMBIMBING

SISTEM INFORMASI PENDAFTARAN PASIEN DI POLIKLINIK BERBASIS SMS

(Studi Kasus : Poliklinik Universitas Islam Indonesia)

TUGAS AKHIR

Oleh :

Nama : Erita Yuliastuti
No. Mahasiswa : 03523220

Yogyakarta, 14 Juni 2007
Menyetujui,
Pembimbing I,

Sri Kusumadewi, Hj, S.Si., MT.

LEMBAR PENGESAHAN PENGUJI

SISTEM INFORMASI PENDAFTARAN PASIEN DI POLIKLINIK BERBASIS SMS

(Studi Kasus : Poliklinik Universitas Islam Indonesia)

TUGAS AKHIR

Oleh :

Nama : Erita Yuliastuti

No. Mahasiswa : 03523220

Telah Dipertahankan di Depan Sidang Penguji sebagai Salah Satu Syarat untuk Memperoleh Gelar Sarjana Jurusan Teknik Informatika Fakultas Teknologi Industri Universitas Islam Indonesia

Yogyakarta, 27 Juni 2007

Tim Penguji,
Sri Kusumadewi S Si., MT.
Ketua

Nur Wijayaning Rahayu, S.Kom.
Anggota I

Hendrik ST.
Anggota II

Mengetahui,
Ketua Jurusan Teknik Informatika
Fakultas Teknologi Industri
Universitas Islam Indonesia

Yudi Prayudi, SSi., M.Kom.

HALAMAN PERSEMBAHAN

Skripsi ini kupersembahkan untuk:

Allah SWT dan Nabi Muhammad SAW,
Ayah, Ibu serta saudara yang senantiasa memberiku motivasi,
dukungan serta do'a,
Guru, Pendidik, Pengajar dan Dosen, terimakasih telah
mengajarkanku ilmu yang bermanfaat,
Teman-teman yang tak pernah berhenti memberi dukungan.
Terimakasih

MOTTO

"Kutanamkan di dalamnya mutiara, hingga tiba saatnya ia dapat menyinari tanpa mentari dan berjalan di malam hari tanpa rembukan
Karena kedua matanya ibarat sihir dan keningnya laksana pedang buatan india
Milik Allah-lah setiap bulu mata, leher dan kulit yang mempesona"

'Aidh al-Qarni

"Barangsiapa Kuambil dua kekasihnya (matanya) tetap bersabar, maka Aku akan mengganti kedua (mata)nya itu dengan surga"

Hadist

Sebutlah di dalam hati,
Hasbunallah wa ni'mal wakil,
Hasbunallah wani'mal wakil,
Hasbunallah wani'mal wakil,

"Dan cukuplah Rabb-mu menjadi Pemberi Petunjuk dan Penolong."

QS. Al-Furqan:31)

KATA PENGANTAR

Assalamu 'alaikum Wr. Wb

Segala puji dan syukur penulis panjatkan kehadirat Allah SWT atas segala rahmat dan hidayah-Nya sehingga penulis dapat menyelesaikan Laporan Tugas Akhir sebagai salah satu prasyarat untuk mendapatkan gelar kesarjanaan.

Tugas Akhir merupakan upaya untuk mendidik mahasiswa agar dapat menerapkan ilmu dan kemampuan yang diperoleh dari bangku kuliah untuk diabdikan pada masyarakat. Sehingga mahasiswa dapat berlatih menerapkan ilmu dan kemampuannya pada masyarakat umum. Pada kesempatan ini penulis ingin mengucapkan terima kasih yang sebesar-besarnya kepada :

1. Allah SWT, Muhammad SAW atas seluruh rahmat dan karunia.
2. Kedua orang tua dan saudara atas dukungan dan support penuh dalam menyelesaikan Tugas Akhir.
3. Bapak Fathul Wahid, ST., M.Sc. selaku Dekan Fakultas Teknologi Industri, Universitas Islam Indonesia.
4. Bapak Yudi Prayudi, SSi., M.Kom. selaku Ketua Jurusan Teknik Informatika.
5. Ibu Sri Kusumadewi Hj., S.Si, MT. Selaku dosen pembimbing.
6. Rekan-rekan Jurusan Teknik Informatika angkatan 2003 khususnya Bastian, Opik, Budi, Setyo, Itit, Nia, Nesi, Fyda, Echi, Erwin atas dukungannya.
7. Sahabat-sahabatku, Reni, Ajeng, Titiz, Anisa dan Ririn atas dukungan dan doa tanpa henti.
8. Dan pihak-pihak yang tidak bisa disebutkan satu per satu.

Penulis menyadari sepenuhnya akan keterbatasan kemampuan yang dimiliki, oleh karena itu segala kritik dan saran yang bersifat membangun diterima dengan senang hati. Semoga laporan Tugas Akhir ini dapat bermanfaat bagi semua pihak yang membutuhkan sebagai referensi.

Billahitaufik walhidayah, Wassalamu 'alaikum Wr. Wb.

Yogyakarta, 27 Juni 2007

Penulis

SARI

Materi Tugas Akhir yang dilaksanakan adalah membangun sebuah aplikasi desktop Sistem Informasi Pendaftaran Pasien di Poliklinik Berbasis SMS dengan menggunakan bahasa pemrograman visual dan diintergrasikan menggunakan DBMS atau *Database Management System.* Untuk itu penulis menggunakan Microsoft Visual Basic 6.0 dan Java 2 SDK untuk editor sekaligus *compiler* program dan untuk DBMS-nya digunakan Microsoft SQL Server 7.0.

Proses perancangan sistem menggunakan UML. Sedangkan langkah penyelesaian pembangunan sistem dipergunakan metodologi GRAPPLE. Dari Segmen-segmen yang dimiliki GRAPPLE diperoleh kebutuhan-kebutuhan sistem. Dan tentunya dengan sistem yang secara keseluruhan efektif dan aman yaitu dengan menerapkan authentifikasi pengguna. Pengguna yang akan menggunakan sistem dibagi menjadi 3 yaitu Admin, Petugas Pendaftaran dan Dokter, yang masing-masing mempunyai hak akses yang berbeda-beda. Dari segmen-segmen dalam GRAPPLE juga diperoleh rancangan antarmuka yang diharapkan lebih user friendly dan easy to use.

Pada bagian akhir laporan dituliskan kesimpulan pelaksanaan dan pembuatan laporan materi Tugas Akhir disertai saran pengembangan sistem untuk ke depannya.

Keyword : Pendaftaran pasien, SMS, Poliklinik, UML, GRAPPLE.

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR MODUL

BAB I

PENDAHULUAN

1.1 Latar Belakang

Perkembangan zaman semakin maju dan kehidupan manusia pun semakin kompleks, karena itu keberadaan informasi yang cepat dan mudah diakses merupakan hal yang sangat penting dan diharapkan oleh masyarakat.

Wireless device seperti telepon gengam adalah salah satu teknologi komunikasi terbaru yang sudah tidak asing lagi bagi masyarakat. Telepon genggam sudah tidak dianggap sebagai barang mewah lagi, melainkan sudah menjadi kebutuhan bagi setiap orang.

Sebagai perangkat komunikasi, telepon genggam menawarkan banyak keunggulan. Bentuknya yang kecil dan ringan serta kepraktisannya yang mudah dibawa kemana saja membuat telepon genggam begitu digemari dan mengalami perkembangan yang pesat. Berbagai fitur yang tersedia dalam perangkat ini menawarkan berbagai kemudahan bagi pengguna telepon genggam tersebut.

Salah satu kemudahan yang ditawarkan perangkat telepon genggam adalah layanan *Short Message Service* (SMS). Fasilitas tersebut memungkinkan pengguna untuk dapat mengirim dan menerima pesan singkat dalam waktu yang relatif cepat.

Dengan adanya perkembangan teknologi khususnya teknologi informasi berbasis SMS, memungkinkan untuk membuat suatu sistem informasi yang cepat dan mudah diakses sebagai media interaktif dan komunikatif antara sesama pengguna sisten informasi dalam bertukar dan memanfaatkan informasi. Dengan berkembang pesatnya teknologi informasi dengan menggunakan SMS, semua keterbatasan sarana, jarak dan waktu transaksi dapat teratasi dengan mudah. Hal ini disebabkan karena teknologi informasi berbasis SMS bisa digunakan oleh

siapa saja dan bersifat dinamis bahkan *real-time* sehingga dapat memberikan informasi dengan cepat dan akurat.

Poliklinik merupakan suatu unit usaha kerjasama dalam bidang kesehatan. Sampai saat ini, semua kegiatan pendaftaran pasien masih menggunakan cara manual yaitu dengan menggunakan kertas bertabel atau catatan-catatan diatas kertas sehingga diperlukan waktu yang cukup lama untuk melakukan pencarian data dan akan timbul kesulitan jika terjadi kehilangan data. Untuk meningkatkan efisiensi waktu dan mempermudah pengaksesan data dalam proses pendaftaran pasien di poliklinik tersebut perlu dibuat sebuah sistem informasi yang mampu mengolah data diri dan proses pendaftaran pasien secara cepat, akurat dan efisien.

Penerapan komputerisasi mendukung adanya penyelesaian terhadap masalah yang dihadapi dan hasilnya juga dirasakan lebih efektif dan efisien karena dengan adanya sistem informasi berbasis komputer akan dapat dengan cepat melayani pendaftaran pasien yang akan diperiksa di polikinik tersebut. Selain itu, komputer juga dapat menyelesaikan pekerjaan dengan lebih cepat dan kesalahan-kesalahan yang sering terjadi dapat diminimalisir. Terutama dalam proses penyimpanan data, pengolahan data, seperti perhitungan, atau pencarian data tertentu yang kemudian menyajikannya dalam bentuk informasi yang berguna sebagai referensi pengambilan keputusan.

1.2 Rumusan Masalah

Rumusan masalah pada penelitian ini adalah bagaimana membuat sistem informasi pendaftaran pasien di poliklinik secara *realtime* melalui SMS sehingga memudahkan calon pasien dan perawat untuk melakukan registrasi data sebelum pemeriksaan kesehatan dilakukan.

1.3 Batasan Masalah

Adapun batasan masalah yang digunakan pada penelitian ini adalah:

a. *Error handling* pada telepon genggam tidak dibahas dalam Sistem Informasi Pendaftaran Pasien di Poliklinik berbasis SMS yang dibangun.

b. Keamanan data pada saat proses pengiriman pesan tidak di bahas dalam Sistem Informasi ini.

1.4 Tujuan Penelitian

Tujuan dari penelitian ini adalah menghasilkan sistem informasi pendaftaran pasien secara *realtime* melalui SMS sehingga memudahkan calon pasien dan perawat untuk melakukan registrasi data sebelum pemeriksaan kesehatan dilakukan.

1.5 Manfaat Penelitian

Manfaat yang diperoleh dengan adanya sistem pendaftaran pasien di poklinik tersebut, diantaranya :

a. Mengubah sistem pendataan konvensional yang digunakan dalam melakukan pendaftaran pasien menjadi sistem terkomputerisasi.
b. Mempermudah proses pendataan, penyimpanan dan pencarian data kembali.
c. Mengurangi kerusakan serta kehilangan data akibat dari pendataan secara konvensional.
d. Mempermudah proses pendaftaran pasien yang memiliki mobilitas tinggi.
e. Dapat memberikan pelayanan dengan cepat, mudah, efektif dan efisien.

1.6 Metodologi Penelitian

Dalam penelitian ini metode pengembangan perangkat lunak yang digunakan adalah *GRAPPLE* (*Guidlines for Rapid APPLication Engineering*). Metode pengumpulan data dimasukkan dalam salah satu tahap dalam *GRAPPLE* yaitu tahap *requirement gathering* dan akan dijelaskan lebih lanjut pada BAB IV. *GRAPPLE* (*Guidelines for Rapid APPLication Engineering*) terdiri dari lima tahap (Schmuller, 1999) yaitu :

1. *Requirement gathering*
2. *Analysis*
3. *Design*
4. *Development*
5. *Deployment*

Dalam penelitian ini hanya 4 segmen *GRAPPLE* yang akan dikerjakan, segmen penyebaran (*deployment*) tidak akan dikerjakan. Tidak semua aksi dalam setiap segmen *GRAPPLE* akan dikerjakan dalam penelitian ini.

1.7 Sistematika Penulisan

Untuk mempermudah pemahaman isi laporan tugas akhir ini dikemukakan sistematika penulisan laporan agar menjadi suatu kesatuan yang utuh. Selanjutnya laporan tugas akhir ini ditulis dalam 5 Bab, meliputi:

Bab I **PENDAHULUAN**

Pada awal bab dibahas mengenai gambaran umum dari skripsi ini, yang menyajikan Latar Belakang Masalah, Rumusan Masalah, Batasan Masalah, Tujuan Penelitian, Manfaat Penelitian, Metodologi Penelitian dan Sistematika Penulisan.

Bab II **LANDASAN TEORI**

Pada bab ini dibahas mengenai teori yang digunakan sebagai acuan di dalam pembahasan masalah dan mengimplementasikan sistem. Antara

lain pengertian singkat mengenai sistem, informasi, sistem informasi, SMS, *AT Commands*, GRAPPLE, UML, dll.

Bab III METODOLOGI

Pada bab ini memuat uraian tentang langkah-langkah penyelesaian masalah selama melakukan penelitian pada poliklinik. Langkah-langkah yang dibuat sesuai dengan topik penelitian.

Bab IV HASIL DAN PEMBAHASAN

Pada bab ini memuat uraian mengenai hasil dan bagaimana hasil tersebut dicapai dari setiap aktifitas yang dilakukan selama penelitian berlangsung. Pada bagian ini juga disertai pembahasan hasil aktivitas yang diperoleh selama melakukan penelitian. Pembahasan dapat berupa uraian tentang mengapa hasil diperoleh, kelebihan (keunggulan) dan kelemahan penerapan hasil penelitian bagi institusi.

Bab V SIMPULAN DAN SARAN

Bab yang memuat kesimpulan dari permasalahan serta beberapa saran pengembangan untuk menyempurnakan sistem, yang diperoleh dari apa yang telah dihasilkan.

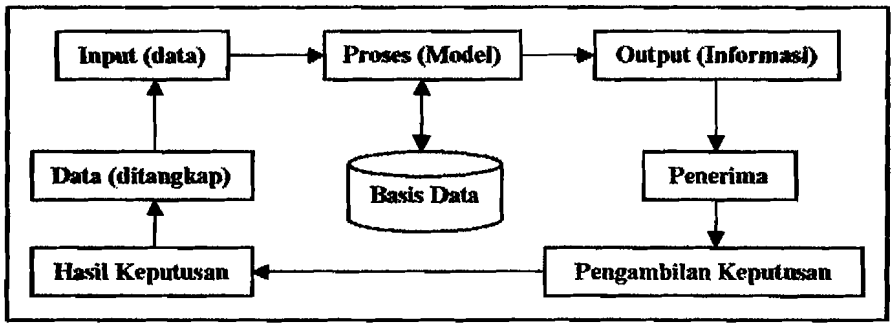
BAB II

LANDASAN TEORI

2.1 Informasi

Informasi dapat didefinisikan sebagai hasil dari pengolahan data dalam suatu bentuk yang lebih berguna dan lebih berarti bagi penerimanya yang menggambarkan suatu kejadian-kejadian (*event*) yang nyata (*fact*) yang digunakan untuk pengambilan keputusan. [Jogiyanto, 1999].

Sumber dari informasi adalah data. Data merupakan bentuk jamak dari bentuk tunggal datum atau *data-item*. Data adalah kenyataan yang menggambarkan suatu kejadian-kejadian dan kesatuan nyata. Kejadian (*event*) adalah sesuatu yang terjadi pada saat yang tertentu. Kesatuan nyata (*fact and entity*) adalah berupa suatu obyek nyata seperti tempat, benda dan orang yang betul-betul ada dan terjadi. Data dinyatakan dengan nilai (angka, deretan karakter, atau simbol). Data merupakan bentuk yang masih mentah, belum dapat bercerita banyak, sehingga perlu diolah lebih lanjut. Data diolah melalui suatu model menjadi informasi, penerima kemudian menerima informasi tersebut, membuat suatu tindakan yang lain yang akan membuat sejumlah data kembali. Data tersebut akan ditangkap sebagi *input*, diproses kembali lewat suatu model dan seterusnya membentuk suatu siklus. Siklus ini oleh Burch disebut dengan siklus informasi (*information cycle*) (Gambar 2.1).



Gambar 2.1 Siklus Informasi

Kualitas dari suatu informasi (*quality of information*) tergantung dari tiga faktor, yaitu keakuratan (*accurate*), ketepatan waktu (*timeliness*) dan kesesuaian (*relevance*) [Kadir, 1999].

a. Keakuratan

Informasi harus bebas dari kesalahan-kesalahan dan tidak bias atau menyesatkan. Akurat juga bisa diartikan informasi harus jelas mencerminkan maksudnya.

b. Ketepatan waktu

Informasi yang datang pada penerima tidak boleh terlambat. Informasi yang sudah usang tidak akan mempunyai nilai karena informasi merupakan landasan pengambilan keputusan sehingga, bila informasi terlambat maka keputusan yang diambil menjadi tidak sesuai dengan keadaan.

c. Kesesuaian

Informasi tersebut mempunyai manfaat untuk pemakainya. Kesesuaian untuk tiap-tiap orang berbeda-beda tergantung dari cara memandang dan memperlakukan informasi yang telah didapatkannya.

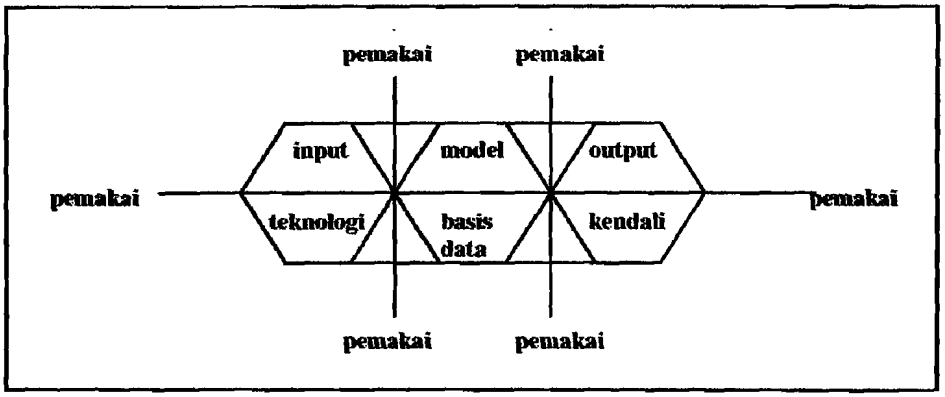
Suatu informasi dikatakan bernilai bila manfaatnya lebih efektif dibandingkan dengan biaya mendapatkannya. Kegunaan informasi adalah untuk mengurangi hal ketidakpastian di dalam proses pengambilan keputusan tentang suatu keadaan.

Telah diketahui bahwa informasi merupakan hal yang sangat penting bagi manajemen di dalam pengambilan keputusan. Informasi dapat diperoleh dari sistem informasi (*information system*) atau disebut juga dengan *processing system* atau *information processing system* atau *information-generating system*.

2.2 Sistem Informasi

Sistem informasi didefinisikan oleh Leitch dan Davis (1983) sebagai suatu sistem di dalam suatu organisasi yang mempertemukan kebutuhan pengolahan transaksi harian, mendukung operasi, bersifat manajerial dan kegiatan strategi dari suatu organisasi dan menyediakan pihak luar tertentu dengan laporan-laporan

yang diperlukan. Sistem informasi terdiri dari komponen yang disebut dengan istilah blok bangunan (*building block*), yaitu blok masukan (*input block*), blok model (*model block*), blok keluaran (*output block*), blok teknologi (*technology block*), blok basis data (*database block*) dan blok kendali (*controls block*). Sebagai suatu sistem, keenam blok tersebut masing-masing saling berinteraksi satu dengan yang lainnya membentuk satu kesatuan untuk mencapai sasarannya [Jogiyanto, 1999]. (Gambar 2.2).



Gambar 2.2 Blok sistem informasi yang berinteraksi

a. Blok masukan (*Input*)

Input mewakili data yang masuk ke dalam sistem informasi. *Input* di sini termasuk metode-metode dan media untuk menangkap data yang akan dimasukkan, yang dapat berupa dokumen-dokumen dasar.

b. Blok model

Blok ini terdiri dari kombinasi prosedur, logika dan model matematis yang akan memanipulasi data *input* dan data yang tersimpan di basis data dengan cara yang sudah tertentu untuk menghasilkan keluaran yang diinginkan.

c. Blok keluaran (*Output*)

Produk dari sistem informasi adalah keluaran yang merupakan informasi yang berkualitas dan dokumentasi yang berguna untuk semua tingkatan manajemen serta semua pemakai sistem.

d. Blok teknologi

Teknologi merupakan kotak alat (*tool-box*) dalam sistem informasi. Teknologi digunakan untuk menerima, menjalankan model, menyimpan dan

mengakses data, menghasilkan dan mengirimkan keluaran dan membantu pengendalian dari sistem secara keseluruhan. Teknologi terdiri dari 3 macam bagian utama, yaitu teknisi (*humanware atau brainware*), perangkat lunak (*software*) dan perangkat keras (*hardware*).

e. Blok basis data

Basis data (*database*) merupakan kumpulan dari data yang saling berhubungan satu dengan yang lainnya, tersimpan di perangkat keras komputer dan digunakan perangkat lunak untuk memanipulasinya. Data perlu disimpan dalam basis data untuk keperluan penyediaan informasi lebih lanjut. Data di dalam basis data perlu diorganisasikan sedemikian rupa, supaya informasi yang dihasilkan berkualitas. Basis data diakses atau dimanipulasi dengan menggunakan perangkat lunak paket yang disebut dengan DBMS (*Database Management System*).

f. Blok kendali

Banyak hal yang dapat merusak sistem informasi, seperti misalnya bencana alam, temperatur air, kegagalan-kegagalan sistem itu sendiri dan lain sebagainya. Beberapa pengendalian perlu dirancang dan diterapkan untuk meyakinkan bahwa hal-hal yang dapat merusak sistem dapat dicegah ataupun bila terlanjur terjadi kesalahan-kesalahan dapat langsung diatasi.

2.3 *Short Message Service* (SMS)

Salah satu kemudahan dari telepon genggam yang dapat dirasakan oleh pemakainya adalah adanya layanan mengirim pesan antar pengguna atau yang biasa disebut dengan *Short Message Service* (SMS).

Aplikasi SMS pada telepon genggam menyediakan layanan untuk mengirim dan menerima pesan pendek berupa huruf dan angka. SMS hanya terbatas pada pengiriman dan penerimaan data berupa teks dengan panjang pesan antara 120-160 huruf [Agung, 2003]. SMS ditangani oleh jaringan melalui pusat pelayanan atau *SMS Center* (SMSC) yang berhubungan dengan PSTN melalui

MSC. SMSC berfungsi untuk menyimpan dan meneruskan pesan dari pengirim ke penerima.

Dalam pengiriman dan penerimaan pesan SMS terdapat dua mode, yaitu mode teks dan mode *Protocol Data Unit* (PDU). Mode teks adalah format pesan dalam bentuk teks asli yang dituliskan pada saat akan mengirim dan menerima pesan. Sedangkan mode PDU adalah format pesan dalam bentuk oktet heksadesimal dan oktet semidesimal dengan panjang mencapai 160 (7 bit) atau 140 (8 bit) karakter.

2.4 *Attention* (AT) *Command*

AT command adalah perintah-perintah yang digunakan dalam komunikasi dengan Serial Port. Dengan AT command kita dapat mengetahui vendor dari telepon genggam yang digunakan, kekuatan sinyal, membaca pesan yang ada pada SIM Card, mengirim pesan, mendeteksi pesan SMS baru yang masuk secara otomatis, menghapus pesan pada SIM Card, dan masih banyak lagi.

AT command digunakan untuk komunikasi antara komputer dengan modem, namun untuk beberapa aplikasi komunikasi yang menggunakan *Graphical User Interface* (GUI), AT command ini tidak akan dimunculkan untuk memudahkan penggunaan oleh user [Agung, 2003].

2.5 *Guidelines for Rappid APPLication Engineering* (GRAPPLE)

Metodologi pengembangan sistem yang digunakan dalam perancangan dan pembuatan perangkat lunak ini adalah GRAPPLE (*Guidelines for Rapid APPLication Engineering*). Metodologi GRAPPLE ditemukan oleh Joseph Schmuller pada tahun 1999, yang kemudian dituliskan pada buku karangannya yang berjudul *Sams Teach Yourself UML in 24 Hours*. Ide utama metodologi GRAPPLE tersebut disari dari beberapa metodologi lain yang telah ditemukan sebelumnya dilengkapi dengan beberapa perbaikan dan penyempurnaan yang diperlukan. Metodologi dapat diperumpamakan sebagai motor penggerak pada

proses pengembangan perangkat lunak dengan UML sebagai bahan bakarnya. GRAPPLE mempunyai tahapan-tahapan yang sederhana dalam melakukan pengembangan aplikasi.

GRAPPLE merupakan kerangka pengembangan yang fleksibel dan memberikan panduan yang jelas dalam proses pengembangan sistem. Metodologi ini terdiri dari lima segmen. Masing-masing segmen terdiri atas beberapa aksi. Masing-masing aksi menghasilkan *work product* dan masing-masing aksi merupakan tanggung jawab dari sebagian pembuat sistem [Schmuller, 1999]. Segmen-segmen tersebut antara lain :

a. Requirements Gathering

Pada tahap pertama yang dilakukan oleh pengembang perangkat lunak adalah mengambil informasi lengkap dari pengguna tentang sistem yang akan dibangun. Dalam penelitian ini, proses pengambilan informasi dilakukan dengan metode wawancara. Wawancara dilakukan langsung dengan pengguna yang menginginkan adanya sistem ini dan dengan pengguna yang berhubungan langsung dengan sistem. Tahap ini menyarankan untuk mewawancarai pengguna yang memiliki kemampuan teknis.

b. Analysis

Di tahap *analysis* yang dilakukan adalah menggali lebih dalam hasil yang diperoleh dalam tahap sebelumnya. Tahap ini akan mengkaji permasalahan pengguna dan menganalisis solusinya.

c. Design

Tahap *design* dilakukan untuk merancang solusi yang dihasilkan pada tahap *analysis*. Tahap *analysis* dan tahap *design* dapat berjalan dua arah saling menyesuaikan sampai diperoleh rancangan yang benar-benar tepat.

d. Development

Tahap ini ditangani oleh pengembang program untuk membangun kode-kode program dan *user interface*. Pengujian program dan dokumentasi sistem juga dilakukan pada tahap ini.

e. Deployment

Tahap *deployment* adalah tahap pendistribusian produk yang dihasilkan kepada pengguna. Tahap ini mencakup instalasi dan perencanaan *backup* data bila diminta oleh pengguna sesuai dengan perjanjian sebelumnya.

