Sistem Monitoring Server Berbasis SMS (Studi Kasus : Server Siakad Unila)

Mardiana[1], Wahyu Eko Sulistiono[1], Johan Iwan Santoso[2]
1. Dosen Jurusan Teknik Elektro Universitas Lampung
2. Alumni Jurusan Teknik Elektro Universitas Lampung
Jl. S. Brojonegoro No. 1 Gedung Meneng Bandar Lampung 35145
mardiana@unila.ac.id, eko@unila.ac.id, johan_is@unila.ac.id

Abstrak–Siakad Universitas Lampung telah aktif dipergunakan sejak tahun 2000, dengan jumlah *user* yang terus bertambah seiring dengan bertambahnya mahasiswa Universitas Lampung. Dengan jumlah *user* yang demikian banyak, maka penanganan *user* juga memerlukan jumlah server dan administrator yang memadai untuk mengakomodir jumlah *user*. Keterbatasan administrator sering memicu terlambatnya penanganan pada kinerja server yang buruk. Dengan memperhatikan keterbatasan administrator agar dapat dilakukannya pengawasan terhadap kinerja server tersebut, maka layak untuk membuat sebuah *tool* yang dapat melakukan pengawasan kinerja server untuk dilaporkan kepada administrator pada waktu administrator tidak berada di tempat. Sehingga ketika kinerja server dianggap buruk, administrator dapat mengambil tindakan secara cepat dan tepat.
Berdasarkan kondisi tersebut dan dengan mempertimbangkan sarana SMS(*Short Message Service*) sebagai layanan yang telah banyak digunakan, maka dalam penelitian ini dibuat sebuah aplikasi *Monitoring* Server Siakad Unila dengan menggunakan SMS sebagai media *error reporting*-nya. Pengembangan aplikasi ini dilakukan dengan pendekatan berorientasi objek, menggunakan Java sebagai bahasa pemrograman berorientasi objek dan UML untuk pemodelannya. Aplikasi ini akan menganalisa server untuk menilai kinerja server dan mengirimkan *error report* pada saat ditemukan indikasi kinerja server yang buruk yang meliputi indikasi kegagalan *web server* dengan membaca *status code* pada *access log*, akses ilegal pada *database server*, kondisi server yang *overloaded*, dan koneksi yang terputus pada *web server* maupun *database server*.
Hasil pengujian dari aplikasi yang dibuat ini menunjukkan bahwa aplikasi ini dapat menganalisa setiap aspek yang dibutuhkan oleh server Siakad Unila, namun dengan *delay* yang cukup besar.

Kata kunci *:* Siakad Unila, server, Aplikasi *Monitoring* Server Siakad Unila, SMS, *error report*.

Abstract--*University of Lampung's Siakad has been used since 2000, with increasing number of users aligning to increasing number of students. The management needs more servers and administrators to accommodate this vast the number of users. The lack of administrators often causes late handling of bad server performances. By concerning the limitation of server performance monitoring, it is needed to create a tool that can monitor server performance in order to report to the remote administrators. Therefore when server performs not well, administrator can overcome this quickly and accurately.*
By considering that condition and the availability SMS(Short Message Service) as a widely-used service, in this research a monitoring application for Unila's Siakad server is developed using SMS as the tool for reporting. Application is developed using UML as modeling tool and using Java as programming language. This application analyses server for performance indicators including failure condition by reading status code at access log, illegal access on database server, overloaded server condition, and connection status on web server and database server, then it sends error report.
From testing result, this application can analyze every aspect needed by Siakad Unila, but the result shows high delay of reporting.

Keywords : *Siakad Unila, monitoring application, SMS.*

A. Pendahuluan

Universitas Lampung sebagai sebagai salah satu institusi perguruan tinggi memiliki jaringan komputer beserta *server*-nya dalam lingkup intranet dan internet yang dapat dipergunakan untuk mengakses sistem informasi yang ada, misalnya

Naskah ini diterima pada tanggal 15 Juni 2008, direvisi pada tanggal 20 Juli 2008 dan disetujui untuk diterbitkan pada tanggal 1 Agustus 2008

Sistem Informasi Akademik (Siakad). Sistem informasi ini dipergunakan oleh para dosen maupun mahasiswa untuk memperoleh informasi secara cepat dan akurat tentang hal-hal yang bersifat akademis.