2.6 *Unified Modelling Language* (UML)

Unified Modelling Language (UML) merupakan sistem arsitektur yang bekerja dalam OOAD dengan suatu bahasa yang konsisten untuk menentukan visualisasi, mengkonstruksi dan mendokumentasikan *artifact* yang terdapat dalam sistem perangkat lunak [Munawar, 2005]. Didalam UML terdapat beberapa diagram dan notasi yang digunakan.

Diagram-diagram yang digunakan dalam UML adalah sebagai berikut :

a. *Use Case Diagram*

Use case diagram menjelaskan manfaat sistem jika dilihat menurut pandangan orang yang berada di luar sistem atau *actor*. Diagram ini menunjukkan fungsionalitas suatu sistem atau kelas dari bagaimana sistem berinteraksi dengan dunia luar.

b. Sequence Diagram

Sequence diagram menggambarkan interaksi antar obyek didalam maupun diluar sistem biasanya digunakan untuk menggambarkan skenario atau rangkaian langkah-langkah yang dilakukan sebagai tanggapan dari sebuah kejadian untuk menghasilkan sebuah keluaran (output) tertentu.

c. Class Diagram

Class diagram menunjukkan deskripsi dan struktur dari *class, package* dan obyek beserta hubungan satu dengan yang lainnya. Obyek merupakan hasil dari sebuah instansiasi dari sebuah *class*.

d. StateChart Diagram

StateChart diagram menggambarkan transisi dan perubahan keadaan suatu obyek pada sistem sebagai akibat dari *message* yang diterima. Keadaan-keadaan yang dialami obyek akibat mendapat *message* digambarkan dalam bentuk *state*.

e. Activity Diagram

Activity diagram menggambarkan berbagai aliran aktivitas dalam sistem yang sedang dirancang, bagaimana awal dari masing-masing aliran, percabangan yang mungkin terjadi serta bagaimana akhirnya.

f. Collaboration Diagram

Collaboration diagram menggambarkan interaksi antar obyek seperti *sequence diagram*. Perbedaan antara kedua diagram tersebut adalah *Collaboration diagram* lebih menekankan pada peran masing-masing obyek dan bukan pada waktu penyampaian *message*.

g. Component Diagram

Component diagram menggambarkan struktur dan hubungan antar komponen perangkat lunak, termasuk *dependency* (ketergantungan) diantaranya. Diagram ini juga menggambarkan alokasi semua kelas dan obyek kedalam komponen-komponen perangkat lunak.

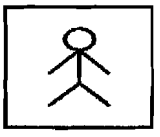
h. Deployment Diagram

Deployment diagram menggambarkan detail bagaimana komponen di-*deploy* dalam infrastruktur sistem, dimana komponen akan terletak (mesin, *server* atau perangkat keras), bagaimana kemampuan jaringan pada lokasi tersebut, spesifikasi *server* dan hal-hal lain yang bersifat fisik. Diagram ini memperlihatkan pemetaan perangkat lunak (*software*) kepada perangkat keras (*hardware*).

Notasi-notasi yang digunakan didalam UML untuk membangun diagram adalah sebagai berikut :

a. Actor

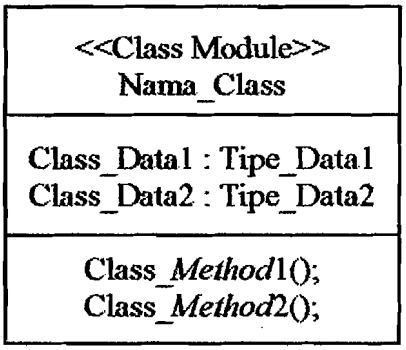
Actor menggambarkan pengguna perangkat lunak aplikasi yang dapat membantu memberikan suatu gambaran jelas tentang apa yang harus dikerjakan perangkat lunak aplikasi [Munawar, 2005]. *Actor* bisa berupa orang, perangkat keras, ataupun obyek lain yang ada dalam sistem. (Gambar 2.3).



Gambar 2.3 Notasi *Actor*

b. Class

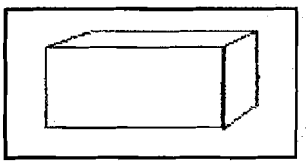
Class adalah definisi umum (pola, *template*, atau cetak biru) untuk himpunan obyek sejenis [Munawar, 2005]. *Class* menetapkan spesifikasi perilaku dan atribut obyek. (Gambar 2.4).



Gambar 2.4 Notasi Class

c. Node

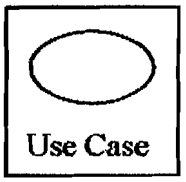
Node adalah bagian utama *hardware*/perangkat keras yang digambar dalam *deployment diagram*. Ada dua tipe *node* yaitu *processor* dan *device*. Notasi *node* dapat dilihat pada gambar 2.5.



Gambar 2.5 Notasi *Node*

d. Use Case

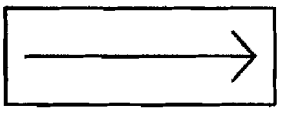
Use case menjelaskan urutan kegiatan yang dilakukan *actor* dan sistem untuk mencapai suatu tujuan tertentu. Walaupun menjelaskan kegiatan, *use case* hanya menjelaskan apa yang dilakukan oleh *actor* dan sistem, bukan bagaimana *actor* dan sistem melakukan kegiatan tersebut. (Gambar 2.6).



Gambar 2.6 Notasi *Use Case*

e. Interaction

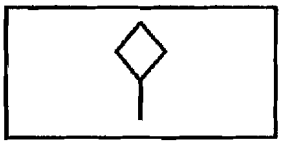
Interaction digunakan untuk menunjukkan aliran pesan atau informasi antar obyek maupun hubungan antar obyek. Biasanya *interaction* dilengkapi juga dengan teks penanda suatu proses yang tersusun dari nama operasi, parameter yang dikirim dan tipe parameter yang dikembalikan. (Gambar 2.7).



Gambar 2.7 Notasi Interaction

f. Aggregation

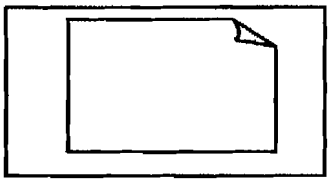
Aggregation adalah relasi dengan perlakuan khusus yang disebut dengan 'bagian dari (*part of*)' yang menangani antar objek-objek dimana salah satunya adalah bagian dari yang lain. (Gambar 2.8).



Gambar 2.8 Notasi Aggregation

g. Note

Note digunakan untuk memberikan keterangan dan komentar tambahan dari suatu elemen sehingga bisa langsung terlampir dalam model. *Note* ini bisa ditempelkan ke semua elemen notasi yang lain. (Gambar 2.9).

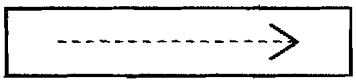


Gambar 2.9 Notasi *note*

h. Dependency

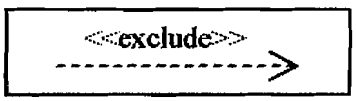
Dependency merupakan relasi yang menunjukkan bahwa perubahan pada salah satu elemen memberikan pengaruh pada elemen yang lain. Elemen yang ada di bagian tanda panah adalah elemen yang tergantung pada elemen yang ada di bagian tanda panah. Terdapat dua jenis dari *dependency* (gambar 2.10) yaitu *include dependency* gambar (2.11) dan *extend dependency* (gambar 2.12). *Include*

menunjukkan suatu bagian dari elemen yang ada di garis tanpa panah memicu eksekusi bagian dari elemen yang ada di garis dengan panah. *Extend* menunjukkan bahwa suatu bagian dari elemen di garis tanpa panah bisa disisipkan kedalam elemen yang terdapat pada garis dengan panah.



Gambar 2.10 Notasi *dependency*

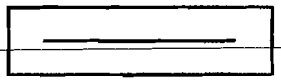
Gambar 2.11 Notasi *include dependency*



Gambar 2.12 Notasi *extend dependency*

i. Association

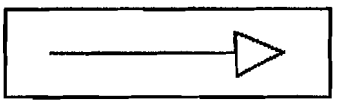
Association (gambar 2.13) menggambarkan navigasi antar *class* (*Navigation*), berapa banyak obyek lain yang bisa berhubungan dengan suatu obyek (*Multiplicity* antar *class*), dan apakah suatu *class* menjadi bagian dari *class* lainnya (*Agregation*).



Gambar 2.13 Notasi *Association*

j. Generalization

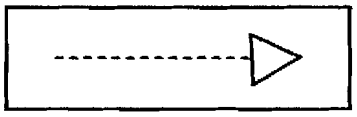
Generalization menunjukkan hubungan antara elemen yang lebih umum ke elemen yang lebih khusus atau lebih spesifik. Dengan *generalization* (gambar 2.14), *class* yang lebih spesifik (*subclass*) akan menurunkan atribut dan operasi yang sama dari *class* yang lebih umum (*superclass*), atau "*subclass is a superclass*".



Gambar 2.14 Notasi *Generalization*

k. Realization

Realization menunjukkan hubungan bahwa elemen yang ada di bagian tanpa panah akan merealisasikan apa yang dinyatakan oleh elemen yang ada di bagian dengan panah. Misalnya *class* merealisasikan *package*, *component* merealisasikan *class* atau *interface*. (gambar 2.15).



Gambar 2.15 Notasi *Realization*

l. Start state

Digunakan untuk menunjukkan titik awal sebuah *activity diagram*. Notasi *start state* dapat dilihat pada gambar 2.16.

Gambar 2.16 Notasi *Start state*

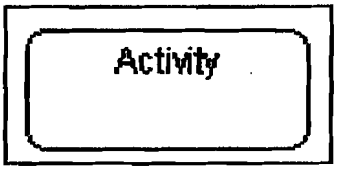
m. End state

Digunakan untuk menunjukkan titik akhir sebuah *activity diagram*. Notasi *end state* dapat dilihat pada gambar 2.17.

Gambar 2.17 Notasi *End state*

n. Activity

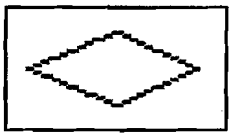
Digunakan untuk menunjukkan sebuah *activity*. Notasi *activity* dapat dilihat pada gambar 2.18.



Gambar 2.18 Notasi *Activity*

o. Decision

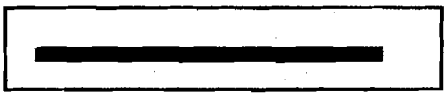
Digunakan untuk menunjukan sebuah pilihan dalam pengambilan keputusan. Notasi *decision* dapat dilihat pada gambar 2.19.



Gambar 2.19 Notasi *Decision*

p. Fork

Digunakan untuk menunjukkan kegiatan yang dilakukan secara pararel atau menggabungkan dua kegiatan pararel menjadi satu. Notasi *fork* dapat dilihat pada gambar 2.20.



Gambar 2.20 Notasi *Fork*

2.7 Perangkat Lunak *Microsoft Visio Profesional Edition*

Seiring dengan perkembangannya, terdapat beberapa alat pemodelan yang mendukung pemodelan dalam bahasa UML. Diantaranya, Rational Rose buatan Rational Software, Microsoft Visio Profesional Edition dari Microsoft, Together dari Borland dan Poseidon dari Gentleware.

Salah satu fitur dasar yang harus dimiliki oleh sebuah alat pemodelan UML adalah papan elemen yang digunakan untuk membuat diagram dengan cara memilih elemen yang diinginkan dari papan kemudian meletakkannya pada halaman gambar. Alat tersebut juga mengijinkan koneksi antar elemen dalam bidang gambar. Fitur penting yang lain adalah adanya dialog box untuk memodifikasi elemen dalam diagram serta menyimpan informasi-informasi dari elemen tersebut. Selain itu, alat pemodelan UML juga mengijinkan pengguna untuk mengatur informasi dengan cara yang diinginkan.

Fitur yang paling penting yang harus dimiliki oleh sebuah alat pemodelan UML adalah *dictionary*. Ini adalah rekaman dari seluruh elemen yang dibuat dan fitur-fiturnya. Selain digunakan untuk mengetahui aliran perancangan, *dictionary* ini memungkinkan pengguna untuk menggunakan kembali elemen pada diagram

yang berbeda. Fitur terakhir yang diperlukan adalah kemampuan alat pemodelan untuk untuk meng-*generate* kode dari diagram model yang telah dibuat.

Salah satu perangkat lunak terkemuka yang mendukung fitur-fitur diatas serta sering digunakan untuk membuat diagram UML adalah Microsoft Visio Profesional Edition. Hal yang membuat perangkat lunak ini dianggap sebagai alat pemodelan yang kuat adalah karena alat ini dilengkapi dengan sejumlah kemampuan dalam merancang diagram UML. UML merupakan salah satu dari sekian banyak fasilitas yang dimiliki oleh Microsoft Visio Profesional Edition.

2.8 Basis Data

Basis data merupakan kumpulan dari data yang saling berhubungan satu sama lainnya, tersimpan di perangkat keras komputer dan digunakan perangkat lunak untuk memanipulasinya [Jogiyanto, 1999]. Perangkat lunak yang dapat digunakan untuk mengelola basis data disebut *Database Management Sistem* (DBMS).

2.9 Microsoft SQL Server 7.0

Microsoft SQL server 7.0 termasuk jenis RDBMS (*Relational Database Management System*) [Siebold, 2001]. Sehingga istilah seperti tabel, baris dan kolom tetap digunakan dalam Microsoft SQL server 7.0. Pada Microsoft SQL server 7.0 sebuah *database* mengandung satu atau beberapa tabel, tabel terdiri dari sejumlah baris dan kolom. Ada empat kumpulan tipe data yang didukung oleh Microsoft SQL server 7.0 yaitu data numerik, data *string*, data *char*() dan *varchar*(), dan data tanggal.

Selain sebagai database server, Microsoft SQL server 7 merupakan suatu sistem manajemen database. Sebagai sistem manajemen database relasional, Microsoft SQL server 7.0 mempunyai kecepatan dan fleksibilitas. Tabel-tabel yang dihubungkan dengan relasi yang ditentukan membuatnya bisa mengkombinasikan data dari beberapa tabel pada suatu permintaan. Konektivitas,

kecepatan dan keamanannya membuat Microsoft SQL server 7.0 cocok untuk pengaksesan database pada sebuah sistem informasi.

2.10 Bahasa Pemrograman Java

Java adalah sebuah bahasa pemrograman yang juga merupakan suatu *platform*, dikeluarkan oleh *Sun Microsystem*. Bahasa Java awalnya bernama *Oak*, yang merupakan bagian dari proyek *Green* yang dikembangkan khusus oleh *Sun Microsystem* untuk memprogram perangkat-perangkat elektronik rumah tangga semacam televisi, mesin cuci, dll [Hermawan, 2004]. Seiring dengan perkembangannya kini java mulai digunakan untuk pemrograman secara umum.

Salah satu teknologi java adalah "*write once run everywhere*" yang menawarkan kehandalan java untuk dapat berjalan secara lintas sistem operasi. *Java Virtual Machine* (JVM) yang dapat menjembatani antara program dengan keunikan serta kompleksitas sistem operasi dimana program itu berjalan. Java 2 memiliki tiga kategori sebagai berikut [Hermawan, 2004]:

a. *Java 2 Platform, Standard Edition* (J2SE)

Kategori ini berisi *class-class* inti pada Java, *graphical user interface* (GUI) termasuk *applet*, fitur konektivitas basisdata, menangani *input/output* dan pemrograman jaringan, yang diaplikasikan pada level *Personal Computer* (PC).

b. *Java 2 Platform, Enterprise Edition* (J2EE)

J2EE digunakan untuk menjalankan dan mengembangkan aplikasi-aplikasi Java pada lingkungan *enterprise*, dengan menambah fungsionalitas-fungsionalitas Java seperti *Enterprise Java Bean* (EJB), Java CORBA, *Servlet* dan JSP serta *Java eXtensible Markup Language* (XML).

c. *Java 2 Platform, Micro Edition* (J2ME)

Kategori J2ME digunakan untuk menjalankan dan mengembangkan aplikasi-aplikasi Java pada *handled devices* atau perangkat-perangkat semacam telepon genggam, *Palm*, PDA dan *Pocket* PC.

Java memiliki beberapa keunggulan jika dibanding bahasa pemrograman lain, yaitu:

a. Java bersifat sederhana dan relatif mudah

Berbagai kemudahan yang ada dalam bahasa pemrograman Java antara lain Java menggantikan konsep pewarisan lebih dari satu (*multiple inheritance*) dengan *interface*, menghilangkan konsep *pointer* yang sering membingungkan dan otomatisasi sistem alokasi memori.

b. Java berorientasi pada obyek (*Object Oriented*)

Java adalah bahasa pemrograman berorientasi obyek yang membagi masalah menjadi obyek-obyek, kemudian memodelkan sifat dan tingkah laku masing-masing. Setelah itu Java menentukan dan mengatur interaksi antar obyek yang satu dengan yang lain.

c. Java bersifat terdistribusi

Java mendukung sistem komputerisasi terdistribusi mulai dari *workstation client*, *database server*, *web server* dan *proxy server*.

d. Java bersifat *Multiplatform*

Java bersifat *multiplatform*, yaitu dapat diterjemahkan oleh Java *interpreter* pada berbagai sistem operasi.

2.11 Microsoft Visual basic 6.0

Microsoft Visual Basic 6.0 (VB6) merupakan salah satu aplikasi pemrograman *visual* yang dibuat oleh Microsoft. Microsoft Visual Basic 6.0 berjalan dalam sistem operasi Windows dan tergabung dalam *suite* aplikasi Microsoft Visual basic 6.0 yang dikeluarkan pada akhir tahun 1998.

Aplikasi Visual Basic mulai diproduksi pertama kali pada tahun 1991. setelah itu munculah versi-versi lanjutan dari Visual Basic, yaitu Visual basic 3, 4, 5, dan 6. Pada Visual Basic 4, dukungan terhadap aplikasi 32 bit mulai diberikan. Versi visual basic yang terbaru adalah Visual Basic.NET yang dirilis pada tahun 2002. Visual basic 6.0 terdiri dari tiga buah edisi, yaitu [Siebold, 2001] :

a. *Standart Edition*, merupakan produk dasar
b. *Profesional Edition*, berisi tambahan Microsoft *Jet Data Access Engine* dan pembuatan *server OLE Automation*.

c. *Enterprise Edition*, merupakan edisi untuk membuat program aplikasi *client-server*.

Microsoft Visual Basic 6.0 menyediakan berbagai perangkat yang dapat digunakan untuk mambuat program aplikasi baik aplikasi kecil dan sederhana untuk kepeluan sendiri, hingga aplikasi untuk sistem *enterprise* yang besar dan rumit atau bahkan aplikasi yang dijalankan melalui internet.

Microsoft Visual basic 6.0 memanfaatkan pendekatan *visual*/GUI (*Graphical User Interface*) dalam proses penggunaannya. Dengan pendekatan GUI, proses pembuatan program aplikasi menjadi lebih mudah dan nyaman. Basis bahasa pemrograman yang digunakan dalam VB6 adalah bahasa BASIC (*Beginners All-Purpose Symbolic Instruction Code*).

2.12 Poliklinik

Poliklinik adalah sebuah badan usaha yang bergerak dibidang pelayanan kesehatan. Poliklinik dibangun bertujuan untuk mempermudah pasien dalam memperoleh pertolongan pertama dan terdekat pada saat sakit. Poliklinik merupakan unit kecil dari seluruh bidang pelayanan kesehatan yang dapat menjangkau area-area yang cukup jauh dari rumah sakit.

Poliklinik UII merupakan fasilitas kesehatan milik Universitas Islam Indonesia yang ditujukan untuk melayani kesehatan bagi seluruh karyawan dan mahasiswa. Fasilitas ini dilayani oleh tiga orang dokter dan dua orang paramedis yang bertugas secara bergantian dalam satu minggu. poliklinik ini juga dilengkapi dengan obat-obatan yang dibutuhkan. Obat-obatan tersebut dapat diperoleh di apotek UII Polifarma yang berada di dalam poliklinik tersebut. Besarnya biaya pengobatan, ditentukan secara variatif. Untuk mahasiswa, dibebani biaya sebesar nilai obat yang diberikan, sedangkan bagi dosen dan karyawan dikenai biaya Rp. 1000,00 (seribu rupiah). Poliklinik UII berada di tempat yang strategis dan nyaman serta dilengkapi dengan fasilitas apotek UII Polifarma yang buka sampai pukul 21.00 WIB.

BAB III

METODOLOGI

Dalam bab metodologi ini akan dibahas uraian langkah-langkah penyelesaian masalah. Sesuai dengan hal tersebut, dalam bab ini akan dibahas empat segmen dalam kerangka pengembangan GRAPPLE (*Guidelines for Rapid APPLication Engineering*) yaitu *requirements gathering*, analisis, desain dan *development*. Pada segmen *requirements gathering* digunakan lima buah diagram antara lain *activity* diagram menggunakan *swimlane* dan tidak menggunakan *swimlane*, *high-level class* diagram, *deployment* diagram serta *package* diagram.

Kemudian pada segmen analisis akan digunakan enam buah diagram UML yaitu *use case* diagram, *state* diagram, *sequence* diagram, *collaboration* diagram, penyempurnaan dari *class* diagram dan *deployment* diagram. Sedangkan dalam segmen desain digunakan *component* diagram, *object* diagram, perluasan dari diagram *activity* dan *deployment* diagram serta *prototype* antarmuka. Penggunaan diagram-diagram UML tersebut dirasa cukup untuk menjelaskan sistem yang akan dibuat.

3.1 *Requirements Gathering*

Dalam segmen ini akan dibahas 4 aksi yaitu pengumpulan informasi proses bisnis, analisis domain, mengidentifikasi sistem yang saling bekerjasama dan analisis kebutuhan sistem.

3.1.1 Pengumpulan informasi proses bisnis

Sebelum membangun sebuah sistem yang baik, sangat diperlukan pemahaman yang cukup terhadap proses bisnis yang terjadi pada obyek penelitian. Aksi ini diperlukan untuk memahami tentang proses pendaftaran pasien baru

maupun lama yang terjadi di lokasi penelitian. Untuk mendapatkan informasi yang dibutuhkan, metode yang digunakan adalah sebagai berikut :

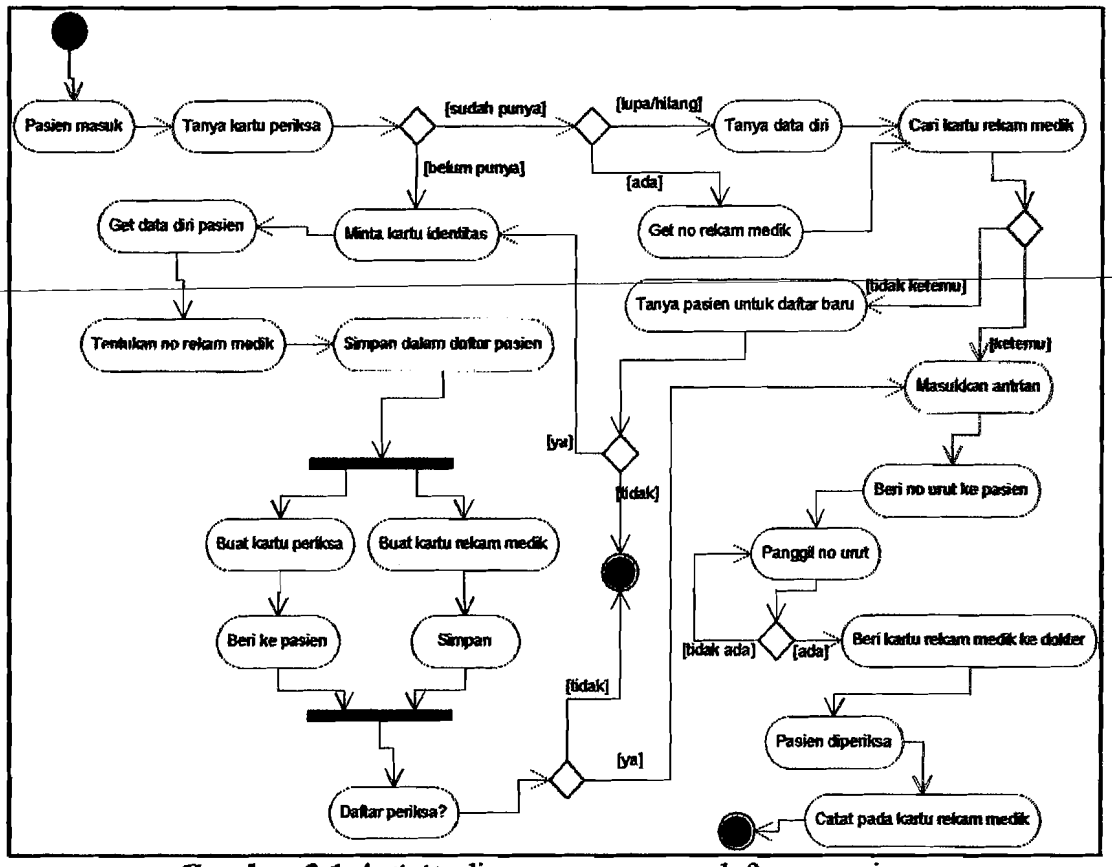
a. Metode Observasi

Observasi dilakukan dengan mengamati secara langsung proses pendaftaran pasien pada obyek penelitian di Poliklinik Universitas Islam Indonesia.

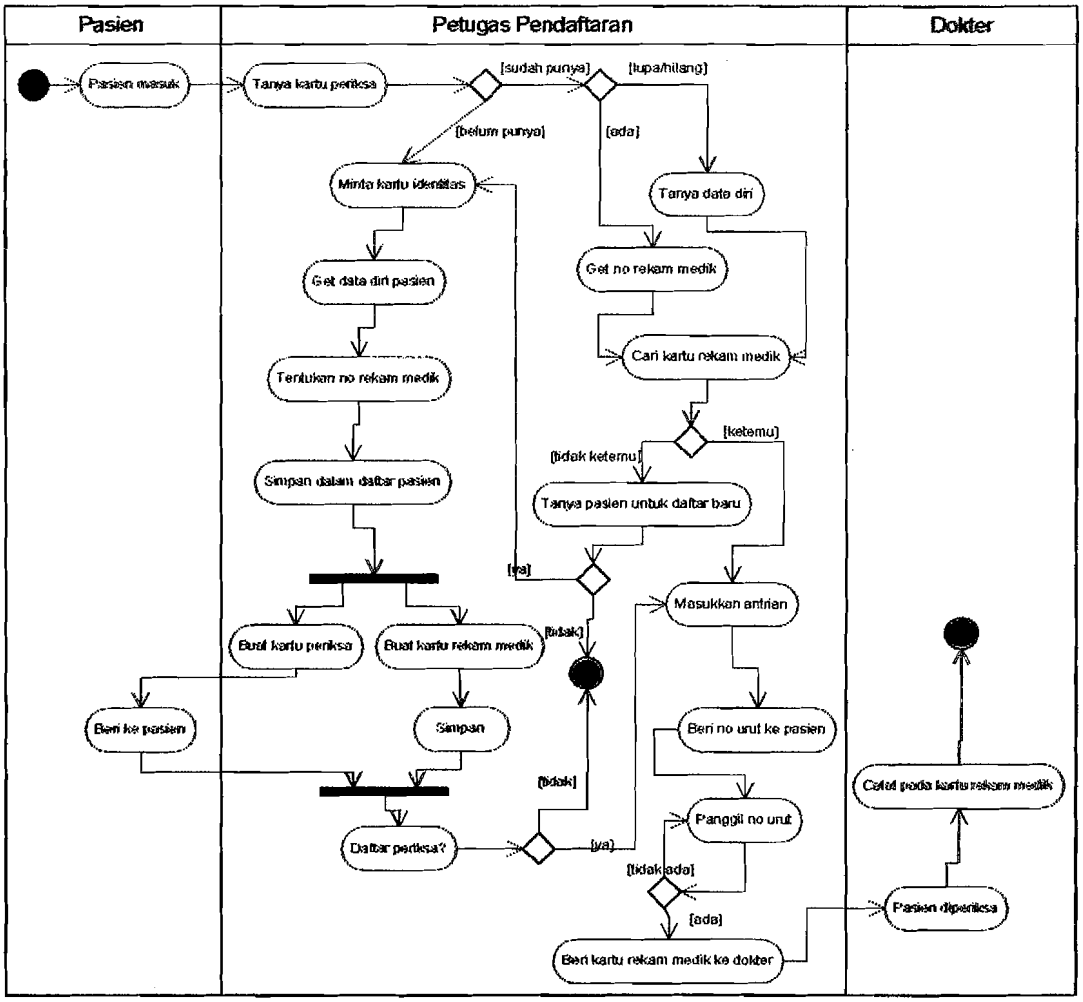
b. Metode *Interview*

Interview atau wawancara secara langsung dilakukan pada bulan Februari 2007 dengan petugas pendaftaran pasien di poliklinik tersebut secara sistematik dan berdasarkan pada tujuan penelitian.

Hasil yang diperoleh dari pengumpulan informasi proses bisnis melalui kedua metode diatas berupa gambaran secara umum dari proses pendaftaran pasien di poliklinik tersebut. Aliran proses pendaftaran pasien di poliklinik tersebut diperlihatkan dalam bentuk *activity* diagram pada gambar 3.1 dan gambar 3.2.



Gambar 3.1 *Activity* diagram proses pendaftaran pasien



Gambar 3.2 *Activity* diagram proses pendaftaran pasien menggunakan ***swimlane***

3.1.2 Analisis domain

Analisis domain dilakukan untuk mengumpulkan informasi mengenai para pelaku proses bisnis dan operasi apa saja yang dilakukannya. Sebagai catatan, kedua buah aksi yang pertama kali dilakukan ini bukan membahas mengenai sistem yang akan dibangun, melainkan membahas mengenai konsep dasar proses bisnis obyek penelitian dan para pelakunya. Berikut ini beberapa pernyataan yang menjelaskan mengenai proses pendaftaran pasien di poliklinik.

a. Pasien masuk, petugas pendaftaran akan menanyakan apakah pasien telah memiliki nomor rekam medik. Jika belum, maka petugas akan menentukan nomor rekam medik pasien, kemudian menyimpan data diri pasien dan

membuat kartu rekam medik serta kartu periksa sesuai dengan data diri tersebut,

b. Petugas pendaftaran menyerahkan kartu periksa yang berisi data diri dan nomor rekam medik kepada pasien untuk dipergunakan pada pemeriksaan selanjutnya serta menyimpan kartu rekam medik,

c. Petugas pendaftaran memasukkan nomor rekam medik pasien pada daftar antrian dan memberikan nomor urut antrian kepada pasien,

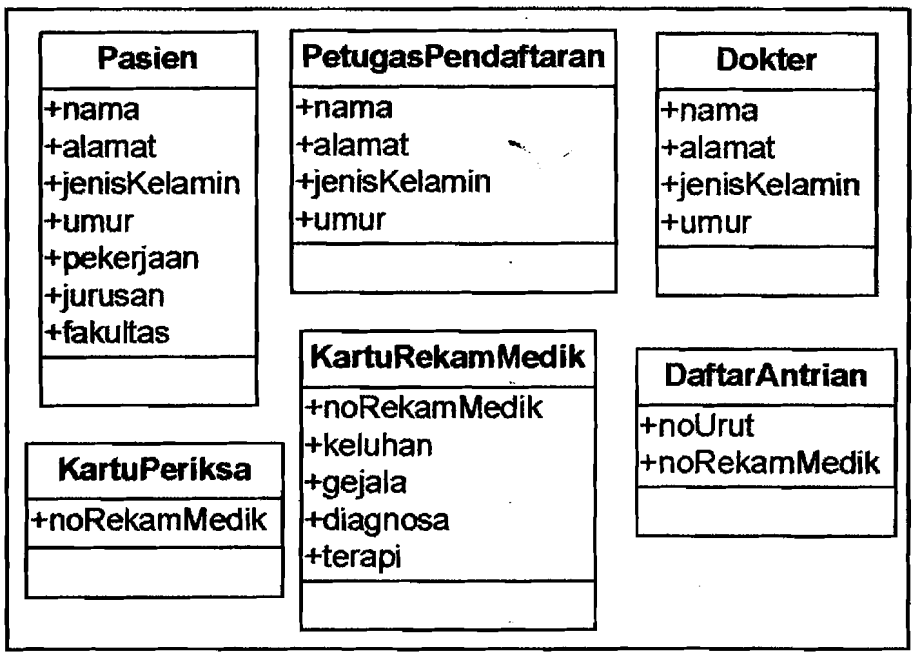
d. Petugas pendaftaran memanggil nomor urut antrian pasien, kemudian menyerahkan kartu rekam medik yang sesuai kepada dokter,

e. Dokter melakukan pemeriksaan terhadap pasien sesuai dengan nomor urut yang dipanggil oleh petugas pendaftaran,

f. Setelah pemeriksaan, dokter mencatat tanggal kunjungan, keluhan, diagnosa dan jenis terapi pada kartu rekam medik pasien,

g. Pasien yang telah selesai dilakukan pemeriksaan, maka nomor rekam mediknya akan dihapus dari daftar antrian.

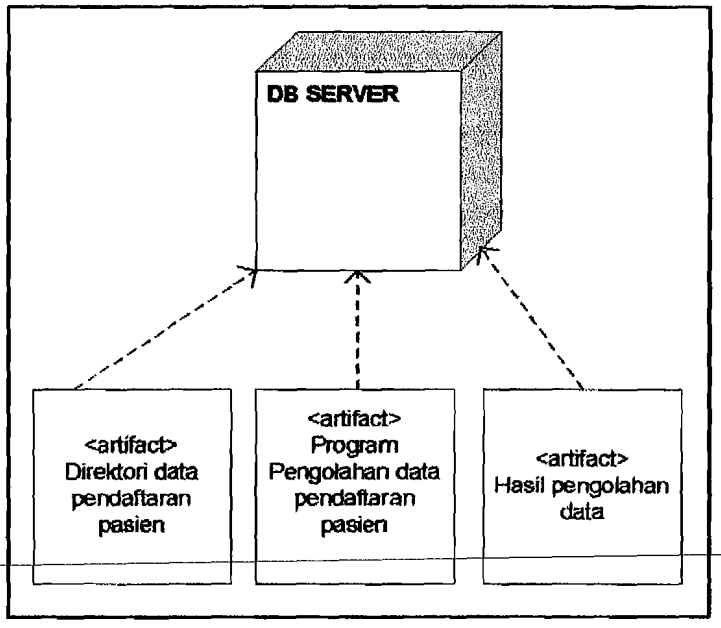
Dari pernyataan-pernyataan diatas dapat diketahui obyek-obyek yang terlibat beserta atribut dan operasi setiap obyeknya, obyek-obyek ini kemudian dibangun dalam diagram *high-level-class*. (Gambar 3.3)



Gambar 3.3 *High-level-class* diagram

3.1.3 Mengidentifikasi sistem yang saling bekerjasama

Tidak ada satupun sistem yang mampu berdiri sendiri tanpa bekerjasama dengan sistem yang lain. Begitu pula sistem yang akan dipergunakan dalam penelitian ini. Mengindentifikasi sistem yang saling bekerjasama adalah menggambarkan model diagram untuk menunjukkan ketergantungan yang terjadi pada sistem yang akan dibangun. *Deployment* diagram berikut ini akan menunjukkan sistem sebagai *node-node* yang dihubungkan dengan garis *internode* sebagai bentuk komunikasi antara sistem, komponen-komponen serta ketergantungan antar komponen. (gambar 3.4)



Gambar 3.4 *Deployment* diagram

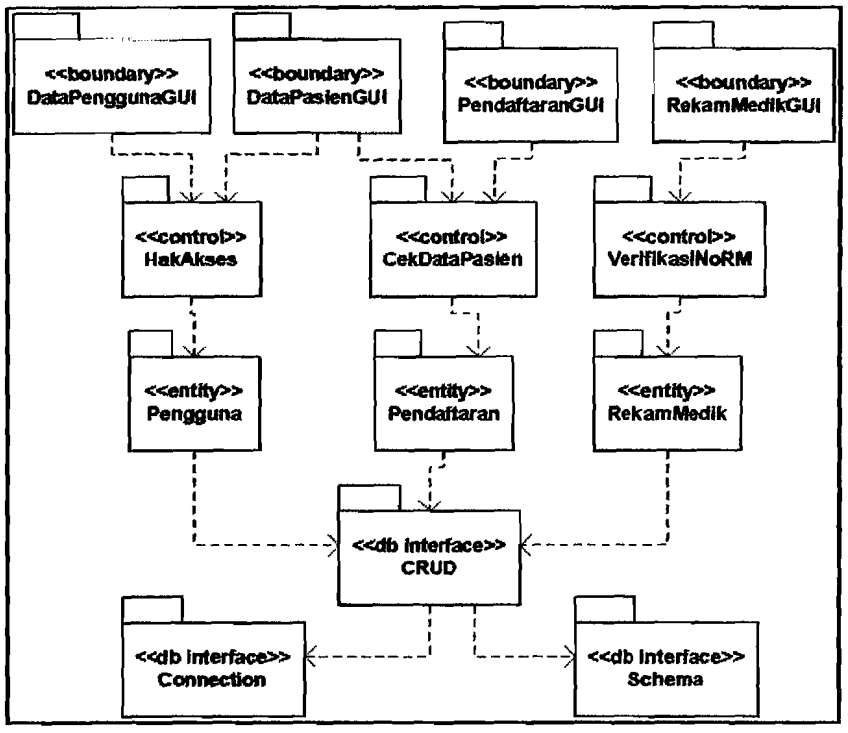
3.1.4 Analisis sistem yang diperlukan

Aksi ini merupakan aksi yang paling penting dalam GRAPPLE karena merupakan awal dari *Joint Application Development* (JAD) *session*. JAD *session* ini adalah diskusi yang melibatkan petugas pendaftaran, pasien dan seluruh pelaku yang terlibat dalam proses pendaftaran pasien. Pada aksi ini para pelaku tersebut menentukan, sistem seperti apa yang mereka inginkan. Hal apa saja yang dapat dilakukan oleh sistem tersebut.

Class-class yang sudah didefinisikan di depan kebanyakan adalah obyek-obyek bisnis atau obyek-obyek *database*. Sebuah sistem yang lengkap seharusnya mencakup juga *class-class* untuk program aplikasi. Untuk itu pendekatan BCED (*Boundary-Control-Entity-Database*) bisa dilakukan.

Berdasarkan keterangan para pelaku bisnis, dapat disimpulkan sistem informasi ini akan menangani tiga buah fungsi utama antara lain manipulasi data *user* sistem, proses pendaftaran pasien dan manipulasi data rekam medik pasien. Fungsi-fungsi tersebut membutuhkan *window* yang berbeda-beda. Oleh sebab itu dibutuhkan *boundary class* antara lain, DataUserGUI, DataPasienGUI PendaftaranGUI dan RekamMedikGUI.

Dari sisi bisnis, *class-class* yang telah diperoleh sebelumnya dapat dikelompokkan menjadi 3 *package entity class* yaitu, Pengguna, Pendaftaran dan RekamMedik. Agar *boundary* dan *entity class* diatas dapat disatukan, maka diperlukan *class control*. *Class control* bertanggung jawab terhadap logika dari aplikasi. *Class control* yang diperlukan antara lain, HakAkses, CekDataPasien dan VerifikasiNoRM. Hasil yang dicapai pada aksi ini adalah sebuah *package* diagram. (Gambar 3.5)



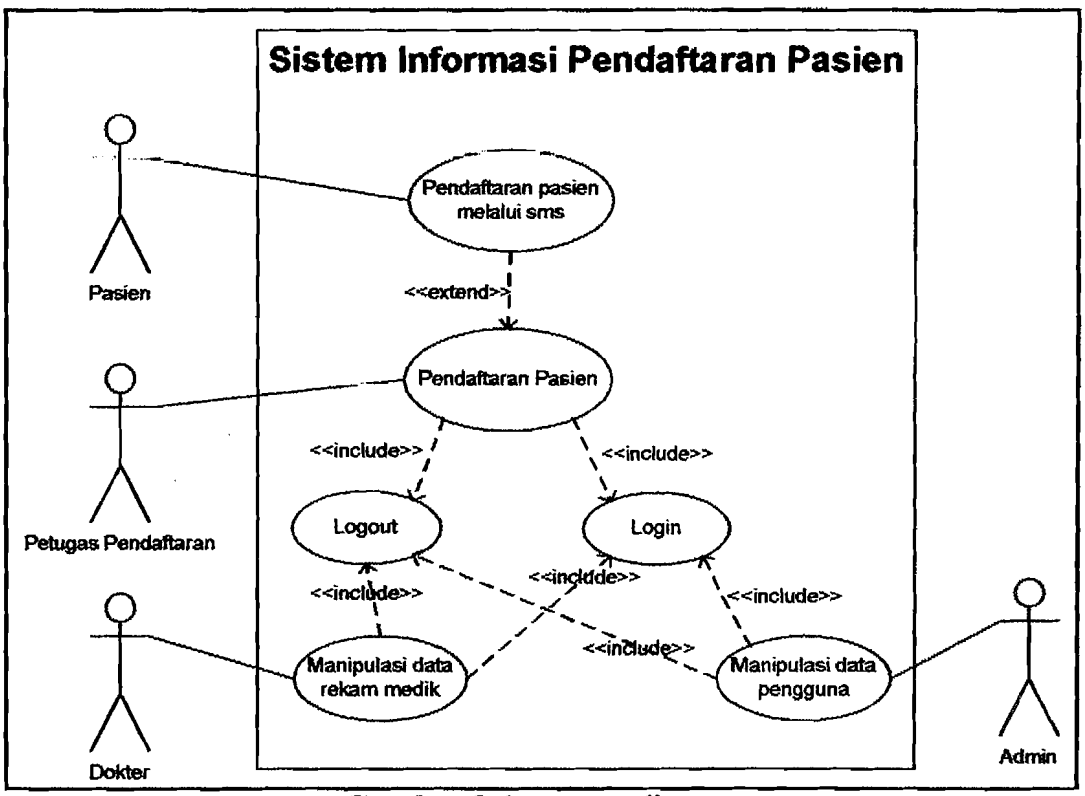
Gambar 3.5 *Package* diagram

3.2 Analisis

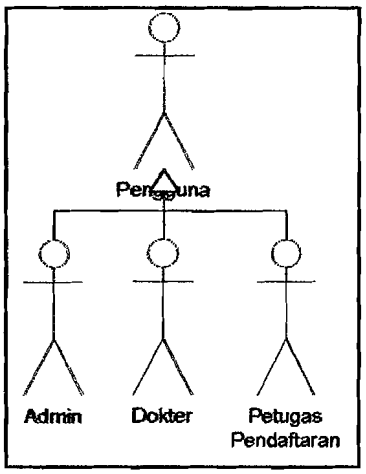
Dalam segmen ini akan dibahas enam buah aksi yang akan menganalisis lebih dalam hasil dari segmen sebelumnya, aksi-aksi tersebut antara lain identifikasi kegunaan sistem, analisis kumpulan *use case*, menyempurnakan *class* diagram, analisis perubahan *state* dalam obyek, menentukan relasi antar obyek dan analisa integrasi dengan sistem terkait.

3.2.1 Identifikasi kegunaan sistem

Pada aksi ini dibahas kegunaan sistem bagi masing-masing *user* yang digambarkan menggunakan *use case* diagram. *Use case* diagram berfungsi untuk menjelaskan manfaat sistem menurut sudut pandang orang yang berada di luar sistem tersebut. Fungsi utama dari sistem informasi ini adalah mampu memproses pendaftaran pasien melalui SMS. Untuk itu diperlukan beberapa fungsi penting tambahan untuk mendukung proses pendaftaran yang terjadi. (Gambar 3.6)

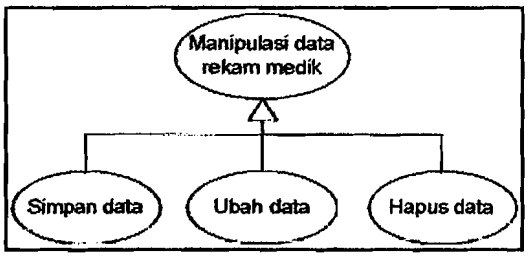


Gambar 3.6 *use case* diagram

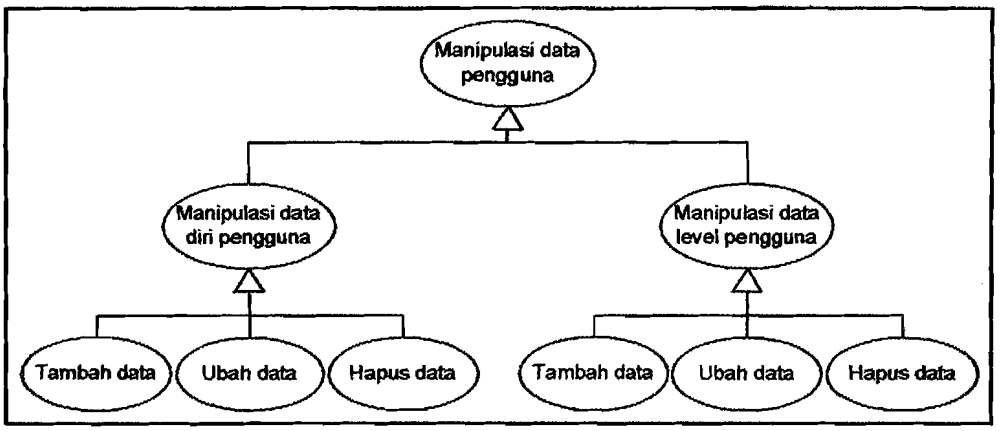


Gambar 3.7 *Generalization relationship* pada *actor*

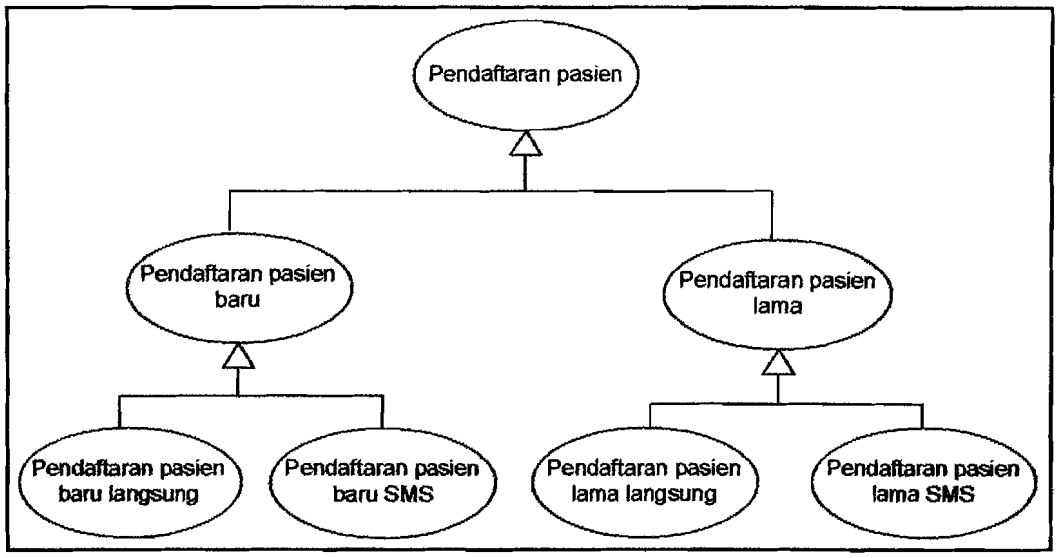
Dalam hubungan generalisasi, sebuah kelas dapat mewarisi atribut dan *method* dari kelas induknya, begitu juga dengan *actor* dan *use case*. Gambar 3.7 menunjukan hubungan generalisasi yang dibentuk dari *actor* admin, dokter dan petugas pendaftaran yang *inherit* dari *actor* pengguna. Sedangkan gambar 3.8, gambar 3.9 dan gambar 3.10 menunjukan hubungan generalisasi yang dibentuk dari *use case-use case*. *Use case* anak mewarisi fungsi dari *use case* induknya.



Gambar 3.8 *Generalization relationship* pada *usecase* manipulasi data rekam medik



Gambar 3.9 *Generalization relationship* pada *usecase* manipulasi data pengguna



Gambar 3.10 *Generalization relationship* pada *usecase* pendaftaran pasien

3.2.2 Analisis kumpulan *use case*

Pada aksi ini dibahas mengenai deskripsi langkah-langkah pada masing-masing *use case* diatas. Urutan langkah-langkah tersebut dijelaskan dalam bentuk dokumen yang disebut dengan dokumen *flow of event*. Dokumen ini mendefinisikan apa yang harus dilakukan oleh sistem ketika *actor* mengaktifkan *use case*.

3.2.2.1 Use case specification : Pendaftaran pasien

1. Use Case Name : Pendaftaran pasien

1.1 Brief description

Proses ini dapat dilakukan oleh *actor* pengguna dengan hak akses petugas pendaftaran dan pasien. Pengguna sistem melakukan proses pendaftaran pasien dengan cara menggunakan sistem ini secara langsung. Sedangkan pasien melakukan pendaftaran untuk dirinya sendiri melalui SMS.

2. Flow of events

2.1 Basic flow

a. [UC1.1: Pendaftaran pasien baru]

- *[UC1.1.1: Pendaftaran pasien baru secara langsung]*
  a. Pengguna mengaktifkan *form* penambahan data pasien. *Form* ini berfungsi untuk menyimpan data pasien ke dalam *database*.
  b. Pengguna mengisi data diri pasien pada isian-isian yang terdapat dalam *form*.
  c. Setelah terisi, pengguna menekan tombol [SIMPAN] untuk menyimpan data ke dalam *database*.
- *[UC1.1.2: Pendaftaran pasien baru melalui SMS]*
  a. Pasien mengetikkan BARU nama#alamat#umur#jurusan#fakultas. Jika benar maka sistem akan menyimpan data pasien tersebut sementara sesuai dengan batas waktu yang diberikan.
  b. SMS server akan membalas dengan pesan, anda memiliki kesempatan sampai dengan [batas waktu] untuk memverifikasi data tersebut dan mendapatkan kartu periksa. Terimakasih.

b. [UC1.2: Pendaftaran pasien lama]

- *[UC1.2.1: Pendaftaran pasien lama secara langsung]*
  a. Pengguna mengaktifkan *form* penambahan data antrian.
  b. Jika pasien mempunyai nomor rekam medik, pengguna akan memasukkan nomor rekam medik ke daftar antrian. Jika nomor rekam medik hilang/lupa maka petugas mencari nomor rekam medik sesuai dengan data diri yang diberikan pasien.
  c. Setelah terisi, pengguna menekan tombol [SIMPAN] untuk menyimpan data ke dalam *database*.
- *[UC1.2.2: Pendaftaran pasien lama melalui SMS]*
  a. Pasien mengetikkan DAFTAR nomor rekam medik. Jika benar maka sistem akan menyimpan data pasien tersebut ke dalam daftar antrian.
  b. SMS server akan membalas dengan pesan, anda terdaftar dengan nomor antrian 7.

2.2 Alternative flow

- *[UC1.1.1.1: Jika pasien baru ingin langsung diperiksa, tambahkan nomor rekam medik pada daftar antrian.]*

- *[UC1.2.1.1: Jika nomor rekam medik pasien tidak ditemukan, maka sistem akan menanyakan apakah pasien ingin mendaftar kembali?]*

3. Assumption

3.1 Pasien belum pernah melakukan pendaftaran.

3.2 Pasien telah mengetahui prosedur pendaftaran melalui SMS.

3.3 Koneksi SMS server dengan terminal dan database yang dilakukan oleh pengguna berhasil.

3.4 Tidak terjadi kesalahan dalam *sim card* dan telepon genggam SMS server.

4. Pre-conditions

4.1 Pengguna mengaktifkan Sistem Informasi Pendaftaran Pasien berbasis SMS tersebut dan telah berhasil melewati proses login.

4.2 Pengguna mengaktifkan SMS server dan melakukan koneksi

4.3 Pasien mengetikkan kata kunci DAFTAR PERIKSA untuk memperoleh informasi pendaftaran melalui SMS.

5. Post-Conditions

5.1 Pasien yang telah memiliki nomor urut dipanggil untuk dilakukan pemeriksaan.

3.2.2.2 Use case specification : Manipulasi data pengguna

1. Use Case Name : Manipulasi data pengguna

1.1 Brief description

Actor pengguna dengan hak akses admin dapat melakukan pencetakan, penyimpanan, pengubahan dan penghapusan terhadap data pengguna-pengguna sistem yang lain serta dapat menambahkan data level pengguna.

2. Flow of events

2.1 Basic flow

a. [UC2.1: Manipulasi data diri pengguna]

- *[UC2.1.1: Menambahkan data pengguna]*

  a. Pengguna mengaktifkan *form* penambahan data pengguna. *Form* ini berfungsi untuk menyimpan data pengguna sistem ke dalam *database*.

b. Pengguna mengisi data diri pengguna pada isian-isian yang terdapat dalam *form*.

c. Setelah terisi, pengguna menekan tombol [SIMPAN] untuk menyimpan data ke dalam *database*.