Siakad Universitas Lampung sendiri telah aktif dipergunakan sejak tahun 2000, dengan jumlah *user* terus bertambah seiring dengan bertambahnya mahasiswa Universitas Lampung. Dengan jumlah *user* yang demikian banyak, maka penanganan *user* juga memerlukan jumlah *server* dan administrator yang memadai untuk mengakomodir jumlah *user*. Keterbatasan administrator sering memicu terlambatnya penanganan pada kinerja *server* yang buruk.

Kinerja dari sebuah *server* sebenarnya dapat diawasi dengan memperhatikan *server log file* maupun *database log file* pada *database server*. Pada sebuah *web server* informasi tentang kegiatan *user* dan informasi penting lainnya terekam dalam *access log*. Begitu juga pada *database server* yang menyimpan aktivitas *user* dalam sebuah *log file*. Dari log file yang ada kita dapat memperhatikan kondisi web server dan adanya usaha-usaha ilegal untuk mengakses sebuah database server.

Dengan memperhatikan keterbatasan administrator dan dapat dilakukannya pengawasan terhadap kinerja *server* tersebut, maka layak untuk membuat sebuah *tool* yang dapat melakukan pengawasan kinerja *server* untuk dilaporkan pada administrator sewaktu-waktu administrator tidak berada di tempat. Sehingga ketika kinerja *server* dianggap buruk, administrator dapat mengambil tindakan secara cepat dan tepat. Penganalisa *server* yang telah ada membuat *report* kepada administrator melalui *e-mail* yang tidak bersifat *real time*. Sehingga alternatif yang lebih baik adalah dengan mengirimkan *report* kinerja *server* kepada administrator melalui SMS (*Short Message Service*).

Dengan pembuatan aplikasi yang dapat dijadikan *tools* penganalisa *server* sekaligus pemberi *report* pada administrator, tindakan yang diperlukan dalam menghadapi situasi kinerja *server* yang buruk dapat diambil secepatnya. Hal ini akan berdampak pada keandalan dari *server* yang dimaksud.

B. Tinjauan Pustaka

Web Server

Sebuah *web server* dapat didefinisikan sebagai sebuah komputer yang menerima *HTTP request* dari *clients*, umumnya melalui *web browsers* dan menampilkan halaman web, misalnya dokumen HTML dan objek lain (gambar, *link*) [1]. Walaupun berbagai *web server* yang ada memiliki detail yang berbeda-beda, namun mereka memiliki persamaan dalam fitur dasar, yaitu :

a) Merespons *HTTP request* ; semua *web server* bekerja dengan menerima *HTTP request* dari jaringan dan memberikan respons terhadap *request* tersebut. Respons yang dihasilkan dapat berupa dokumen HTML, gambar, teks, dan format lainnya bahkan respons dapat berupa *error message* untuk kondisi kegagalan tertentu.

b) *Logging* ; biasanya *web server* memiliki kemampuan untuk melakukan *logging* terhadap beberapa detail informasi, tentang *HTTP request* dan respons yang diberikan ke dalam *log file*. Administrator dapat melakukan analisa terhadap *web server* dengan cara menganalisa *log file* tersebut.

Secara umum gambaran mengenai aktivitas *request-response* pada *web server* dapat dilihat pada gambar di bawah ini :

Gambar 1. Proses *request-response* pada *server*

Pada umumnya sebuah *web server* memiliki batasan pelayanan terhadap jumlah koneksi tertentu dalam satu waktu (*concurrent*). Batasan itu membuat *server* hanya mampu melayani jumlah tertentu dari *request* tiap detiknya. Pada saat *request* pada *server* hampir mencapai batasan tersebut, *server* menjadi *overloaded* dan akan mengakibatkan *server* menjadi *unresponsive*. Gejala-gejala yang ditunjukkan oleh *web server* yang *overloaded* adalah :

a. *Delay* yang panjang untuk merespons suatu *request* (antara satu hingga beberapa ratus detik).
b. 500 dan 503 *HTTP errors* sebagai respons yang mengindikasikan kegagalan *server*.
c. Koneksi TCP yang terputus.