- *[UC2.1.2: Mengubah data pengguna]*

a. Pengguna mengaktifkan *form* pengubahan data pengguna. *Form* ini berfungsi untuk mengubah data pengguna sistem yang ada pada *database*.

b. Pengguna mengubah data diri pengguna pada isian-isian yang terdapat dalam *form*.

c. Setelah pengubahan selesai, pengguna menekan tombol [SIMPAN] untuk menyimpan data ke dalam *database*.

- *[UC2.1.3: Menghapus data pengguna]*

a. Pada form daftar pengguna, pengguna memilih data pengguna yang ingin dihapus.

b. Kemudian, pengguna menekan tombol [HAPUS] untuk menghapus data di dalam *database*.

b. [UC2.2: Manipulasi data level pengguna]

- *[UC2.2.1: Menambahkan data level pengguna]*

a. Pengguna mengaktifkan *form* penambahan data level pengguna. *Form* ini berfungsi untuk menyimpan data level pengguna sistem ke dalam *database*.

b. Pengguna mengisi data level dan keterangannya pada isian-isian yang terdapat dalam *form*.

c. Setelah terisi, pengguna menekan tombol [SIMPAN] untuk menyimpan data ke dalam *database*.

- *[UC2.2.2: Mengubah data level pengguna]*

a. Pengguna mengaktifkan *form* pengubahan data level pengguna. *Form* ini berfungsi untuk mengubah data level pengguna sistem yang ada pada *database*.

b. Setelah pengubahan selesai, pengguna menekan tombol [SIMPAN] untuk menyimpan data ke dalam *database*.

- *[UC2.2.3: Menghapus data level pengguna]*

a. Pada form daftar level pengguna, pengguna memilih data level pengguna yang ingin dihapus.

b. Kemudian, pengguna menekan tombol [HAPUS] untuk menghapus data di dalam *database*.

2.2 Alternative flow

- *[UC2.1.3.1 : Sistem akan menolak jika data pengguna yang dihapus berada pada level tertinggi]*
- *[UC2.2.3.1 : Sistem akan menolak jika data level pengguna yang dihapus adalah level tertinggi]*

3. Assumption

3.1 Sistem telah mempunyai seorang pengguna dengan level tertinggi.

3.2 Tidak ada kendala dalam proses koneksi *database*.

4. Pre-conditions

4.1 Pengguna mengaktifkan Sistem Informasi Pendaftaran Pasien berbasis SMS tersebut dan telah berhasil melewati proses login.

4.2 Terdapat data pengguna lain selain data pengguna dengan level tertinggi.

5. Post-Conditions

5.1 Data pengguna dan level pengguna berhasil ditambah, diubah dan dihapus.

3.2.2.3 Use case specification : Manipulasi data rekam medik

1. Use Case Name : Manipulasi data rekam medik

1.1 Brief description

Actor pengguna level dokter dapat melakukan pencetakan, penyimpanan, pengubahan dan penghapusan terhadap data rekam medik pasien.

2. Flow of events

2.1 Basic flow

a. [UC3.1: Menambahkan data rekam medik]

- Pengguna mengaktifkan *form* penambahan data rekam medik. *Form* ini berfungsi untuk menyimpan data rekam medik ke dalam *database*.
- Pengguna mengisi data rekam medik pada isian-isian yang terdapat dalam *form*.
- Setelah terisi, pengguna menekan tombol [SIMPAN] untuk menyimpan data ke dalam *database*.

b. [UC3.2 : Mengubah data rekam medik]

- Pengguna mengaktifkan *form* pengubahan data rekam medik. *Form* ini berfungsi untuk mengubah data rekam medik yang ada pada *database*.
- Pengguna mengubah data rekam medik pada isian-isian yang terdapat dalam *form*.
- Setelah pengubahan selesai, pengguna menekan tombol [SIMPAN] untuk menyimpan data ke dalam *database*.

c. [UC3.3 : Menghapus data rekam medik]

- Pada form daftar rekam medik, pengguna memilih data rekam medik yang ingin dihapus.
- Kemudian, pengguna menekan tombol [HAPUS] untuk menghapus data di dalam *database*.

2.2 Alternative flow

- *[UC3.3.1 : Sistem akan memberitahukan jika data rekam medik yang dipilih masih kosong]*

3. Assumption

3.1 Sistem telah mempunyai daftar data diri pasien.

3.2 Sistem telah mempunyai daftar data diri pengguna dengan hak akses dokter.

3.3 Tidak ada kendala dalam proses koneksi *database*.

4. Pre-conditions

4.1 Pengguna mengaktifkan Sistem Informasi Pendaftaran Pasien berbasis SMS tersebut dan telah berhasil melewati proses login.

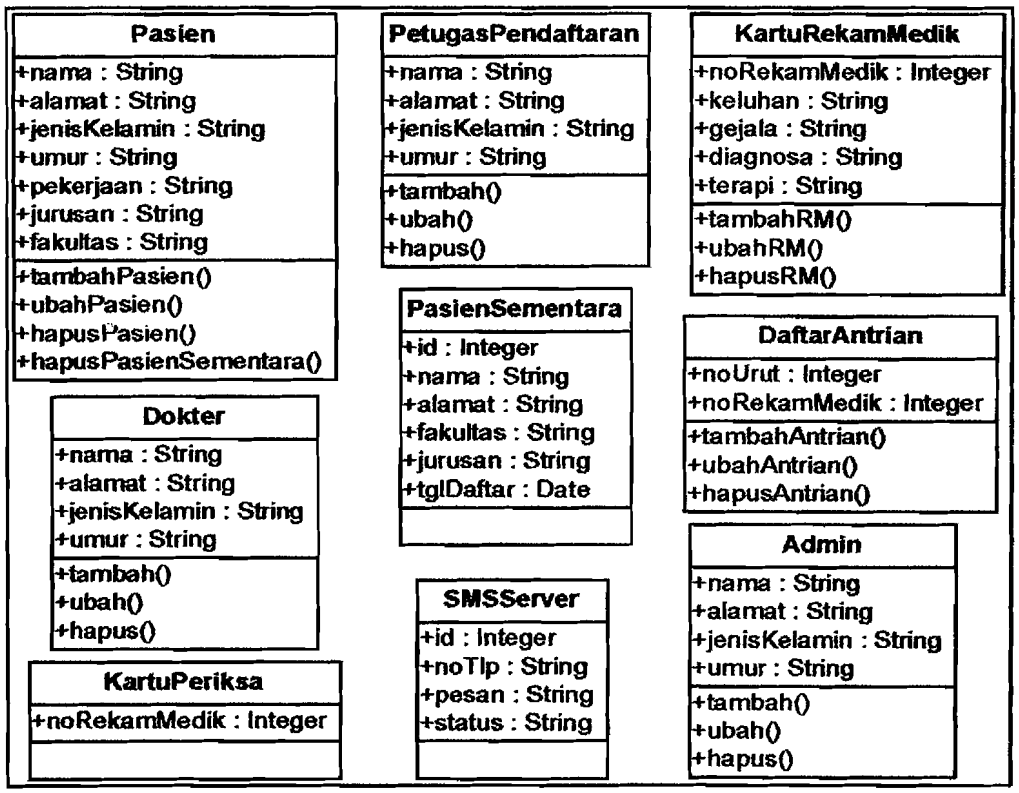
5. Post-Conditions

5.1 Data rekam medik pasien berhasil ditambah, diubah dan dihapus.

3.2.3 Memperinci *class* diagram

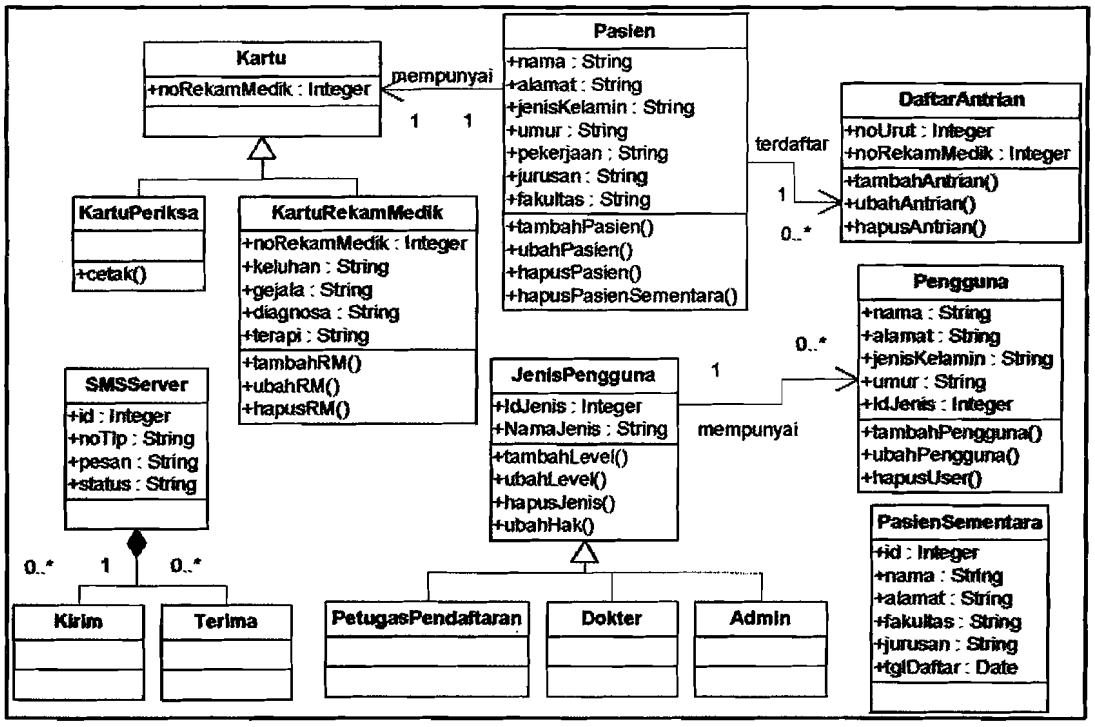
Pada aksi ini dilakukan perbaikan dan perincian pada diagram *high-level-class* yang dihasilkan pada subbab 3.1.2 diatas dengan menambahkan kelas-kelas yang dibutuhkan dan relasi-relasi yang terjadi antar kelas seperti asosiasi, *multiplicity* dan generalisasi.

Pada pembahasan sebelumnya, diketahui bahwa *user* menginginkan proses pendaftaran pasien dapat dilakukan secara langsung maupun melalui SMS, serta dilengkapi dengan hak akses yang berbeda-beda untuk masing-masing *user* dalam mengakses sistem tersebut. Dari proses-proses yang diinginkan oleh *user* tersebut diagram *high-level-class* yang telah diperoleh sebelumnya disempurnakan menjadi gambar 3.11.



Gambar 3.11 penyempurnaan *high-level class* diagram

UML mendefinisikan struktur data yang dibutuhkan oleh sebuah aplikasi dalam *class* diagram. Struktur data yang tetap di *database* dimodelkan sebagai *class entity* dan sebagai relasi diantara *class entity*. *Class entity* ini perlu di*mapping*kan ke struktur data yang dikenal oleh *database*. (Gambar 3.12)



Gambar 3.12 *Class* diagram

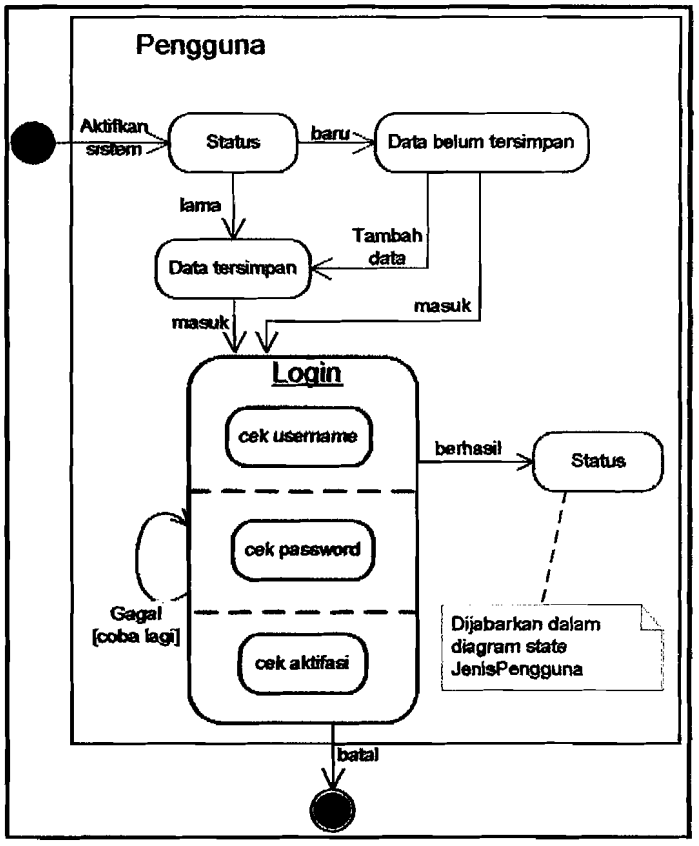
3.2.4 Analisis perubahan *state* dalam obyek

Interaction diagram dan *statechart* menampilkan dua pandangan yang saling melengkapi mengenai perilaku dinamis sebuah sistem. *Statechart* diagram diperlukan untuk membantu analis, perancang dan pengembang untuk memahami perilaku obyek dalam sistem.

Pada aksi ini dibahas lebih jauh tentang pemodelan obyek dengan menunjukkan perubahan *state* yang dibutuhkan. *Statechart* diagram menampilkan *state-state* yang mungkin dari sebuah obyek, *event* yang bisa dideteksi dan respon atas *event-event* tersebut. *State-state* yang mungkin digambarkan dengan bentuk *round rectangle* sedangkan tanda panah menunjukkan *event*.

Gambar 3.13 menunjukkan kumpulan *state* dan *event* yang mungkin dari kelas Pengguna. Awalnya pengguna berada pada *state* `status`, jika pengguna belum menyimpan data dirinya dalam sistem, maka pengguna mendapatkan *event* `baru`. *Event* `baru` mengubah *state* awal pengguna menjadi *state* `data belum tersimpan`. Begitu juga *state* `status` dengan *event* `lama` akan berpindah ke *state*

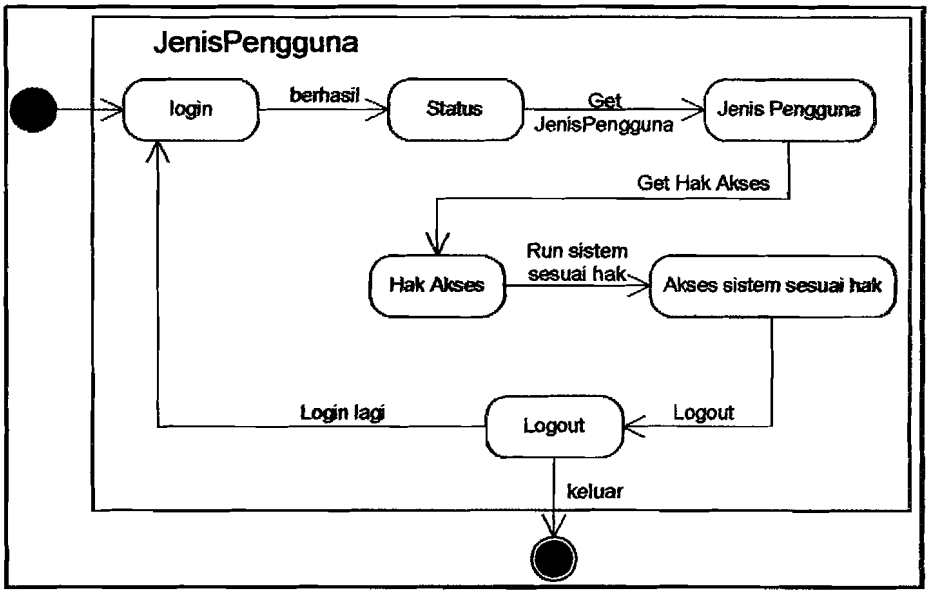
`data tersimpan`. Selanjutnya *state* `data belum tersimpan` mempunyai dua buah *event*, jika *event* `tambah data` diaktifkan maka *state* akan berpindah ke *state* `data tersimpan`, sebaliknya jika *event* `masuk` diaktifkan maka *state* akan berpindah ke *state* `login`. *State* `login` mempunyai 3 buah *substate* yang akan dieksekusi secara bersamaan atau *pararel*, aksi ini ditandai dengan garis putus-putus yang membatasi *state* `cek username`, `cek password` dan `cek aktifasi`. Jika pengguna gagal login maka *event* `gagal` diaktifkan dan posisi tetap berada *state* `login`, jika berhasil *event* `berhasil` diaktifkan *state* berpindah ke *state* `status` yang kemudian akan dilanjutkan pada kelas JenisPengguna. Jika *event* `batal` diaktifkan, maka *state* akan berpindah ke *finish*.



Gambar 3.13 *Statechart* diagram untuk *class* Pengguna

Gambar 3.14 menunjukkan kumpulan *state* dan *event* yang mungkin dari kelas JenisPengguna. Awalnya setelah *state* `login` pada kelas Pengguna diatas berhasil maka *state* akan berpindah ke *state* `status`, jika *state* `status` diberi *event* `get jenis pengguna` maka *state* akan berpindah ke `Jenis Pengguna`. *Event* `get hak akses` dari *state* `jenis pengguna` akan memindahkan *state* ke `hak akses`,

kemudian *event* `run sistem sesuai hak` diaktifkan maka *state* akan berpindah ke *state* `akses sistem`. *event-event* tunggal yang dimiliki sebuah *state* menandakan bahwa state hanya menerima sebuah perintah atau operasi tidak menerima pilihan perintah lain. Pada *state* `logout`, jika pengguna mengaktifkan *event* `login` lagi maka *state* akan berpindah ke *state* awal yaitu `login`. Jika *event* `keluar` diaktifkan, maka *state* akan berpindah ke *finish*.

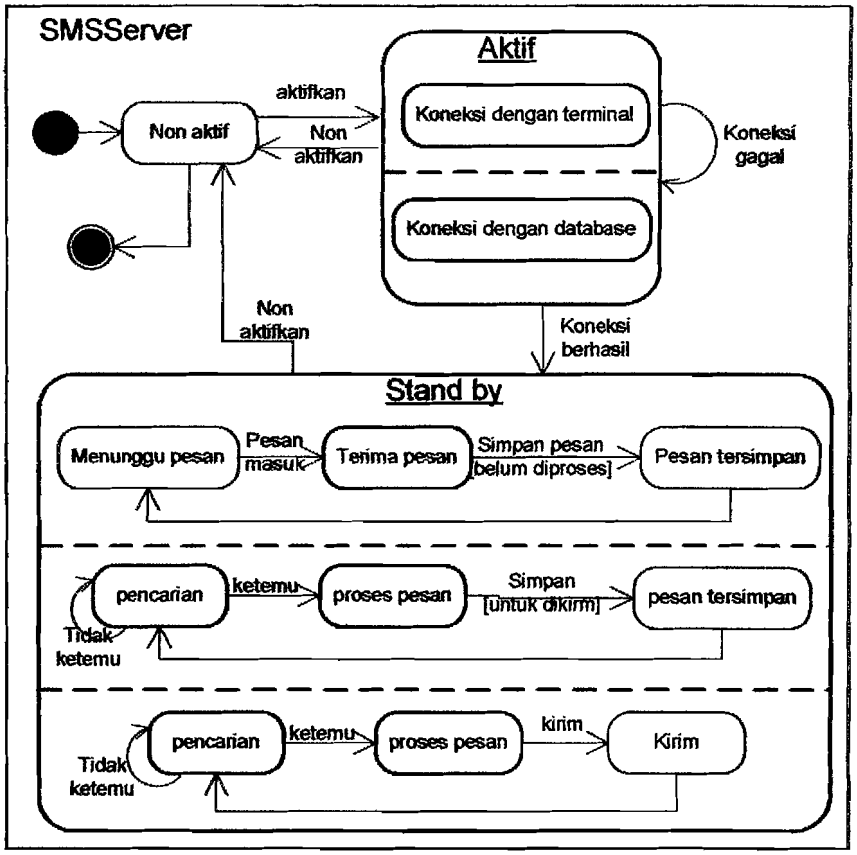


Gambar 3.14 *Statechart* diagram untuk *class* JenisPengguna

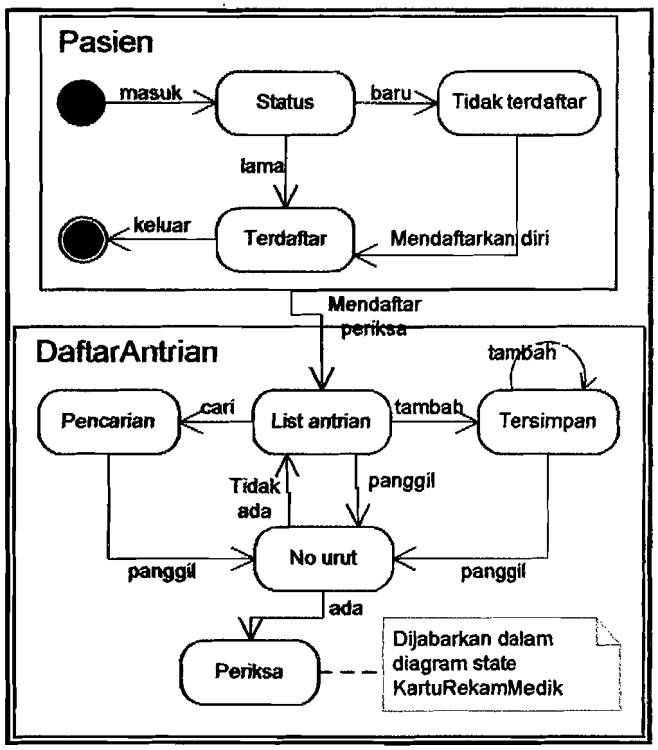
Gambar 3.15 menunjukkan kumpulan *state* dan *event* yang mungkin dari kelas JenisPengguna. Gambaran perpindahan *state* yang terjadi pada kelas tersebut sama dengan penjelasan perpindahan *state* pada kelas-kelas sebelumnya. *State* `stand by` akan mengeksekusi *state* dengan dua cara yakni secara sekuensial untuk *state* dengan posisi horisontal dan secara pararel untuk *state* dengan posisi vertikal.

Gambar 3.16 menunjukkan kumpulan *state* dan *event* yang mungkin dari kelas Pasien dan DaftarAntrian yang digabungkan. Gambar tersebut mempunyai alur yang sama dengan *statechart-statechart* sebelumnya.

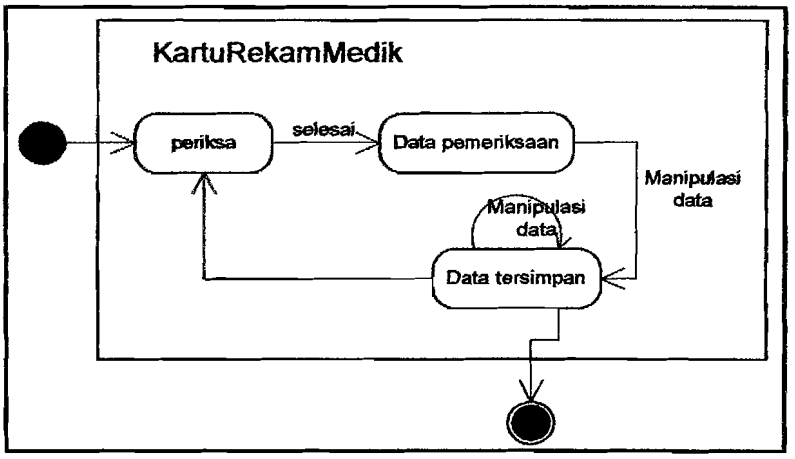
Gambar 3.17 menunjukkan kumpulan *state* dan *event* yang mungkin dari kelas KartuRekamMedik. kelas ini hanya memiliki 3 buah *state* yang mungkin dan 6 buah *event* yang dapat dieksekusi sehingga mengakibatkan perubahan *state*.



Gambar 3.15 *Statechart* diagram untuk *class* SMSServer



Gambar 3.16 *Statechart* diagram untuk *class* Pasien dan DaftarAntrian

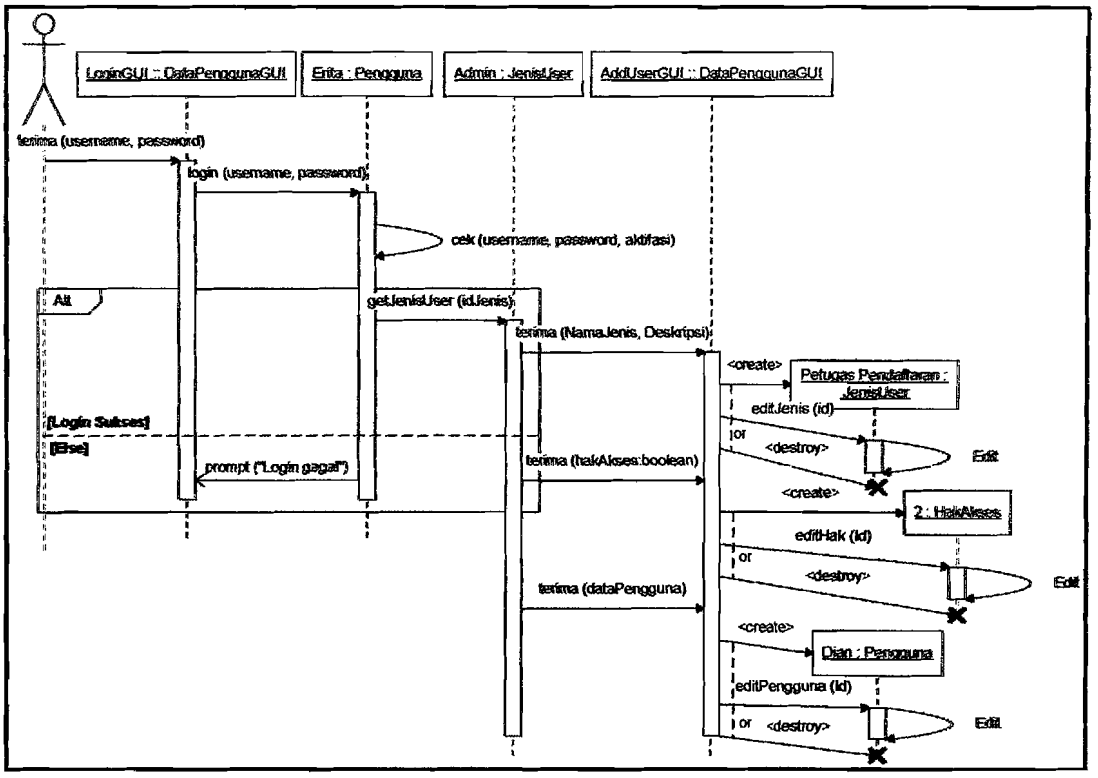


Gambar 3.17 *Statechart* diagram untuk *class* Kartu

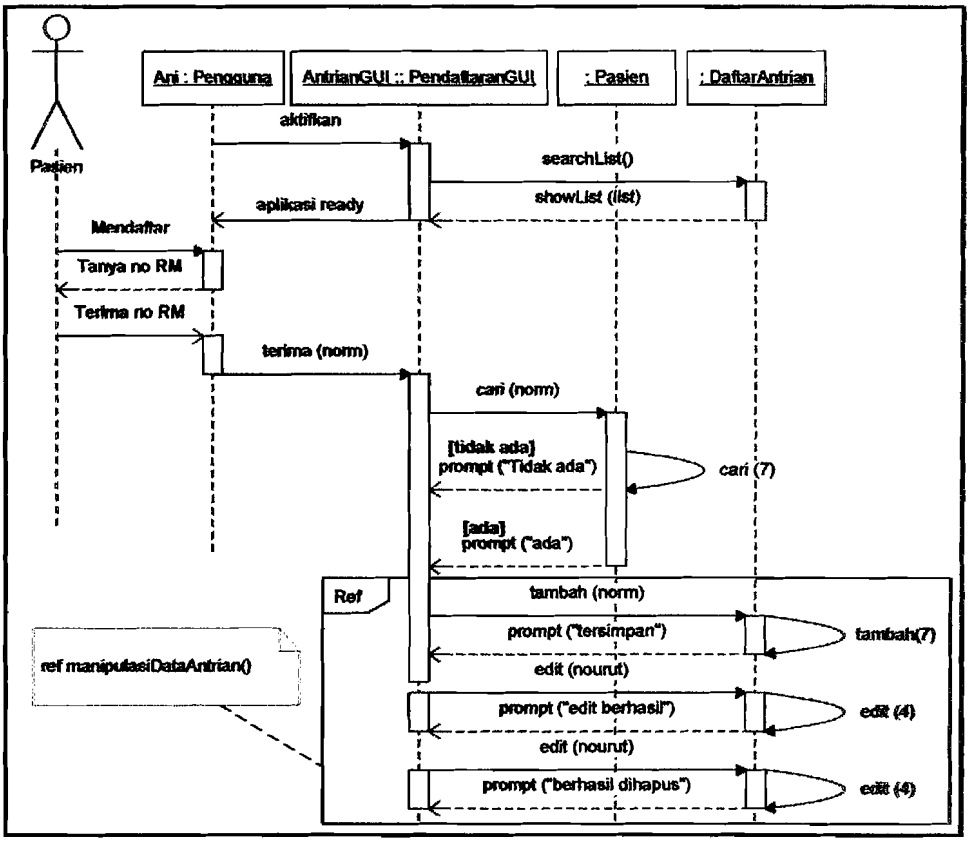
3.2.5 Menentukan interaksi antar obyek

Dalam aksi ini digambarkan bagaimana obyek-obyek saling berinteraksi menggunakan *sequence* diagram dan *collaboration* diagram. *Sequence* diagram digunakan untuk menggambarkan perilaku obyek pada sebuah skenario. Diagram ini menunjukkan sejumlah contoh obyek dan *message* (pesan) yang diletakkan diantara obyek-obyek ini di dalam *use case*. Sedangkan *collaboration* diagram menunjukkan *message-message* obyek yang dikirimkan satu sama lain sehingga memperlihatkan perilaku sistem tesebut. Kedua buah diagram tersebut secara semantik sama, namun diagram *sequence* lebih fokus pada waktu yang dibutuhkan untuk berinteraksi, sedangkan *collaboration* fokus kepada hubungan antar obyek.

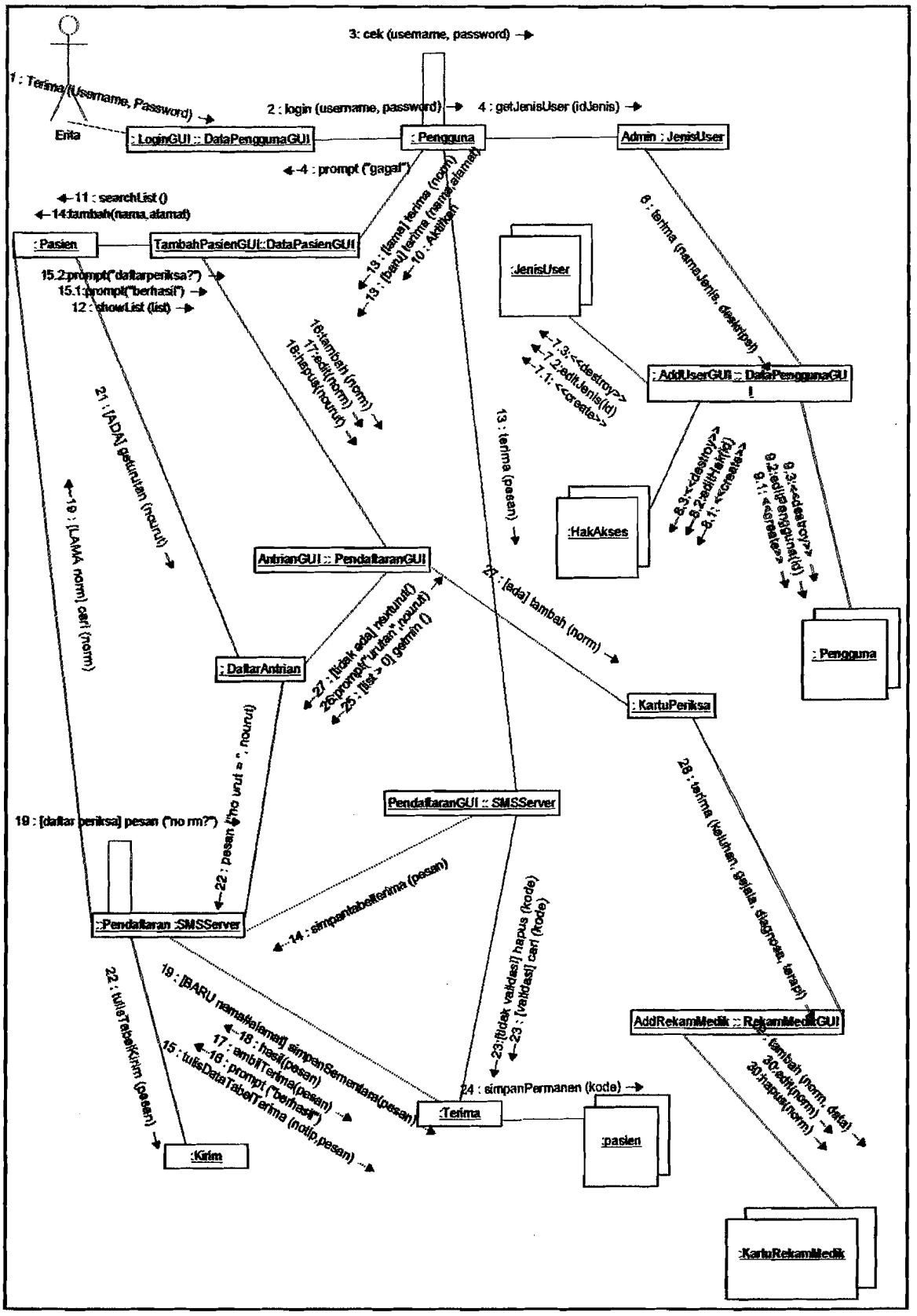
Sesuai dengan *package* diagram pada segmen sebelumnya, pada sistem informasi pendaftaran pasien berbasis SMS ini terdapat tiga buah skenario yaitu skenario proses manipulasi data pengguna gambar 3.18 (dikembangkan dari *package* Pengguna), skenario proses pendaftaran pasien (*package* Pendaftaran) dibagi menjadi empat yaitu skenario proses pendaftaran pasien lama secara langsung (gambar 3.19), proses pendaftaran pasien baru secara langsung (gambar 3.20), skenario proses pendaftaran pasien lama melalui SMS (gambar 3.21), dan proses pendaftaran pasien baru melalui SMS (gambar 3.22), skenario yang terakhir adalah proses manipulasi data rekam medik (*package* RekamMedik). (gambar 3.23)



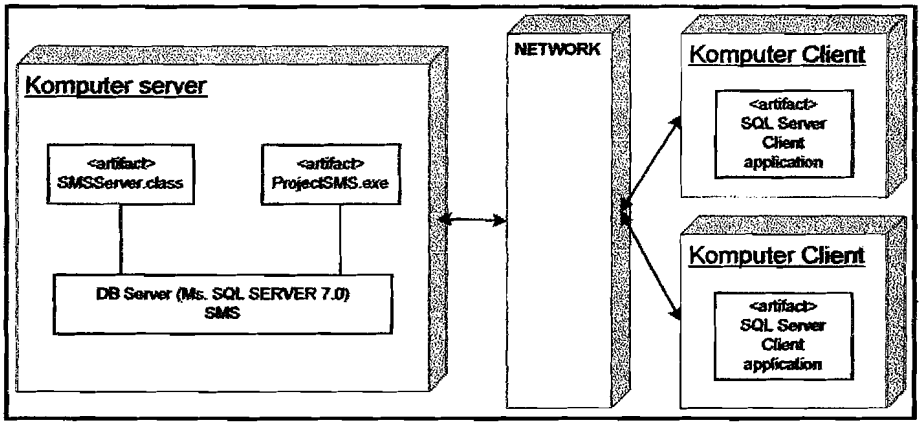
Gambar 3.18 *Sequence* diagram skenario manipulasi data pengguna



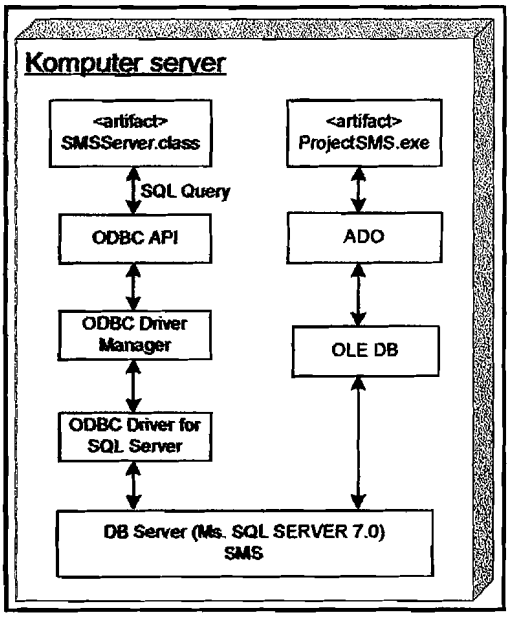
Gambar 3.19 *Sequence* diagram skenario pendaftaran pasien lama langsung



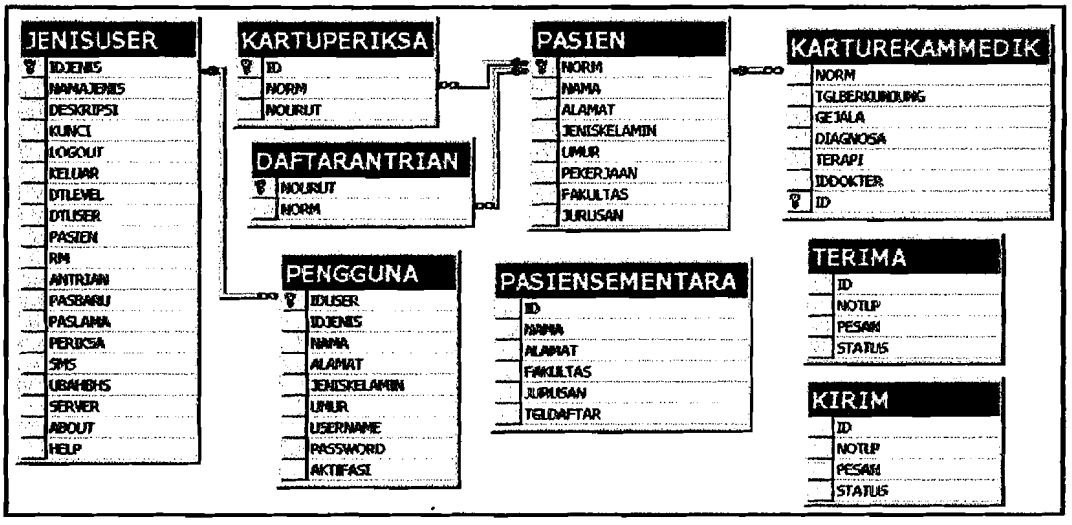
Gambar 3.24 *Collaboration* diagram



Gambar 3.26 Arsitektur jaringan



Gambar 3.27 Arsitektur database secara fisik



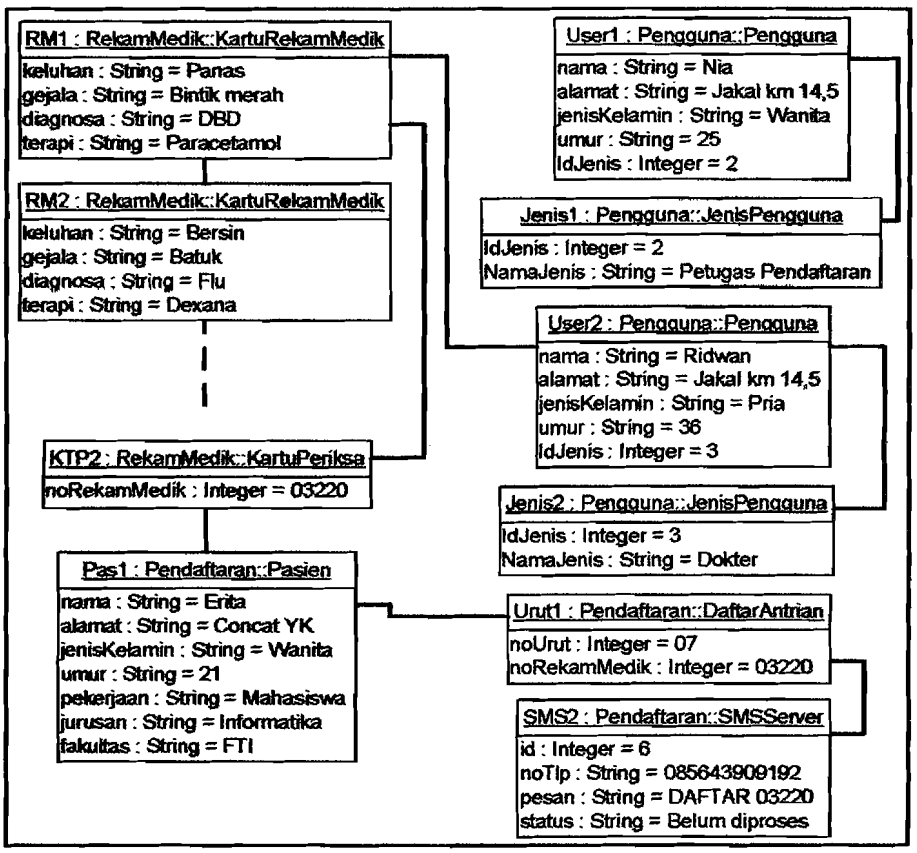
Gambar 3.28 Arsitektur *database* secara logik

3.3 Desain

Dalam segmen ini akan dibahas enam buah aksi dalam mendesain sistem sesuai dengan hasil dari segmen sebelumnya, yaitu: mengembangkan dan menyempurnakan *object* diagram, mengembangkan *component* diagram, perencanaan untuk *deployment* diagram, desain dan *prototype* tampilan antarmuka dan desain uji coba.

3.3.1 Mengembangkan dan menyempurnakan *object* diagram

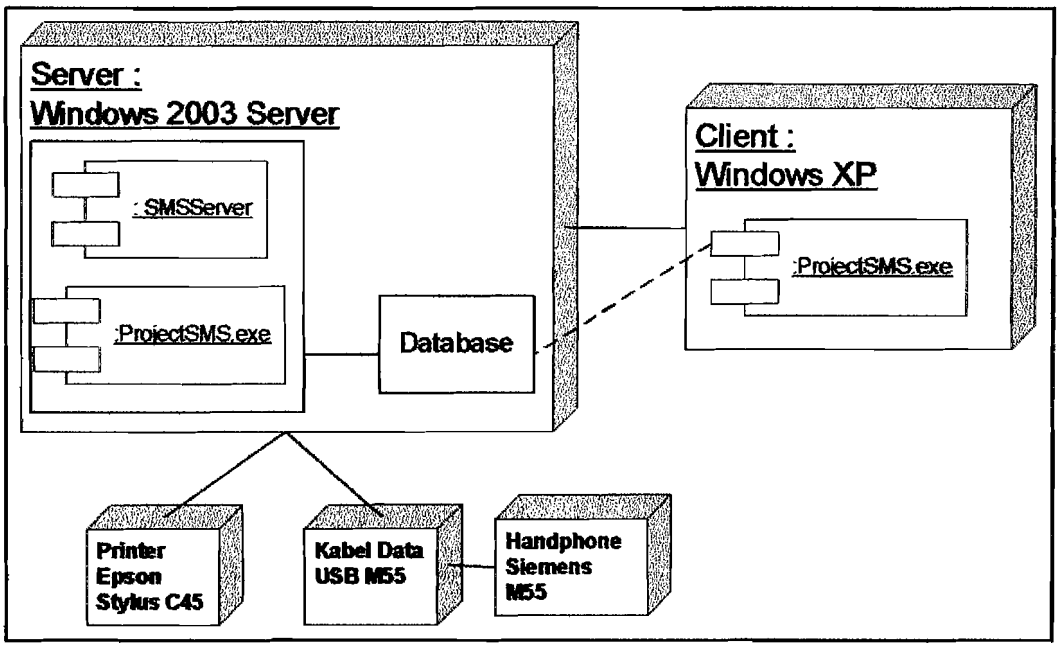
Pada aksi ini dilakukan penyempurnaan pada *object* diagram. *Object* diagram adalah gambaran obyek-obyek secara ringkas di sebuah sistem pada suatu waktu. *Object* diagram sering disebut *instance* diagram karena menunjukkan *instance-instance* dari *class*. *Object* diagram sangat berguna dalam menunjukkan relasi yang kompleks yang terjadi pada suatu *class*. (gambar 3.29)



Gambar 3.29 *Object* Diagram

3.3.3 Perencanaan untuk *deployment* diagram

Ketika *component* diagram telah selesai, aksi selanjutnya adalah merencanakan *deployment* dan integrasi dengan sistem yang dibutuhkan. Hasilnya berupa sebuah diagram yang merupakan bagian dari pengembangan *deployment* diagram sebelumnya. (gambar 3.33)



Gambar 3.33 *Deployment* diagram

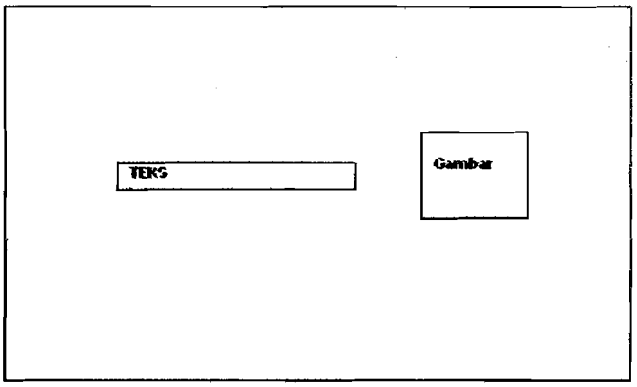
3.3.4 Desain dan *prototype* tampilan antarmuka

Pada pembahasan yang lalu, telah dijelaskan bahwa GRAPPLE mengijinkan hubungan timbal balik yang terjadi antara segmen analisis dan segmen desain yang kontinu sampai ditemukan rancangan aplikasi yang benar-benar tepat, pada aksi inilah proses timbal balik tersebut seringkali terjadi. Hasil dari segmen analisis mempengaruhi hasil dari segmen desain begitu pula sebaliknya.

Pada subsubbab berikut ini akan digambarkan hasil perancangan antarmuka sistem informasi pendaftaran pasien di poliklinik berbasis SMS yang telah benar-benar tepat sesuai dengan hasil dari segmen-segmen sebelumnya.

3.3.4.1 Rancangan antarmuka halaman *splash*

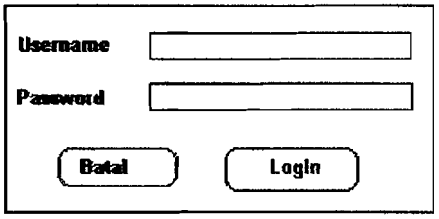
Merupakan halaman yang akan muncul pertama kali ketika aplikasi dijalankan. Halaman *splash* berisi logo gambar atau lambang yang merepresentasikan aplikasi atau perusahaan. (Gambar 3.34)



Gambar 3.34 Rancangan antarmuka halaman *splash*

3.3.4.2 Rancangan antarmuka halaman login

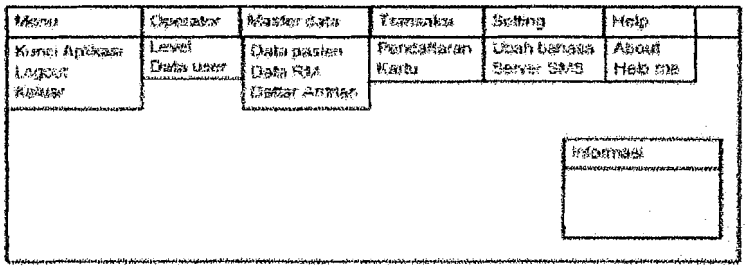
Merupakan halaman otorisasi bagi *user* untuk mengakses halaman selanjutnya sesuai dengan hak-haknya masing-masing. *User* dibedakan menjadi tiga bagian yaitu karyawan, manajer dan admin. (Gambar 3.35)



Gambar 3.35 Rancangan antarmuka halaman login

3.3.4.3 Rancangan antarmuka halaman utama

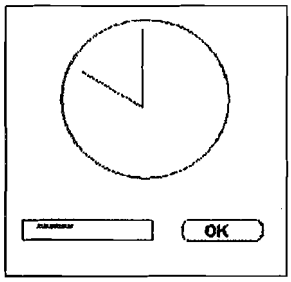
Merupakan halaman yang berisi menu-menu umum untuk memulai menjalankan aplikasi antara lain menu *File*, menu operator, menu master data, menu transaksi, menu *setting* dan menu *help*. Halaman utama akan muncul pertama kali setelah halaman *splash* dan *login*. (Gambar 3.36)



Gambar 3.36 Rancangan antarmuka halaman utama

3.3.4.4 Rancangan antarmuka halaman kunci aplikasi

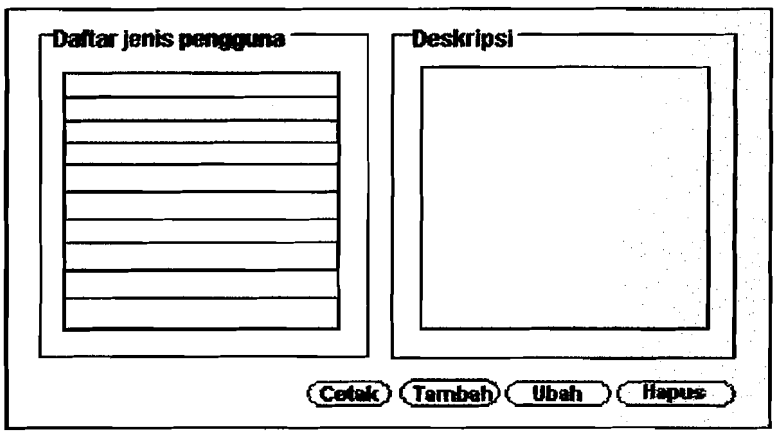
Merupakan halaman yang dipergunakan untuk mengunci aplikasi jika aplikasi akan ditinggalkan sesaat tanpa mematikan aplikasi. (Gambar 3.37)



Gambar 3.37 Rancangan antarmuka halaman kunci aplikasi

3.3.4.5 Rancangan antarmuka halaman jenis pengguna

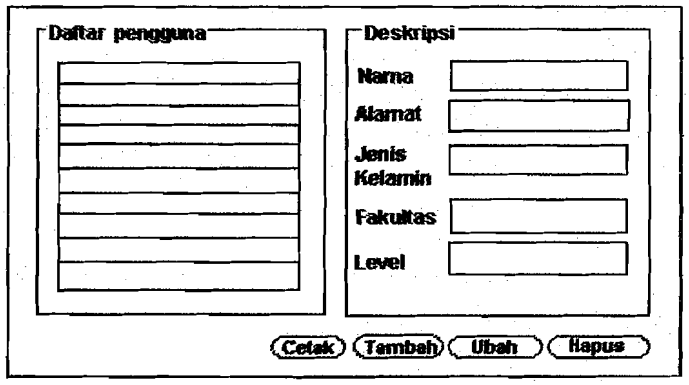
Merupakan halaman yang dipergunakan untuk menambah, mengubah dan mengatur jenis-jenis pengguna yang akan mengakses sistem. (Gambar 3.38)



Gambar 3.38 Rancangan antarmuka halaman jenis pengguna

3.3.4.6 Rancangan antarmuka halaman pengguna

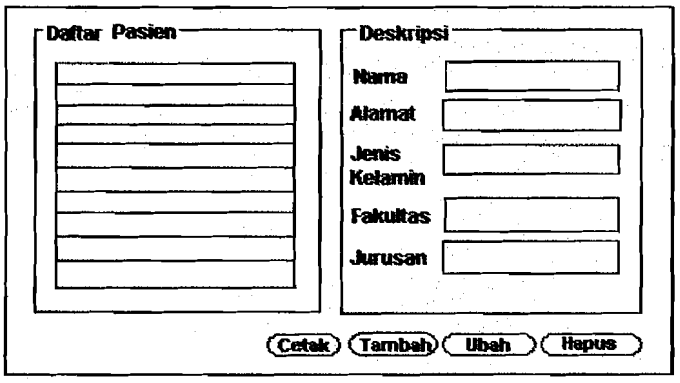
Merupakan halaman yang dipergunakan untuk menambah, mengubah dan mengatur data-data pengguna yang akan mengakses sistem. (Gambar 3.39)



Gambar 3.39 Rancangan antarmuka halaman pengguna

3.3.4.7 Rancangan antarmuka halaman pasien

Merupakan halaman yang dipergunakan untuk menambah, mengubah dan mengatur data-data pasien yang terdaftar di poliklinik. (Gambar 3.40)



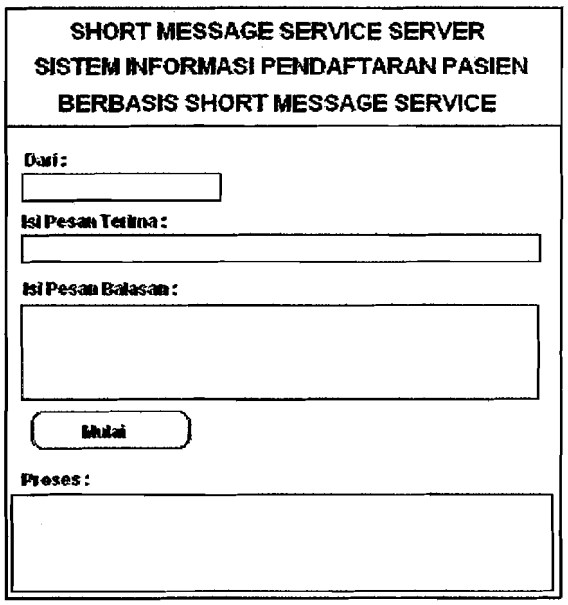
Gambar 3.40 Rancangan antarmuka halaman pasien

3.3.4.8 Rancangan antarmuka halaman rekam medik

Merupakan halaman yang dipergunakan untuk menambah, mengubah dan mengatur data-data rekam medik pasien yang tersimpan dalam *database* poliklinik. (Gambar 3.41)

3.3.4.14 Rancangan antarmuka halaman SMSserver

Halaman ini akan muncul setelah *user* mengaktifkan program utama.java melalui *console*. Halaman ini berisi, *textbox-textbox* untuk menampilkan pesan-pesan yang diterima dan dikirim, juga dilengkapi dengan textbox proses yang akan menampilkan proses-proses yang sedang terjadi, mulai dari koneksi sampai pengiriman SMS balasan. (Gambar 3.47)



Gambar 3.47 Rancangan antarmuka halaman SMS server

3.3.5 Desain uji coba

Tujuan utama dari aksi desain uji coba ini adalah untuk merancang pengujian terhadap sistem, apakah sistem informasi yang dibuat sesuai dengan yang dimaksud. Dalam aksi ini dirancang sebuah desain untuk uji coba sistem informasi tersebut, antara lain :

a. Uji coba sistem dengan meng-input-kan data yang salah atau kosong. Misalnya pada proses login, proses pengisian data, dan lain-lain.
b. Uji coba sistem dengan meng-input-kan data yang benar. Pengujian ini dilakukan juga sebagai pembanding dari pengujian sebelumnya yang dilakukan dengan data yang salah.

3.3.6 Rancangan dokumentasi sistem

Laporan Tugas Akhir merupakan dokumentasi pelaksanaan perancangan dan pembangunan sistem. Sistematika penulisan laporan ini, dibuat dalam bentuk dokumentasi pembangunan perangkat lunak. Rancangan dokumentasi sistem yang merupakan *work product* dari aksi ini menyesuaikan dengan sistematika laporan Tugas Akhir yang dibuat sesuai dengan persetujuan dosen pembimbing. Sesuai dengan laporan, dokumentasi sistem terdiri atas tiga bagian pokok, yaitu bagian pendahuluan, bagian tubuh atau isi laporan, dan bagian akhir.

3.4 Development

Setelah diperoleh rancangan yang benar-benar tepat dari segmen analisis dan desain sebelumnya, maka segmen berikutnya adalah segmen *development* atau segmen pengembangan sistem. Dalam segmen ini akan dibahas empat buah aksi yang akan mengimplementasikan desain sistem menjadi sebuah perangkat lunak yang sesuai dengan hasil perancangan pada segmen sebelumnya, aksi-aksi tersebut antara lain membangun kode, uji coba kode dan membangun antar muka.

3.4.1 Membangun kode

Pembahasan lebih rinci pada aksi ini akan dilakukan kelas per kelas, kelas yang dibentuk dengan Visual Basic 6.0 dibahas paling awal sebab dalam penelitian ini kelas-kelas tersebut merupakan antarmuka untuk memasukkan data yang akan diolah oleh sistem informasi, kemudian pembahasan akan dilanjutkan pada kelas yang dibentuk dengan Java 2 SDK yang berguna untuk melakukan pendaftaran melalui *Short Message Service*.

Sesuai dengan rancangan *class* diagram, *object* diagram, *activity* diagram dan *component* diagram dari segmen sebelumnya, pada aksi ini dilakukan pembangunan kode-kode utama yang diperlukan untuk sebuah sistem.

```
Function tambahPengguna()
Set addpengguna = New Recordset
addpengguna.Open "select * from PENGGUNA", sambung,
adOpenDynamic, adLockOptimistic
With addpengguna
.AddNew
.Fields("IDJENIS") = jnslevel
.Fields("NAMA") = Nama
.Fields("ALAMAT") = Alamat
.Fields("JENISKELAMIN") = JenisKelamin
.Fields("UMUR") = umur
.Fields("USERNAME") = Username
.Fields("PASSWORD") = Password
.Fields("AKTIFASI") = 1
.Update
End With
```

Modul 3.1 *Method* menambahkan data pengguna

```
Function ubahPengguna()
Set editpengguna = New Recordset
editpengguna.Open "update PENGGUNA set PASSWORD='" & Password &
"',UMUR='" & umur & "',JENISKELAMIN='" & JenisKelamin &
"',IDJENIS='" & jnslevel & "',NAMA='" & Nama & "', ALAMAT='" &
Alamat & "' where iduser = '" & idpengguna & "'" & "", sambung,
adOpenDynamic, adLockOptimistic
End Function
```

Modul 3.2 *Method* mengubah data pengguna

```
Public Sub hapusUser()
Set hapuspengguna = New Recordset
With hapuspengguna
.Open "delete PENGGUNA where iduser='" & KodePengguna & "'",
sambung, adOpenDynamic, adLockOptimistic
End With
End Sub
```

Modul 3.3 *Method* menghapus data pengguna

3.4.1.1.2 Kelas JenisPengguna

Kelas ini dibangun untuk menyimpan data jenis-jenis pengguna yang dapat mengakses sistem. Kelas ini membagi pengguna menjadi beberapa jenis, jenis-jenis pengguna tersebut berfungsi untuk membatasi hak akses masing-masing pengguna terhadap sistem yang dibangun. Kelas ini merupakan kelas yang dibangun setelah kelas pengguna karena kelas ini diperlukan untuk proses login pengguna lanjutan. Kelas JenisPengguna mempunyai 18 buah atribut dan 4 buah *method* yang dapat diakses oleh pengguna, antara lain tambahLevel(), ubahLevel(), hapusJenis() dan ubahHak().

a. *Method* tambahLevel()

Method ini dipanggil ketika dilakukan penambahan data jenis pengguna pada kelas JenisPengguna. Atribut yang diperlukan antara lain, namajenis, deskripsi, kunci, logout, keluar, dtlevel, dtuser, pasien, rm, antrian, pasbaru, paslama, sms, ubahbhs, server, about dan help. (Modul 3.4)

b. *Method* ubahLevel()

Method ini dipanggil ketika dilakukan pengubahan data jenis pengguna. Atribut yang diperlukan antara lain idjenis, namajenis dan deskripsi (Modul 3.5)

```
Function tambahLevel()
Set addLevel = New Recordset
addLevel.Open "select * from jenisuser", sambung, adOpenDynamic,
adLockOptimistic
With addLevel
.AddNew
.Fields("NAMAJENIS") = jnslevel
.Fields("DESKRIPSI") = Deskripsi
.Fields("KUNCI") = 1
.Fields("LOGOUT") = 1
.Fields("KELUAR") = 1
.Fields("DTLEVEL") = 0
.Fields("DTUSER") = 0
.Fields("PASIEN") = 0
.Fields("RM") = 0
.Fields("ANTRIAN") = 0
.Fields("PASBARU") = 0
.Fields("PASLAMA") = 0
.Fields("SMS") = 0
.Fields("PERIKSA") = 0
.Fields("UBAHBHS") = 1
.Fields("SERVER") = 1
.Fields("ABOUT") = 1
.Fields("HELP") = 1
.Update
End With
End Function
```

Modul 3.4 *Method* menambahkan data jenis pengguna

```
Function ubahlevel()
Set editlevel = New Recordset
editlevel.Open "update jenisuser set NAMAJENIS='" & jnslevel & "',
DESKRIPSI='" & Deskripsi & "' where idjenis = '" & Kodelevel & "'" &
"", sambung, adOpenDynamic, adLockOptimistic
End Function
```

Modul 3.5 *Method* mengubah data jenis pengguna

c. *Method* hapusJenis()

Method ini dipanggil ketika dilakukan penghapusan data jenis pengguna pada kelas JenisPengguna. Atribut yang diperlukan adalah `iduser`. (Modul 3.6)

```
Public Sub hapusjenis()
Set hapuslevel = New Recordset
With hapuslevel
.Open "delete jenisuser where idjenis='" & Kodelevel & "'",
sambung, adOpenDynamic, adLockOptimistic
End With
End Sub
```

Modul 3.6 *Method* menghapus data jenis pengguna

d. *Method* ubahHak()

Method ini dipanggil ketika dilakukan pengubahan data terhadap hak akses pengguna. Atribut yang diperlukan antara lain `idjenis`, `namajenis` dan `deskripsi` (Modul 3.7)

```
Function ubahhak()
Set edithak = New Recordset
edithak.Open "update jenisuser set KUNCI='" & Kunci & "',
LOGOUT='" & Logout & "', KELUAR='" & Keluar & "', DTLEVEL='" &
DTLevel & "', DTUSER='" & DTUser & "', PASIEN='" & DTPasien &
"', RM='" & rm & "', ANTRIAN='" & Antrian & "', PASBARU='" &
PasBaru & "', PASLAMA='" & pasLama & "', SMS='" & sms & "',
UBAHBHS='" & ubahbhs & "', HELP='" & Help & "', ABOUT='" & About
& "', SERVER='" & Server & "' where idjenis = '" & Kodelevel &
"'" & "", sambung, adOpenDynamic, adLockOptimistic
End Function
```

Modul 3.7 *Method* mengubah data hak akses jenis pengguna

3.4.1.1.3 Kelas Pasien

Kelas ini dibangun untuk menyimpan data diri pasien yang terdaftar di poliklinik tersebut. Kelas ini merupakan kelas yang cukup penting karena kelas ini termasuk kelas yang sangat diperlukan dalam proses pendaftaran. Kelas Pasien mempunyai 8 buah atribut dan 4 buah *method* yang dapat diakses oleh pengguna, antara lain tambahPasien(), ubahPasien(), hapusPasien() dan hapusPasienSementara( ).

a. *Method* tambahPasien()

Method ini dipanggil ketika dilakukan penambahan data diri pasien baru, yaitu pasien yang belum mempunyai nomor rekam medik. Atribut yang diperlukan

antara lain, nama, alamat, jeniskelamin, umur, pekerjaan, fakultas dan jurusan. (Modul 3.8)

```
Function tambahPasien()
Set addpasien = New Recordset
addpasien.Open "select * from PASIEN", sambungan, adOpenDynamic,
adLockOptimistic
With addpasien
.AddNew
.Fields("NAMA") = Nama
.Fields("ALAMAT") = Alamat
.Fields("JENISKELAMIN") = JenisKelamin
.Fields("UMUR") = umur
.Fields("PEKERJAAN") = Pekerjaan
.Fields("FAKULTAS") = Fakultas
.Fields("JURUSAN") = Jurusan
.Update
End With
End Function
```

Modul 3.8 *Method* menambah data pasien

b. *Method* ubahPasien()

Method ini dipanggil ketika dilakukan pengubahan data diri pasien. Atribut yang diperlukan antara lain norm, nama, alamat, jeniskelamin, umur, pekerjaan, fakultas dan jurusan. (Modul 3.9)

```
Function ubahPasien()
Set editpasien = New Recordset
editpasien.Open "update PASIEN set NAMA='" & Nama & "', ALAMAT='"
& Alamat & "', JENISKELAMIN='" & JenisKelamin & "', UMUR='" &
umur & "', PEKERJAAN='" & Pekerjaan & "', FAKULTAS='" & Fakultas
& "', JURUSAN='" & Jurusan & "' where NORM= '" & KodePasien & "'"
& "", sambungan, adOpenDynamic, adLockOptimistic
End Function
```

Modul 3.9 *Method* mengubah data diri pasien

c. *Method* hapusPasien()

Method ini dipanggil ketika dilakukan penghapusan data pasien pada kelas Pasien. Atribut yang diperlukan adalah norm. (Modul 3.10)

```
Public Sub hapusPasien()
Set deletepasien = New Recordset
With deletepasien
.Open "delete PASIEN where NORM='" & KodePasien & "'",
sambungan, adOpenDynamic, adLockOptimistic
End With
End Sub
```

Modul 3.10 *Method* menghapus data pasien

a. *Method* tambahKartuPeriksa()

Method ini dipanggil ketika dilakukan penambahan data pemeriksaan pasien. Atribut yang diperlukan antara lain, `id` dan `norm`. (Modul 3.18)

b. *Method* ubahKartuPeriksa()

Method ini dipanggil ketika dilakukan pengubahan data pemeriksaan pasien. Atribut yang diperlukan antara lain `norm`. (Modul 3.19)

c. *Method* hapusKartuPeriksa()

Method ini dipanggil ketika dilakukan penghapusan pemeriksaan pasien. Atribut yang diperlukan adalah `id`. (Modul 3.20)

```
Public Sub tambahKartuPeriksa()
Set addktp = New ADODB.Recordset
addktp.Open " select * from KARTUPERIKSA ", sambung,
adOpenDynamic, adLockOptimistic
With addktp
    .AddNew
    .Fields("id") = 1
    .Fields("norm") = norm
    .Update
End With
End Sub
```

Modul 3.18 *Method* menambah data pemeriksaan pasien

```
Public Sub ubahKartuPeriksa()
Set editktp = New ADODB.Recordset
With editktp
    .Open "update KARTUPERIKSA set norm='" & norm & "' where id
= '1'", sambung, adOpenDynamic, adLockOptimistic
End With
End Sub
```

Modul 3.19 *Method* mengubah data pemeriksaan pasien

```
Public Sub hapusKartuPeriksa()
Set deletektp = New ADODB.Recordset
With deletektp
.Open "delete KARTUPERIKSA where norm='" & norm & "'", sambung,
adOpenDynamic, adLockOptimistic
End With
End Sub
```

Modul 3.20 *Method* menghapus data pemeriksaan pasien

3.4.1.2 Kelas-kelas yang dibentuk dengan Java 2 SDK

Kelas-kelas yang dibentuk dengan Java 2 SDK ini merupakan kelas yang digunakan untuk menerima, mengirim dan memproses pesan SMS yang masuk

(*SMS Gateway*). Kelas-kelas dalam java ini dapat diaktifkan oleh pengguna dengan hak akses petugas pendaftaran. Kelas yang terbentuk antara lain kelas SMSServer, kelas ProsesJam, kelas ThreadAmbilDataTerima dan kelas ThreadAmbilDataKirim.

3.4.1.2.1 kelas SMSServer

Kelas ini merupakan kelas utama yang berisikan metode-metode yang digunakan dalam program utama SMS server dengan nama kelas ServerSms. Beberapa *method* penting yang terdapat dalam kelas ServerSms antara lain, *method* smsserver, *method* Jam, *method* setDatabase, *method* SetTerminal, *method* balikKarakter, method pduTerimaSms.

a. *Method* smsserver()

Method ini dibuat untuk mengatur *frame*, membuat objek, melakukan pengaturan terhadap komponen-komponen, dll. *Method* ini bertanggung jawab terhadap tampilan GUI dari program. Nama *method* smsserver[1)], *modifier public*.

b. *Method* Jam()

Nama *method* adalah Jam[1)] dengan *modifier public* dan nilai kembaliannya adalah *void*. Method ini digunakan untuk melakukan proses tampilan jam atau waktu yang digunakan sebagai penujuk waktu pada saat program utama dijalankan.

c. *Method* setDatabase()

Proses selanjutnya dari program utama SMS server adalah melakukan koneksi dengan *database*. Proses ini dilakukan agar SMS server dapat melakukan *query* dengan *database*. Sumber data yang digunakan pada SMS server ini disimpan dalam *database* dengan menggunakan SQL Server. Untuk itu dalam penyambungan dengan DBMS menggunakan *driver* JDBC-ODBC Bridge. Nama *method* adalah setDatabase[1)] dengan *modifier public*, dan nilai kembaliannya adalah *void*.

[1)]Modul-modul program diperoleh dari buku berjudul "Pengembangan Aplikasi Sistem Informasi Akademik Berbasis SMS dengan Java" halaman 106-209.

d. *Method* setTerminal()

Proses koneksi dengan terminal dilakukan agar *handphone* yang dijadikan sebagai penerima pesan SMS dapat berkomunikasi dengan program utama SMS server. Perangkat keras yang digunakan pada SMS server adalah *handphone* Siemens M55. Nama *method* adalah setTerminal[1)] dengan *modifier public*, dan nilai kembaliannya adalah *void*.

e. *Method* balikKarakter (String karakter)

Dalam format PDU (*Protocol Data Unit*) ada bagian dimana bilangan desimal diubah dengan menukar posisi atau membalik karakter dari bilangan tersebut. Pada format PDU, proses ini dilakukan pada nomor telepon. Nama *method* adalah balikKarakter[1)] dengan *modifier public*, tipe *method* adalah *static* dengan nilai kembaliannya adalah *string*. Parameter yang diterima oleh *method* ini adalah karakter yang merupakan nilai karakter yang akan dibalik karakternya.

f. *Method* rubahKeHexa()

Format PDU adalah bentuk pesan dalam heksadesimal. Untuk itu diperlukan method yang mengubah dari basis desimal menjadi basis heksadesimal. Proses ini sangat penting untuk penerimaan dan pengiriman pesan SMS. Nama *method* adalah rubahKeHexa[1)] dengan *modifier public*, tipe *method* adalah *static* dengan nilai kembaliannya adalah *string*. Parameter yang diterima oleh *method* ini adalah a bertipe integer yang merupakan nilai karakter yang akan dirubah ke heksadesimal.

g. *Method* PduTerimaSms (String smspdu)

Setiap pesan SMS yang masuk masih dalam bentuk format PDU yang terdiri atas bagian-bagian yang telah diatur dan ditetapkan oleh ETSI. Dalam melakukan proses mengubah pesan SMS PDU menjadi teks yang perlu diperhatikan adalah banyaknya digit dari setiap bagian. Hal ini perlu dilakukan karena kita hanya mengambil nomor pengirim dan isi pesan SMS saja, sedangkan nilai-nilai yang lainnya diabaikan. Nama *method* yang digunakan adalah PduTerimaSms[1)] dengan *modifier public* dan nilai kembaliannya adalah *void*.

[1)]Modul-modul program diperoleh dari buku berjudul "Pengembangan Aplikasi Sistem Informasi Akademik Berbasis SMS dengan Java" halaman 106-209.

method adalah TerimaSMS[1)] dengan *modifier public*, dan nilai kembaliannya adalah *void*. Parameter input yang diterima oleh *method* ini adalah Index dan Pdu.

l. *Method* tulisDataTabelTerima(String notlp, String pesan)

Setiap pesan yang diterima akan langsung disimpan atau dituliskan pada tabel TERIMA yang berfungsi untuk mempermudah dalam melakukan pemilihan atau penyortiran terhadap pesan-pesan SMS yang diterima dan juga untuk menghindari pemrosesan berulang kali terhadap pesan SMS yang sama. Nama *method* adalah tulisDataTabelTerima[1)] dengan *modifier public*, dan nilai kembaliannya adalah *void*. Parameter input yang diterima metode ini adalah *String* notlp yang merupakan nilai nomor telepon tujuan yang didapat dari pesan SMS yang diterima dan *String* pesan yang merupakan isi dari pesan SMS yang diterima.

m. *Method* ambilTerima()

Setelah pesan yang diterima disimpan atau dituliskan pada tabel TERIMA pada *method* ini dilakukan pemilihan atau penyortiran terhadap pesan-pesan SMS yang belum diproses. Nama *method* adalah ambilTerima[1)] dengan *modifier public*, dan nilai kembaliannya adalah *void*.

n. *Method* prosesTabelTerima()

Setelah pesan yang menyaring pesan-pesan yang belum diproses, pada metode ini dilakukan pemrosesan data-data tersebut. Pesan dipisahkan berdasarkan spasi, kemudian diproses sesuai dengan pesan yang diterima. Kata kunci yang diterima oleh sistem antara lain, DAFTAR, CEK, NORM dan BARU, selain keempat kata kunci tersebut sistem akan mengirimkan pesan kesalahan. Nama metode adalah ambilTerima dengan modifier public, dan nilai kembaliannya adalah void. Parameter input yang diterima metode ini adalah String id, String notlp yang merupakan nilai nomor telepon tujuan yang didapat dari pesan SMS yang diterima, dan String pesan yang merupakan isi dari pesan SMS yang diterima. (Modul 3.21)

o. *Method* cek(String tlp)

Method cek berfungsi untuk melakukan pelacakan terhadap nomor rekam

---

[1)]Modul-modul program diperoleh dari buku berjudul "Pengembangan Aplikasi Sistem Informasi Akademik Berbasis SMS dengan Java" halaman 106-209.

```
public void prosesTabelTerima(String id, String notlp, String pesan) {
tulisStatusData(id, "Telah Diproses");
 // Melakukan Parsing (parameter parsing adalah SPASI)
 Pattern pattern = Pattern.compile("#");
 hasil = pattern.split(pesan.trim());
kode1 = hasil[0].trim().toString().toUpperCase(); // Kode pertama
 if (kode1.equals("DAFTAR")) {
String pesan1 = "Apakah anda sudah punya norm? jika sudah ketik
NORM#nomorRekamMedik anda, Jika belum ketik
BARU#nama#alamat#jurusan#fakultas";
      tulisDataTabelKirim(notlp, pesan1);
 } else if(kode1.equals("CEK")) {
cek(notlp);
 } else if (kode1.equals("NORM")) {
      pattern = Pattern.compile("#");
      hasil1 = pattern.split(pesan.trim());
      kode1 = hasil1[0].trim().toString().toUpperCase();
      kode2 = hasil1[1].trim().toString().toUpperCase();
      String K = cari(kode2);
      if (K.equals("ada")) {
      int urutan = daftarAntrian();
String pesan2="data anda tersimpan, no urut anda adalah '" + urutan +
"'. Ketik BATAL ANTRI#noUrut#noRM untuk membatalkan pendaftaran.
Terimakasih";
      tulisDataTabelKirim(notlp, pesan2);
      proses.add(pesan2,++i);
      proses.select(i);
      simpanDiAntrian(urutan, kode2);
      } else {
String pesan3 = "nomor rekam medik "+ kode2 +" tidak ditemukan.Ulangi
pengisian nomor rekam medik";
tulisDataTabelKirim(notlp, pesan3);
proses.add("pencarian gagal.norm tidak ketemu!",++i);
proses.select (i);
}      } else if (kode1.equals("BATAL ANTRI")) {
pattern = Pattern.compile("#");
hasil1 = pattern.split(pesan.trim());
kode1 = hasil1[0].trim().toString().toUpperCase();
kode2 = hasil1[1].trim().toString().toUpperCase();
kode3 = hasil1[2].trim().toString().toUpperCase();
batalAntri(kode2, kode3, notlp);
} else if (kode1.equals("BATAL DAFTAR")) {
pattern = Pattern.compile("#");
hasil1 = pattern.split(pesan.trim());
kode1 = hasil1[0].trim().toString().toUpperCase();
kode2 = hasil1[1].trim().toString().toUpperCase();
batalDaftar(kode2, notlp);
} else if (kode1.equals("BARU")) {
      pattern = Pattern.compile("#");
      hasil1 = pattern.split(pesan.trim());
      koden = hasil1[0].trim().toString().toUpperCase();
      nama = hasil1[1].trim().toString().toUpperCase();
      alamat = hasil1[2].trim().toString().toUpperCase();
      jur    = hasil1[3].trim().toString().toUpperCase();
      fak    = hasil1[4].trim().toString().toUpperCase();
      PasienSementara simpanSM = new PasienSementara();
      simpanSM.simpanDiPasienSM(notlp, nama, alamat, jur, fak);
      simpanSM.daftarPasienBaru(notlp, nama, alamat, fak, jur);
    } else {
kode_1_salah = "Kode1 yang Anda Masukkan Salah" + "\nkode : " + kode1 +
          "\nTidak terdaftar";
kode_1_format = "\nKode1 yang diijinkan adalah DAFTAR / NORM / BARU";
pesanSalahKode_1 = kode_1_salah + kode_1_format;
tulisDataTabelKirim(notlp, pesanSalahKode_1);
    } // Akhir else
  } // Akhir Methode prosesTabelTerima
```

Modul Program 3.21 *Method* prosesTabelTerima

q. *Method* daftarAntrian()

Method daftarAntrian berfungsi untuk memperoleh nilai tertinggi dari antrian. Jika antrian kosong maka no urut adalah 1. Jika antrian tidak kosong, maka nomor urut adalah nomor urut terakhir ditambah 1. Nama metode adalah daftarAntrian dengan modifier public, dan nilai kembaliannya adalah int. (Modul 3.24)

```
public String cari(String kod2) {
String A = null;
sql = "Select * from PASIEN where NORM = '" + kod2 + "'";
      try {
            pStatement = koneksi.prepareStatement(sql);
            try {
                  ResultSet rSet = pStatement.executeQuery();
                  if (rSet.next()) {
                  proses.add("pencarian berhasil.norm ketemu!",++i);
                        proses.select (i);
                        A = "ada";
                  } else {
                        A = "tidak ada";
                  }
                  pStatement.close();
            } // Akhir try ResultSet
            catch (Exception ie) {}
      } // Akhir try pstatement
      catch (Exception ie) {}
return A;
}
```

Modul Program 3.23 *Method* cari

```
public int daftarAntrian() {
int urut = 0;
try {
Statement stat =
koneksi.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,ResultSet.CO
NCUR_READ_ONLY);
ResultSet rs = stat.executeQuery ("select * from DAFTARANTRIAN order
by nourut");
            if (rs.next()) {
                  rs.last();
                  urut = rs.getInt("NOURUT");
                  ++urut;
            }
            else {
            urut = 1;
            }
            stat.close();
      }
      catch (Exception e){}
return urut;
}//akhir method daftarantrian
```

Modul Program 3.24 *Method* daftarAntrian

r. *Method* simpanDiAntrian(int urt, String norm)

Method simpanDiAntrian berfungsi untuk menyimpan nomor rekam medik pada daftar antrian. Nama *method* adalah simpanDiAntrian dengan modifier public, dan nilai kembaliannya adalah void. Parameter input yang diterima *method* ini adalah int urt yang merupakan nomor urut pasien diperoleh dari *method* daftarAntrian dan String norm yang merupakan nilai nomor rekam medik pasien yang didapat dari pesan SMS yang diterima. (Modul 3.25)

```
  public void simpanDiAntrian(int urt, String norm) {
try {
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver"); //Drive Ms. Access
Connection  koneksi = DriverManager.getConnection("jdbc:odbc:TASMSKU",
"", "");

Statement stat = koneksi.createStatement();
String sql="insert into DAFTARANTRIAN(NOURUT,NORM) values('" +
                    urt + "','" +
                    norm + "')";

       stat.executeUpdate(sql);
        proses.add("berhasil menyimpan di antrian", ++i);
       proses.select(i);
    }
    catch (Exception e) {}
  } // Akhir Methode simpanDiAntrian
```

Modul Program 3.25 *Method* simpanDiAntrian

s. *Method* batalAntri(String urut, String norm, String ntlp)

Method batalAntri berfungsi untuk menghapus nomor rekam medik pada daftar antrian. Nama *method* adalah batalAntri dengan *modifier public* dan nilai kembaliannya adalah *void*. Parameter input yang diterima *method* ini adalah String urut yang merupakan nomor urut pasien dan String norm yang merupakan nilai nomor rekam medik pasien. (Modul 3.26)

```
public void batalAntri(String urut, String norm, String ntlp) {
sql = "Delete DAFTARANTRIAN where NOURUT = '" + urut + "' and NORM
= '" + norm + "'";
       try {
              pStatement = koneksi.prepareStatement(sql);
              try {
                     ResultSet rSet = pStatement.executeQuery();
                     pStatement.close();
              } // Akhir try ResultSet
              catch (Exception ie) {}
       } // Akhir try pstatement
       catch (Exception ie) {}
String pesan5 = "Data antrian berhasil dihapus!";
tulisDataTabelKirim(ntlp, pesan5); }
```

Modul Program 3.26 *Method* batalAntri

t. *Method* batalDaftar(String kodeVerifikasi, String ntlp)

Method batalAntri berfungsi untuk menghapus nomor rekam medik pada daftar antrian. Nama *method* adalah batalAntri dengan *modifier public* dan nilai kembaliannya adalah *void*. Parameter input yang diterima *method* ini adalah String urut yang merupakan nomor urut pasien dan String norm yang merupakan nilai nomor rekam medik pasien. (Modul 3.27)

```
public void batalDaftar(String kodeVerifikasi, String ntlp) {

        sql = "Delete PASIENSEMENTARA where ID = '" + kodeVerifikasi
+ "'";
        try {
                pStatement = koneksi.prepareStatement(sql);
                try {
                        ResultSet rSet = pStatement.executeQuery();
                        // Menutup Statement SQL
                        pStatement.close();
                } // Akhir try ResultSet
                catch (Exception ie) {}
        } // Akhir try pstatement
        catch (Exception ie) {}
String pesan6 = "Data pasien baru berhasil dihapus!";
tulisDataTabelKirim(ntlp, pesan6);
}
```

Modul Program 3.26 *Method* batalAntri

u. *Method* tulisDataTabelKirim()

Setiap pesan yang akan dikirim sebelumnya disimpan atau dituliskan pada tabel KIRIM untuk mempermudah dalam melakukan pemilihan terhadap pesan-pesan SMS yang akan dikirim dan juga untuk menghindari pengiriman yang berulang kali terhadap pesan SMS yang sama. Nama *method* adalah tulisDataTabelKirim[1)] dengan *modifier public*, dan nilai kembaliaanya adalah *void*. Parameter input yang diterima metode ini adalah *String* notlp yang merupakan nilai nomor telepon tujuan yang didapat dari pesan SMS yang diterima, dan *String* dataKirim yang merupakan isi dari pesan SMS yang akan dikirim.

v. *Method* ambilKirim()

Pesan SMS yang akan dikirim yang terdapat pada tabel KIRIM akan memiliki status "Belum Dikirim". Untuk itu, dilakukan pengambilan terhadap data

[1)]Modul-modul program diperoleh dari buku berjudul "Pengembangan Aplikasi Sistem Informasi Akademik Berbasis SMS dengan Java" halaman 106-209.

dengan status "Belum Dikirim" untuk dilakukan proses pengiriman pesan tersebut. Nama method adalah ambilKirim[1)] dengan *modifier public*, dan nilai kembaliannya adalah *void*.

w. *Method* public void prosesKirimSms(String id, String notlp, String pesan)

Method ini menangani proses pengiriman pesan yang diinputkan ke nomor yang diinginkan. Nama method adalah prosesKirimSms[1)] dengan *modifier public*, dan nilai kembaliannya adalah *void*. Parameter input yang diterima *method* ini adalah String id, String notlp dan String pesan

3.4.1.2.2 kelas ProsesJam extends Thread

Sifat dari tampilan waktu adalah tetap atau tidak mengalami perubahan saat program SMS Server dijalankan. Sehingga ketika program dijalankan nilai dari waktu tidak pernah berubah. Oleh sebab itu, kelas ini dibuat untuk menangani proses update tampilan waktu. Kelas ini akan selalu memanggil method Jam() agar nilai detik dari jam akan selalu terup-date atau berubah.

Karena proses pemanggilan metode Jam() dilakukan secara terus menerus dan dilakukan bersamaan dengan proses yang lain, maka kelas ini akan menggunakan metode Thread. Penggunaan metode Thread ini dilakukan agar program SMS Server dapat melakukan beberapa proses pada waktu yang bersamaan sehingga kinerja program SMS Server lebih optimal. Proses yang dilakukan pada kelas Thread ini adalah selalu memanggil method Jam() untuk selalu mengup-date nilai jam atau waktu. Proses pemanggilan method Jam() ini dilakukan secara terus-menerus dan dapat dilakukan bersamaan dengan proses lainnya.

3.4.1.2.3 kelas ThreadAmbilDataTerima extends Thread[1)]

Pada sistem informasi ini, setiap pesan SMS yang masuk akan disimpan ke

[1)]Modul-modul program diperoleh dari buku berjudul "Pengembangan Aplikasi Sistem Informasi Akademik Berbasis SMS dengan Java" halaman 106-209.

tabel TERIMA dengan STATUS "Belum Diproses". Sedangkan untuk pesan yang telah diproses akan mempunyai STATUS "Telah Diproses". Hal ini dilakukan agar sistem dapat membedakan antara data pesan SMS yang telah diproses dengan data pesan SMS yang baru diterima atau yang belum diproses.

Untuk itu perlu dilakukan pengecekan terhadap data-data pesan SMS yang terdapat pada tabel TERIMA yang ber-STATUS "Belum Diproses". Jika data yang dimaksud ditemukan, maka dilakukan pengambilan terhadap pesan-pesan SMS tersebut untuk kemudian diproses lebih lanjut. Proses pengecekan ini dilakukan secara terus menerus selama sistem mengaktifkan program SMS Server.

3.4.1.2.4 kelas ThreadAmbilDataKirim extends Thread[1)]

Pada sistem informasi ini, setiap pesan SMS yang akan dikirim ditulis atau disimpan ke tabel KIRIM dengan STATUS "Belum Dikirim". Sedangkan untuk pesan yang telah berhasil dikirim akan mempunyai STATUS "Telah Dikirim". Hal ini dilakukan agar sistem dapat membedakan antara data pesan SMS yang telah dikirim dengan data pesan SMS yang belum dikirim.

Untuk itu perlu dilakukan pengecekan terhadap data-data pesan SMS yang terdapat pada tabel KIRIM yang ber-STATUS "Belum Dikirim". Jika data yang dimaksud ditemukan, maka dilakukan pengambilan terhadap pesan-pesan SMS tersebut untuk kemudian dilakukan proses pengiriman. Proses pengecekan ini dilakukan secara terus menerus selama sistem mengaktifkan program SMS Server.

3.4.1.2.5 kelas PasienSementara

Kelas ini merupakan kelas yang dibangun untuk memanipulasi data-data

---

[1)]Modul-modul program diperoleh dari buku berjudul "Pengembangan Aplikasi Sistem Informasi Akademik Berbasis SMS dengan Java" halaman 106-209.

dalam kelas PasienSementara. Kelas ini diperlukan untuk menyimpan data pasien yang terdaftar melalui SMS secara sementara. Method-method yang digunakan dalam kelas ini antara lain, method simpanDiPasienSM dan method daftarPasienBaru.

a. *Method* simpanDiPasienSM()

Nama metode adalah simpanDiPasienSM dengan *modifier public* dan nilai kembaliannya adalah *void*. Metode ini digunakan untuk menyimpan data pasien baru secara sementara di tabel PasienSementara. (Modul 3.27)

```
public void simpanDiPasienSM(String ntlp, String n, String a,
String j, String f) |
try |
pStatement = koneksi.prepareStatement(
"Insert Into PASIENSEMENTARA (NAMA,ALAMAT,FAKULTAS,JURUSAN)
Values (?,?,?,?)");
      try |
       pStatement.setString(1, n);
       pStatement.setString(2, a);
       pStatement.setString(3, f);
       pStatement.setString(4, j);
        pStatement.executeUpdate();     |
      catch (Exception e) {|
      pStatement.close();     |
    catch (Exception e) {|
    } // akhir method simpanDiPasienSM
```

Modul Program 3.27 *Method* simpanDiPasienSM

b. *Method* daftarPasienBaru()

Proses selanjutnya dari program utama SMS server adalah melakukan koneksi dengan database. Proses ini dilakukan agar SMS server dapat melakukan *query* dengan database. Sumber data yang digunakan pada SMS server ini disimpan dalam database dengan menggunakan SQL Server. Untuk itu dalam penyambungan dengan DBMS menggunakan driver JDBC-ODBC Bridge. Nama metode adalah setDatabase dengan modifier public, dan nilai kembaliannya adalah void. (Modul 3.28)

```
public void daftarPasienBaru(String notlp, String na, String al,
String fa, String ju) {
sql = "select ID from PASIENSEMENTARA where NAMA = ? and ALAMAT
= ? and FAKULTAS = ? and JURUSAN = ?";
      try {
       pStatement = koneksi.prepareStatement(sql);
      try {
       pStatement.setString(1, na);
       pStatement.setString(2, al);
       pStatement.setString(3, fa);
       pStatement.setString(4, ju);
       ResultSet rSet = pStatement.executeQuery();
        if (rSet.next()) {
                    int kodepas = rSet.getInt("ID");
String pesan3="data anda telah tersimpan, kode verifikasi anda
adalah '" + kodepas + "', batas waktu verifikasi adalah 3
hari.";
       tulisDataTabelKirim(notlp, pesan3);     }
        pStatement.close();        }
       catch (Exception ie) {}     }
       catch (Exception ie) {}
    } //Akhir method daftarPasienBaru
```

Modul Program 3.28 *Method* daftarPasienBaru

3.4.2 Uji coba kode

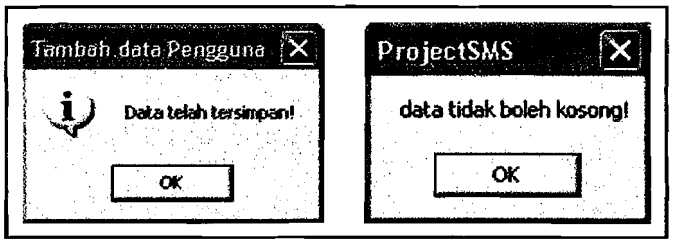
Kode-kode program yang telah dibangun sebelumnya, perlu dilakukan pengujian apakah kode-kode tersebut dapat berjalan sebagaimana seharusnya. Pengujian terhadap kode-kode diatas dilakukan melalui dua cara yaitu, pengujian dengan data yang benar dan pengujian dengan data yang salah. Pada aksi ini terjadi hubungan timbal balik dengan aksi sebelumnya hingga kode benar-benar berjalan sesuai dengan rancangan pada segmen desain.

Berikut ini dua buah script untuk menguji kode kelas pengguna serta hasil pengujian. Script kiri digunakan untuk pengujian data benar, script sebelah kanan untuk pengujian data salah. (Modul 3.29 dan gambar 3.48)

```
With clspengguna
   .Koneksi
   .jnslevel ="2"
   .Nama = "Erita Yuliastuti"
   .Alamat = "Pr. Puri Permata II/3"
   .JenisKelamin = "Wanita"
   .umur = "21"
   .Username = "Erita"
   .Password = "Erita"
   .tambahPengguna
End With
```

```
With clspengguna
   .Koneksi
   .jnslevel =""
   .Nama = ""
   .Alamat = ""
   .JenisKelamin = ""
   .umur = ""
   .Username = ""
   .Password = ""
   .tambahPengguna
End With
```

Modul Program 3.29 *Script* pengujian data untuk kelas pengguna



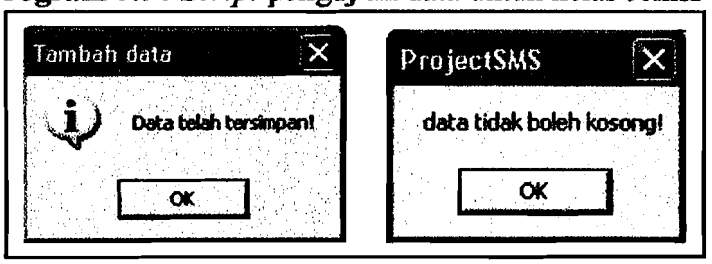
Gambar 3.48 Hasil pengujian script kelas pengguna

Berikut ini dua buah script untuk menguji kode kelas JenisPengguna serta hasil pengujian. Script kiri digunakan untuk pengujian data benar, script sebelah kanan untuk pengujian data salah. (Modul 3.30 dan gambar 3.49)

```
With clsLevel
.Koneksi
.jnslevel = "2"
.Deskripsi = "ADMINISTRATOR"
.tambahLevel
End With
```

```
With clsLevel
.Koneksi
.jnslevel = ""
.Deskripsi = ""
.tambahLevel
End With
```

Modul Program 3.30 *Script* pengujian data untuk kelas JenisPengguna



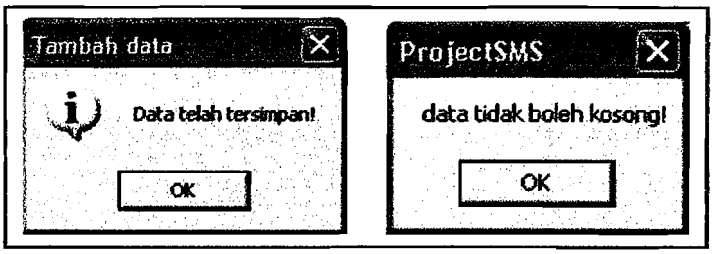
Gambar 3.49 Hasil pengujian script kelas JenisPengguna

Berikut ini dua buah script untuk menguji kode kelas Pasien serta hasil pengujian. Script kiri digunakan untuk pengujian data benar, script sebelah kanan untuk pengujian data salah. (Modul 3.31 dan gambar 3.50)

```
With clspasien
.Koneksi
.Nama = "Erita Yuliastuti"
.Alamat = "Condong catur"
.JenisKelamin ="Wanita"
.umur ="21"
.Pekerjaan = "Mahasiswa"
.Fakultas = "fti"
.Jurusan = "informatika"
.tambahPasien
End With
```

```
With clspasien
.Koneksi
.Nama = ""
.Alamat = ""
.JenisKelamin =" "
.umur =""
.Pekerjaan = ""
.Fakultas = ""
.Jurusan = ""
.tambahPasien
End With
```

Modul Program 3.31 *Script* pengujian data untuk kelas Pasien



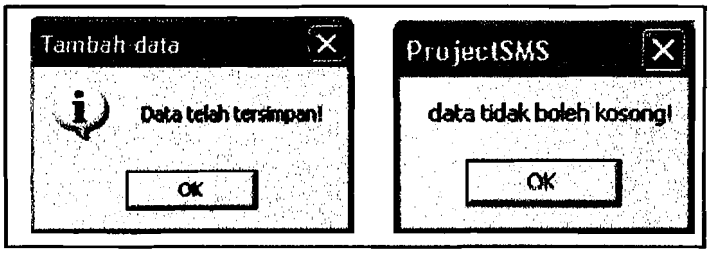
Gambar 3.50 Hasil pengujian script kelas Pasien

Berikut ini dua buah script untuk menguji kode kelas KartuRekamMedik serta hasil pengujian. Script kiri digunakan untuk pengujian data benar, script sebelah kanan untuk pengujian data salah. (Modul 3.32 dan gambar 3.51)

```
With clsrm
.Koneksi
.KodeRM = "2"
.Tanggal = "12/02/2007"
.keluhan = "panas"
.Diagnosa = "flue"
.Terapi = "Parasetamol"
.KodeDokter = "3"
.tambahRM
End With
```

```
With clsrm
.Koneksi
.KodeRM = ""
.Tanggal = ""
.keluhan = ""
.Diagnosa = ""
.Terapi = ""
.KodeDokter = ""
.tambahRM
End With
```

Modul Program 3.32 *Script* pengujian data untuk kelas KartuRekamMedik



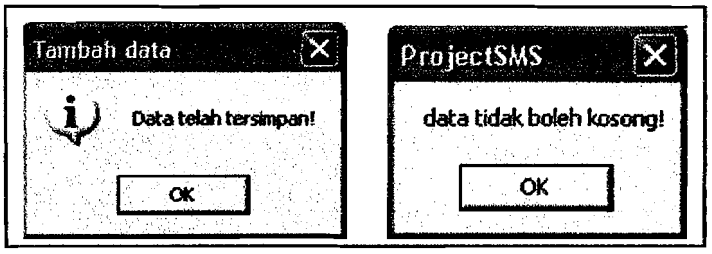
Gambar 3.51 Hasil pengujian script kelas KartuRekamMedik

Berikut ini dua buah script untuk menguji kode kelas DaftarAntrian serta hasil pengujian. Script kiri digunakan untuk pengujian data benar, script sebelah kanan untuk pengujian data salah. (Modul 3.33 dan gambar 3.52)

```
With clsAntrian
.Koneksi
.norm = "2"
.nourut = "1"
.tambahAntrian
End With
```

```
With clsAntrian
.Koneksi
.norm = ""
.nourut = ""
.tambahAntrian
End With
```

Modul Program 3.33 *Script* pengujian data untuk kelas DaftarAntrian



Gambar 3.52 Hasil pengujian script kelas DaftarAntrian

Berikut ini dua buah script untuk menguji kode kelas SMSServer serta hasil pengujian. Script kiri digunakan untuk pengujian data benar, script sebelah kanan untuk pengujian data salah. (Modul 3.34).

Sintaks SMS	Contoh	Pesan Balasan yang Diperoleh
DAFTAR	DAFTAR	Apakah anda sudah punya norm? jika sudah ketik NORM#nomorRekamMedik anda, Jika belum ketik BARU#nama#alamat#jurusan#fakultas. Untuk mengecek antrian ketik cek
CEK	CEK	Nomor urut yang sedang diperiksa adalah 2, dari 14 buah antrian.
NORM#NomorRekamMedik	NORM#5	Data anda tersimpan, no urut anda adalah '" + urutan + "'. Ketik BATAL ANTRI#noUrut#noRM untuk membatalkan pendaftaran. Terimakasih
BARU#Nama#Alamat#Jurusan#Fakultas	BARU#ERITA YULIASTUTI#PR. PURI PERMATA 2/3 CC YK#INFORMATIKA #FTI	Data anda telah tersimpan, kode verifikasi anda adalah '9', batas waktu verifikasi adalah 3 hari. Untuk membatalkan ketik BATAL DAFTAR#KodeVerifikasi
BATAL ANTRI#NoUrut #NoRM	BATAL ANTRI#5#7	Data antrian berhasil dihapus!
BATAL DAFTAR#Kode Verifikasi	BATAL DAFTAR#7	Data pasien baru berhasil dihapus!

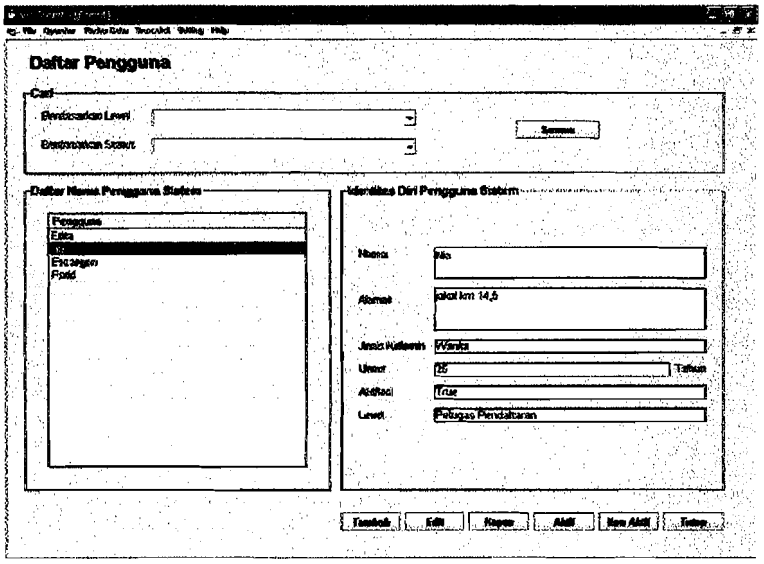
Modul Program 3.34 *Script*dan hasil pengujian data untuk kelas SMSServer

3.4.3 Membangun antarmuka

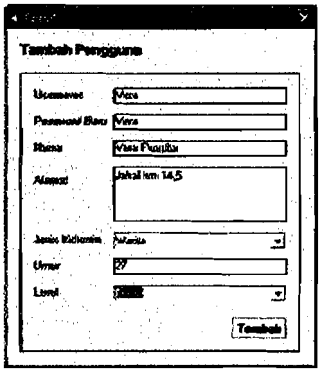
Pada aksi ini dilakukan pembangunan antarmuka yang diperlukan sebagai *user interface* bagi pengguna dalam mengeksekusi kode-kode diatas dengan lebih mudah. Antarmuka yang telah dibangun kemudian dikoneksikan dengan kode-kode yang telah dibangun sebelumnya.

3.4.3.1 Antarmuka kelas Pengguna

Merupakan halaman yang dipergunakan untuk menambah, mengubah dan mengatur data-data pengguna yang akan mengakses sistem. Pengguna dapat dilihat berdasarkan kelompok jenis maupun kelompok aktifasi. (Gambar 3.53 dan Gambar 3.54)



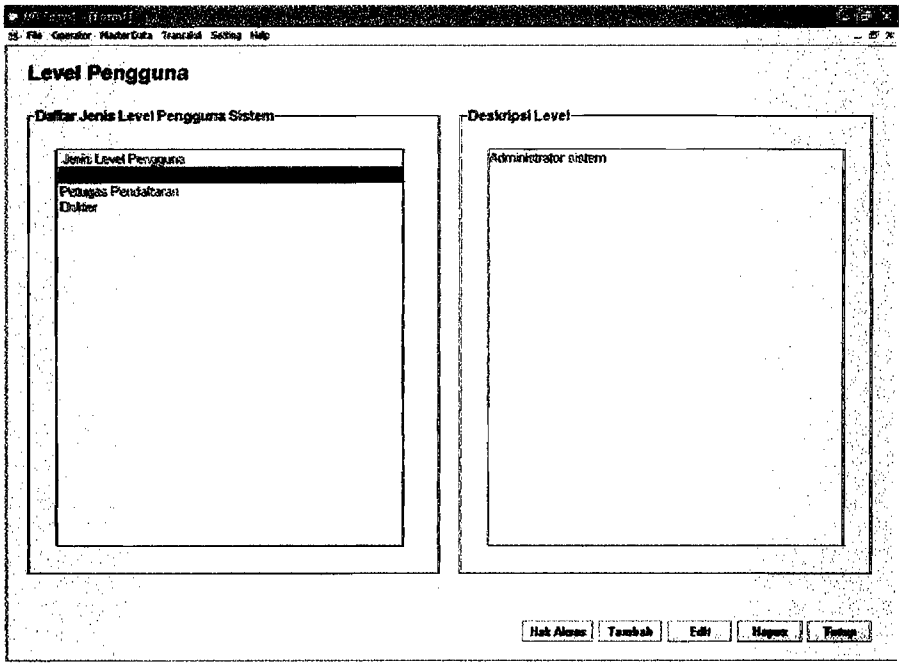
Gambar 3.53 Antarmuka halaman pengguna



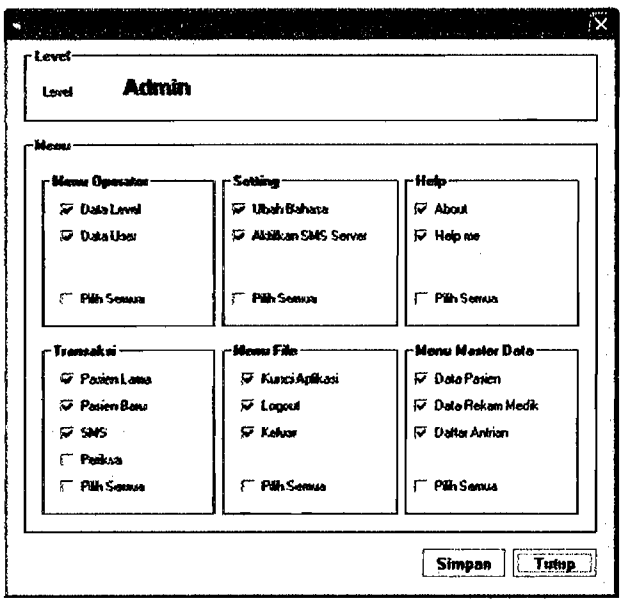
Gambar 3.54 Antarmuka untuk menambah dan mengubah data pengguna

3.4.3.2 Antarmuka kelas JenisPengguna

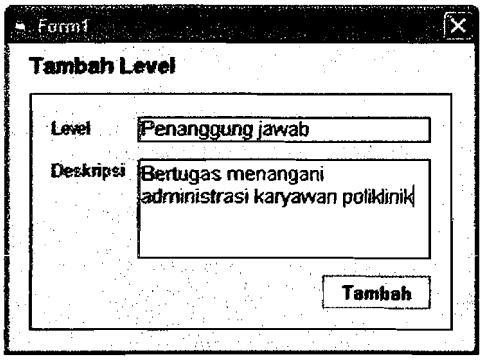
Merupakan halaman yang dipergunakan untuk menambah, mengubah dan mengatur data-data jenis pengguna yang mengakses sistem. Setiap jenis pengguna dapat diatur hak-hak aksesnya melalui halaman hak akses. (Gambar 3.55, Gambar 3.56 dan Gambar 3.57)



Gambar 3.55 Antarmuka halaman pengguna



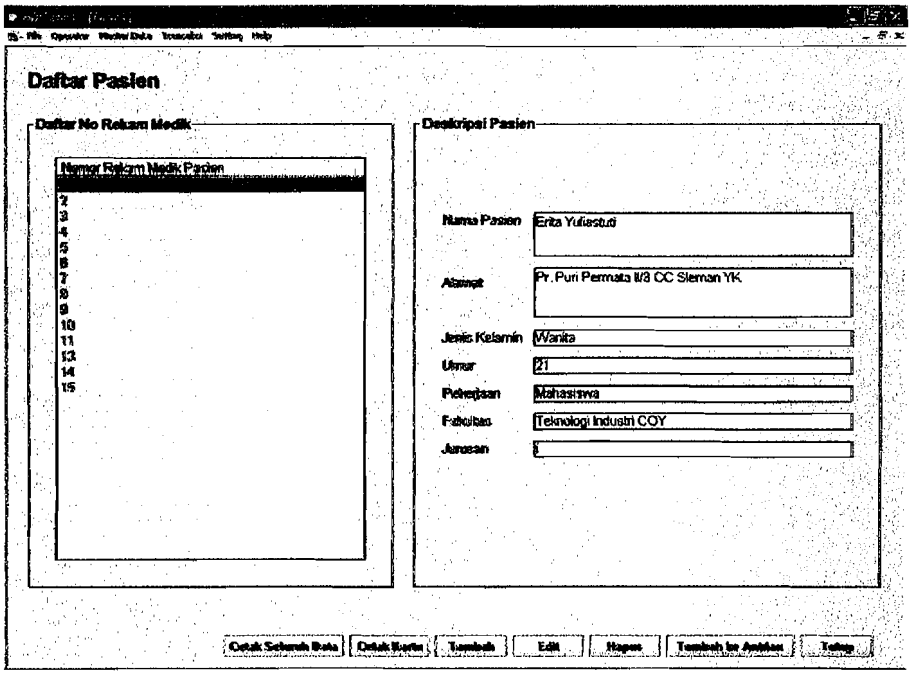
Gambar 3.56 Antarmuka halaman pengguna



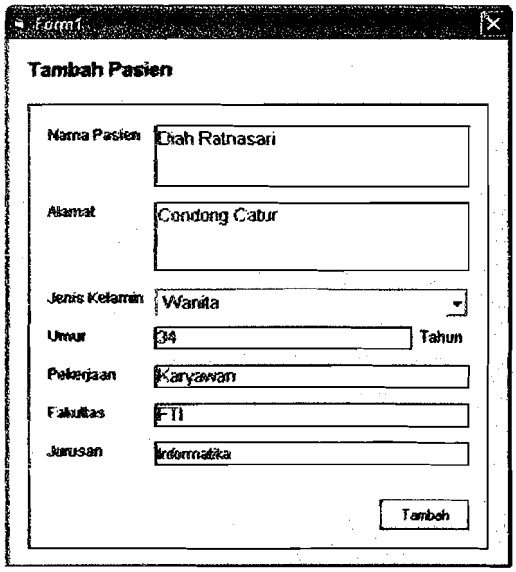
Gambar 3.57 Antarmuka halaman menambah dan mengubah data jenis pengguna

3.4.3.3 Antarmuka kelas Pasien

Merupakan halaman yang dipergunakan untuk menambah, mengubah dan mengatur data-data pasien yang terdaftar di poliklinik. (Gambar 3.58 dan Gambar 3.59)



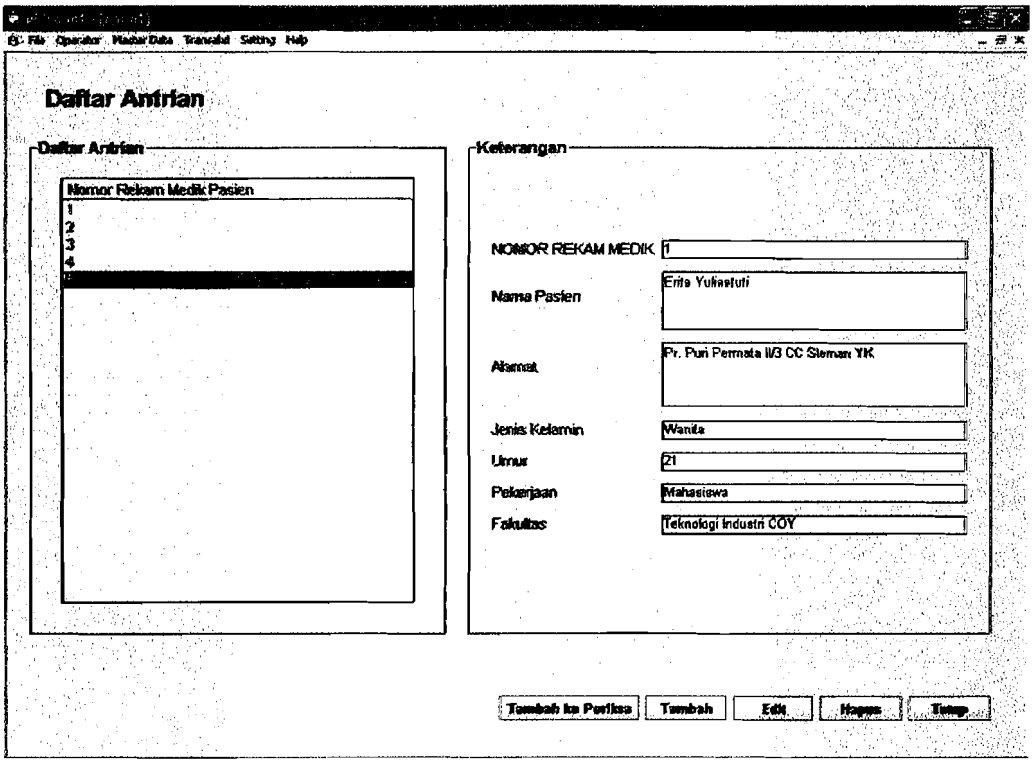
Gambar 3.58 Antarmuka halaman Pasien



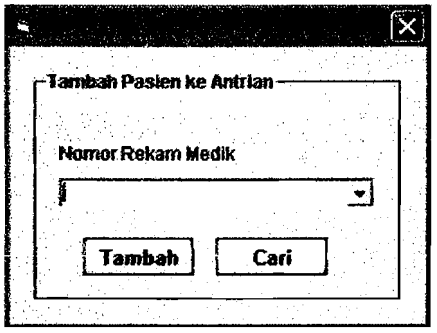
Gambar 3.59 Antarmuka halaman tambah dan ubah data pasien

3.4.3.4 Antarmuka kelas DaftarAntrian

Merupakan halaman yang dipergunakan untuk menambah, mengubah dan mengatur data-data antrian pasien. (Gambar 3.60 dan Gambar 3.61)



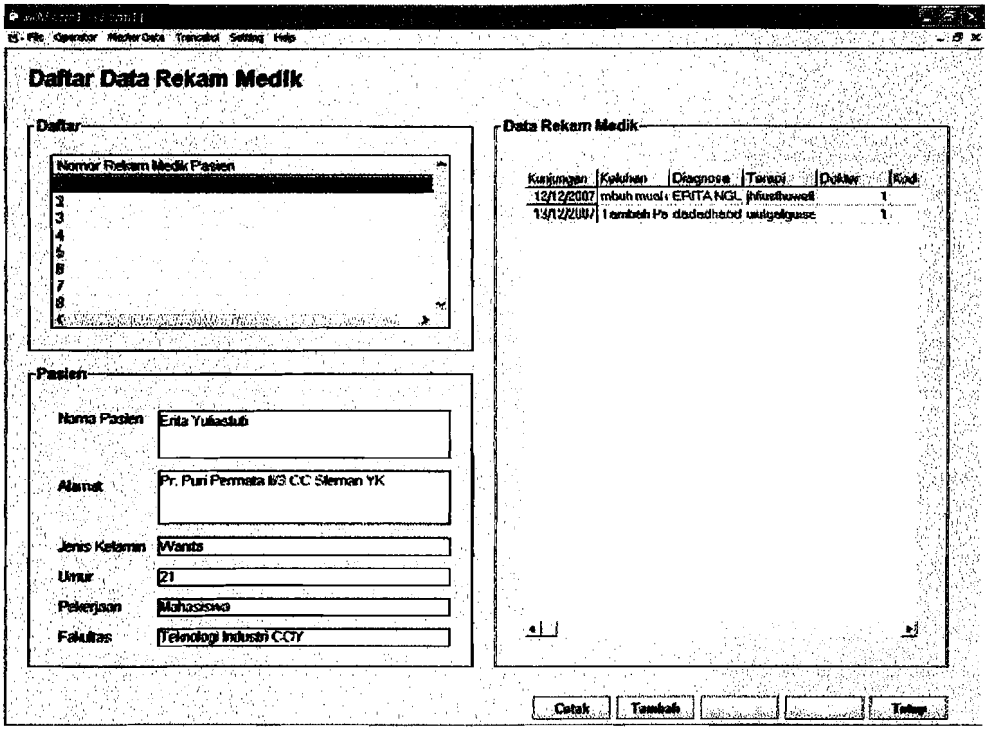
Gambar 3.60 Antarmuka halaman tambah dan ubah data pasien



Gambar 3.61 Antarmuka halaman tambah dan ubah data pasien

3.4.3.5 Antarmuka kelas KartuRekamMedik

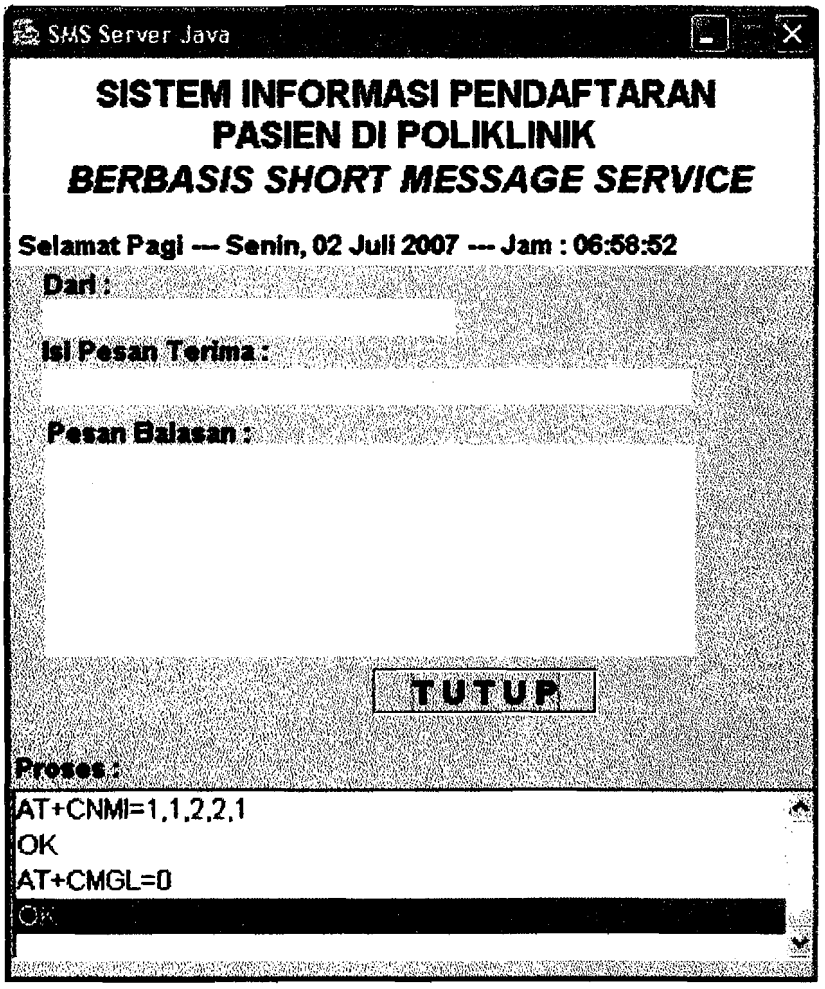
Merupakan halaman yang dipergunakan untuk menambah, mengubah dan mengatur data-data rekam medik pasien yang terdaftar. (Gambar 3.62 dan Gambar 3.63)



Gambar 3.62 Antarmuka halaman rekam medik

3.4.3.7 Antarmuka kelas SMSServer

Merupakan halaman yang dipergunakan untuk mengatur pesan-pesan SMS yang masuk, kemudian memproses sesuai dengan permintaan pesan dan mengirimkannya kembali. (Gambar 3.65)



Gambar 3.65 Antarmuka halaman pasien sementara

BAB IV

HASIL DAN PEMBAHASAN

Dalam bab ini akan dibahas segmen terakhir dalam kerangka pengembangan GRAPPLE (*Guidelines for Rapid APPLication Engineering*) yaitu *Deployment*. Segmen *deployment* memiliki tiga buah aksi antara lain perencanaan *back up* dan *recovery*, instal sistem pada hardware yang sesuai dan uji coba sistem yang terinstal.

4.1 Deployment

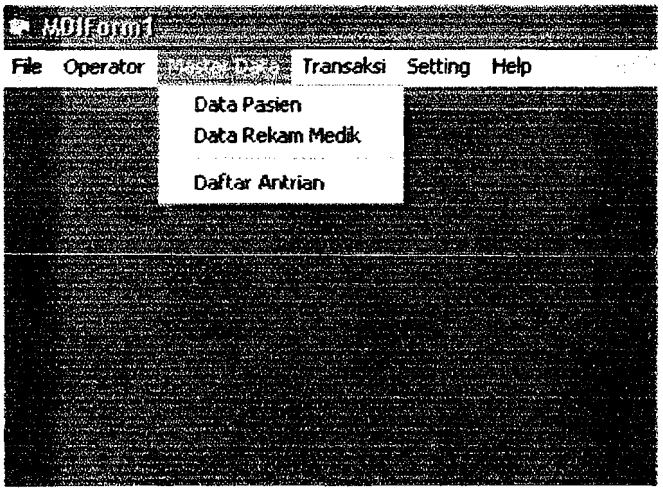
Dalam segmen ini akan dibahas tiga buah aksi yang membahas mengenai hasil dari pembangunan sistem, yaitu: perencanaan *back up* dan *recovery*, instalasi sistem pada hardware yang sesuai dan uji coba sistem yang terinstal.

4.1.1 Perencanaan Back Up dan Recovery

Perencanaan *back up* dan *recovery* diperlukan untuk melakukan penyelamatan data jika sistem mengalami kegagalan. Proses *Back up* data dapat dilakukan dengan menggunakan fasilitas *back up* yang dimiliki oleh SQL SERVER. Cara untuk melakukan pem*back-up*an terhadap data melalui fasilitas yang disediakan oleh Microsoft SQL Server dapat dilihat pada literatur-literatur yang membahas database tersebut secara lebih detil.

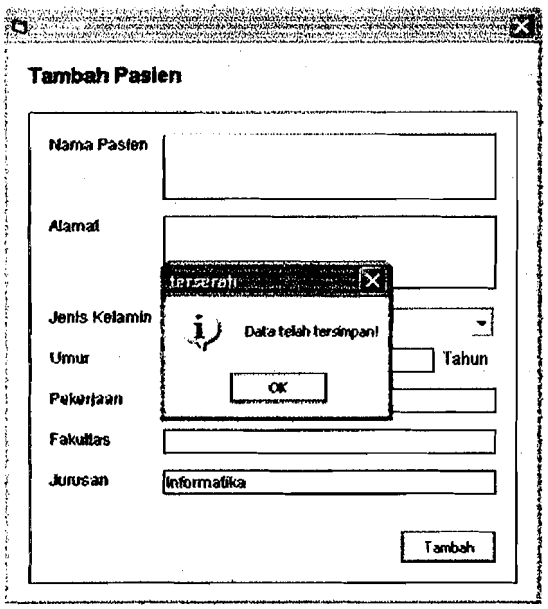
4.1.2 Instalasi Sistem Pada *Hardware* yang Sesuai

Setelah sistem yang dibangun dapat berjalan sesuai dengan rancangan, maka dilakukan instalasi sistem pada *hardware* yang kompatibel. Berikut ini proses instalasi sistem pada *hardware* :



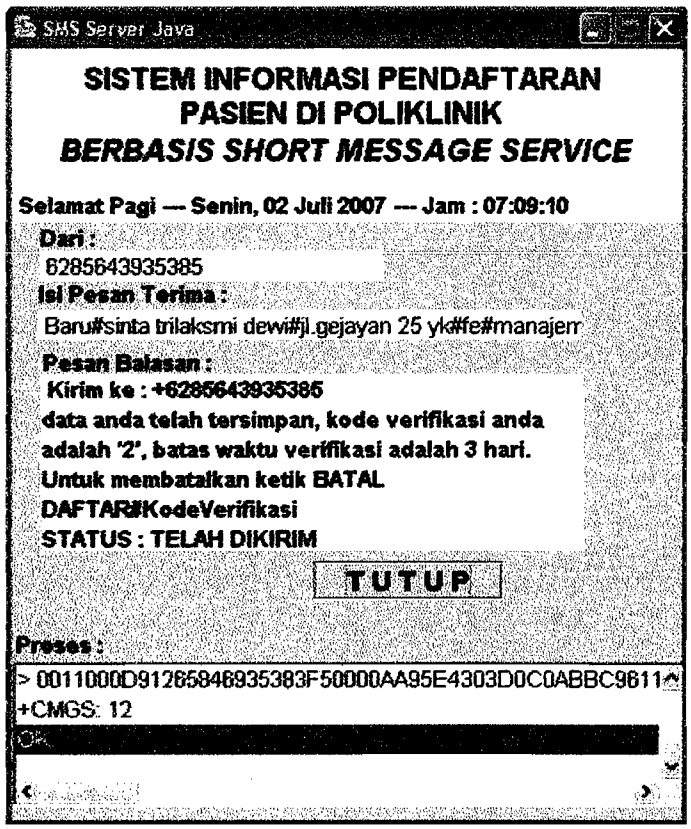
Gambar 4.11 Antarmuka mengakses menu

Pengujian terhadap kelayakan sistem yang terinstal telah dilakukan, jika gambar-gambar diatas berhasil ditampilkan, maka sistem telah berhasil berjalan dengan baik. Pengujian akan dilakukan pada salah satu proses penyimpanan data. Contohnya proses penyimpanan data pasien. (Gambar 4.12)



Gambar 4.12 Antarmuka halaman Hak Akses

Pengujian juga dilakukan terhadap SMS Server, Pengujian akan dilakukan pada seluruh sintaks yang dilayani oelh SMS Server. (Gambar 4.13)



Gambar 4.13 Antarmuka halaman SMS Server keyword daftar

Urutan proses pendaftaran melalui SMS pertama kali dimulai dengan mengirimkan pesan berisi kata DAFTAR. Kemudian sistem akan membalas dengan pesan "Apakah anda sudah punya norm? jika sudah ketik NORM#nomorRekamMedik anda, Jika belum ketik BARU#nama#alamat#jurusan#fakultas. Jika ingin mengecek urutan antrian ketik CEK".

Jika pasien sudah terdaftar dan ingin mendaftarkan diri untuk pemeriksaan kesehatan maka, ketik NORM#5, angka 5 adalah nomor rekam medik pasien. Jika nomor rekam medik ditemukan, maka dilakukan penyimpanan pada daftar antrian. Pesan balasan berisi no urut antrian dan cara untuk membatalkan pendaftaran.

Jika Pasien belum terdaftar dan ingin mendaftarkan diri maka, ketik BARU#erita yuliastuti#condong catur yk#informatika#fti. Data akan tersimpan, pasien akan memperoleh kode verifikasi. Pesan balasan berisi kode verifikasi dan cara untuk membatalkan pendaftaran.

BAB V

SIMPULAN DAN SARAN

7.1 Simpulan

Setelah dilakukan pengamatan terhadap lokasi penelitian, maka dapat disimpulkan bahwa penggunaan aplikasi Sistem Informasi Pendaftaran Pasien Berbasis SMS dapat memberikan manfaat bagi poliklinik tersebut antara lain :

1. Perkembangan teknologi yang semakin pesat dapat mendukung efisiensi, efektifitas dan fleksibilitas pelayanan terhadap masyarakat khususnya di bidang kesehatan.
2. Proses pendaftaran pasien menjadi lebih efektif dan efisien dibandingkan menggunakan sistem konvensional yang dipergunakan sebelumnya. Dengan sistem ini pasien tidak perlu datang ke lokasi untuk mendaftar lalu menunggu giliran pemeriksaan, karena proses pendaftaran dan proses pengecekan pemanggilan dapat dilakukan melalui SMS.
3. Sistem Informasi tersebut dapat memberikan kemudahan dalam menyimpan dan pengaksesan data kembali,
4. Proses pencetakan kartu periksa maupun data rekam medik dapat dilakukan dengan mudah dan rapi dibandingkan dengan catatan-catatan yang ditulis secara manual sebelumnya.

7.2 Saran

Setelah melihat hasil yang dicapai dalam Tugas Akhir ini, maka ada beberapa saran yang perlu disampaikan, antara lain :

1. Sistem Informasi Pasien di Poliklinik Berbasis SMS yang dibuat masih berupa program sederhana, yang masih dapat dikembangkan lagi untuk memperoleh pemrosesan data yang lebih kompleks.

2. Fasilitas pelayanan dalam sistem informasi ini perlu disempurnakan sehingga sistem tidak hanya dapat melayani proses pendaftaran pasien lama, pasien baru, cek antrian, dan pembatalan transaksi saja.
3. Sistem Informasi Pasien di Poliklinik Berbasis SMS dapat dikembangkan lebih luas dari segi bahasa yang diproses. Sebaiknya sistem dapat memproses bahasa-bahasa yang lebih alami sehingga mempermudah penggunaan.

DAFTAR PUSTAKA

[ALH03] Alhir, Sinan, 2003, *Learning UML*, O'Reilly & Associates, Sebastopol.

[AGU03] Agung, 2003, *Membangun SMS Gateway menggunakan PHP*, Elex Media Komputindo, Jakarta.

[DAVB3] Davis, Leitch, 1983, *Management Information System Conceptual Foundations*. McGraw-Hill.

[HAR99] Hartono, Jogiyanto, 1999, *Pengenalan Komputer*. ANDI, Yogyakarta.

[HEL03] Helmy, SKom. *Pengolahan Database SQL Server 2000 dengan Java2*. Elex Media Komputindo, 2003.

[HER04] Hermawan, Benny. *Menguasai Java 2 dan Object Oriented Programming*. Andi, 2004.

[KAD03] Kadir, Abdul, 2003, *Pengenalan Sistem Informasi*. ANDI, Yogyakarta.

[KOM04] Komputer, Wahana, 2004, *Tutorial Membuat Program dengan Visual Basic*. Salemba Infotek, Jakarta.

[KUS02] Kusumo, A. S., *Pemrograman Database dengan Visual Basic 6.0*. Jakarta: Elex Media Komputindo, 2002.

[MAD02] Madcoms, *Seri Panduan Pemrograman Database Visual Basic 6.0 dengan Cristal Reports*. Yogyakarta: Andi, Juli 2002.

[MUN05] Munawar, 2005, *Pemodelan Visual Dengan UML*. Graha Ilmu, Yogyakarta.

[PET02] Petroutsos, Evangelos. *Menguasai pemrograman Database dengan Visual Basic 6*. Terjemahan Adi Kurniadi. Jakarta: Elex Media Komputindo, April 2002.

[SCH99] Schmuller, Joseph, 1999, *Teach Yourself UML in 24 Hours*. Sams Publishing, Indianapolis.

[SIE01] Siebold, Dianne, 2001, *Visual Basic Developer's Guide to SQL Server*. Elex Media Komputindo, Jakarta.

[WAH05] Wahana Komputer, *Pengembangan Aplikasi Sistem Informasi Akademik berbasis SMS dengan JAVA*. Salemba Infotek, 2005.

[ZWA98] Zwas, Vladimir, 1998, *Foundations of Information System*. McGraw-Hill.

LAMPIRAN

LAMPIRAN 1

DAFTAR REKAM MEDIK PASIEN

PT. UNISIA POLIFARMA

Jl. Kaliurang Km. 14,5 Telp. (0274) 898421
Yogyakarta 55584

ID	NORM	TGL BERKUNJUNG	GEJALA	DIAGNOSA	TERAPI	IDDOKTER
1	1	01/01/2007	Pusing, Mual	Keracunan	Dexanta, Vomitrol, Neuralgin	6
2	1	01/01/2007	Panas, Hidung Tersumbat, Keluar Ingus	Flue	Decolgen, Decadril/Sanadril, Paracetamol	6
3	2	12/03/2007	Sakit Perut, Buang air besar berkali-kali	Diare	Diatab, Tetracyl, Oralit	3
4	3	20/02/2007	Sakit Perut, Buang air besar berkali-kali	Diare	Diatab, Tetracyl, Oralit	3
5	7	22/05/2007	Pusing, Mual, Muntah	Keracunan	Dexanta, Vomitrol, Neuralgin	6
6	7	25/06/2007	Batuk, Panas, Hidung Tersumbat, Keluar Ingus	Flue	Decolgen, Decadril/Sanadril, Paracetamol	6
7	5	13/02/2007 6:12:44	Mual, perut sakit/perih	Mag	Dexanta syrup/Tablet	6

LAMPIRAN 2

DAFTAR DATA DIRI PASIEN

PT. UNISIA POLIFARMA

Jl. Kaliurang Km. 14,5 Telp. (0274) 898421
Yogyakarta 55584

NORM	NAMA	ALAMAT	JENISKELAMIN	UMUR	PEKERJAAN	FAKULTAS	JURUSAN
1	Erita Yuliastuti	Perum Puri Permata II/3 Condong Catur	Wanita	22	Mahasiswa	Fakultas Teknologi Industri	Jurusan Teknik Informatika
2	Bastian Yudhatama	Klabanan 27 Sleman Yogyakarta	Pria	23	Mahasiswa	Fakultas Teknologi Industri	Jurusan Teknik Informatika
3	Triyani	Perumahan Candi Indah, Candi Sleman	Wanita	22	Mahasiswa	Fakultas Teknologi Industri	Teknik Informatika
4	Budi Santoso	Klabanan 27 Sleman Yogyakarta	Wanita	22	Mahasiswa	FTI	Informatika
5	Erna Vintania Dwi Atmadja	Ngemplak, ngaglik Sleman Yogyakarta	Wanita	22	Mahasiswa	FTI	Informatika
6	Muhammad Taufik Rakhman	Mekarsari, Jakal km 6 Sleman Yogyakarta	Pria	22	Mahasiswa	FTI	Informatika
7	Ajeng Anggraini	Ngemplak, Wedomartani Sleman	Wanita	22	Mahasiswa	Ekonomi	Manajemen
8	Setyo Hermawan	Mekarsari, Jakal km 6 Sleman Yogyakarta	Pria	22	Mahasiswa	FTI	Informatika

LAMPIRAN 3

PT. UNISIA POLIFARMA

Boulevard Kampus Terpadu UII
Jl. Kaliurang Km. 14,5 Telp. (0274) 898421 Yogyakarta 55584

KARTU BEROBAT

NAMA: Erita Yuliastuti
ALAMAT: Perum Puri Permata II/3 Condong Catur Sleman
UMUR: 22
PEKERJAAN: Mahasiswa
JENISKELAMIN: Wanita
FAKULTAS: Fakultas Teknologi Industri
JURUSAN: Jurusan Teknik Informatika
NORM: 1

BILA BEROBAT KARTU INI HARAP DIBAWA