Database Server

Selain *web server* dikenal juga *database server*, yang didefinisikan sebagai sebuah program komputer yang menyediakan layanan *database* untuk program komputer atau komputer lainnya dalam hirarki *client-server* [2]. Kondisi ini juga mengacu pada komputer yang menjalankan fungsi *database server* tersebut. *Database Management Systems* (DBMS) menyediakan fungsi *database server* dan beberapa DBMS (misal: MySQL) menggunakan model *client-server* untuk akses *database*-nya.

Beberapa *database* yang banyak digunakan adalah MySQL, Oracle, dan MS Access dimana hanya MySQL yang dapat digunakan tanpa membeli lisensi (*freeware*). Sedangkan *database* yang akan digunakan untuk membangun aplikasi ini adalah Oracle, dimana Oracle didesain untuk menjadi *database* yang *portable*. Oracle tersedia untuk setiap platform dari Windows hingga UNIX dengan arsitektur yang disesuaikan dengan sistem operasi yang digunakan.

Pada *database* Oracle, terdapat dua pengertian yang sangat penting, yaitu *database* dan *instance*. *Database* didefinisikan sebagai sebuah kumpulan dari *file* fisik sistem operasi. Sedangkan *instance* memiliki definisi kumpulan dari *background process* pada Oracle dan sebuah *shared memory area*, dimana *memory* tersebut digunakan dalam setiap *process* yang dijalankan pada sebuah komputer tunggal [3]. Relasi antara keduanya adalah *database* dapat disusun dan dibuka oleh banyak *instance*.

Untuk melakukan koneksi ke Oracle dapat dilakukan dengan dua cara yang paling umum yaitu *dedicated server* dan *shared server*. Koneksi *dedicated server* diindikasikan dengan adanya sebuah proses yang didedikasikan kepada *client* sepanjang *session*. Proses ini bukan bagian dari *instance*, sehingga *client* akan terhubung langsung dengan *dedicated server* melalui media jaringan, misalnya soket TCP/IP. Proses ini juga yang akan menerima SQL dan mengeksekusinya, bila diperlukan dengan membaca data pada *database*. Proses tambahan seperti ini tidak akan ditemukan pada koneksi *shared server* yang juga dikenal sebagai *Multi-Threaded erver* (MTS). Pada koneksi *shared server*, Oracle menggunakan sekumpulan *shared process* untuk melayani komunitas pengguna sehingga tidak menambahkan proses pada *server* untuk setiap *session* yang dibuat.

Koneksi pada jaringan untuk mengakses *database* Oracle dapat dilakukan dengan

melakukan *request* melalui TCP/IP. Dalam hal ini *client* berada pada satu mesin dan *server* pada satu mesin lainnya, dimana keduanya terhubung dalam sebuah jaringan TCP/IP. *Client* akan melakukan *request* menggunakan *client software* pada Oracle (sejumlah *application program interfaces,* API) setelah melakukan koneksi ke *database*.

Dalam sebuah jaringan, akan dijalankan sebuah proses yang bernama TNS (*Transparent Network Substrate*) *listener* pada *server*. Proses *listener* ini menunjukkan hal-hal pada saat terjadi koneksi ke *database*. Ketika *connection request* diterima, proses ini memeriksanya lalu dengan konfigurasi yang dimilikinya dapat menolak *request* tersebut (karena alasan database tidak tersedia atau IP *address* yang tidak diperkenankan untuk melakukan koneksi) atau menerimanya sehingga terjadi sebuah koneksi.

Listener merupakan komponen Oracle yang menunjukkan koneksi ke *database* termasuk *remote connections*. Seperti halnya pada *web server*, *database server* juga membuat *log file* yang diperlukan untuk menganalisa kinerjanya. Pada Oracle, terdapat *listener log* yang menyimpan informasi-informasi tentang koneksi yang terjadi pada *database.*

Aplikasi SMS

SMS (*Short Message Service*) adalah sebuah layanan yang terdapat pada hampir semua ponsel (dan perangkat *mobile* lainnya seperti *Pocket PC* maupun komputer *desktop*) yang memungkinkan pengiriman pesan pendek (dikenal juga dengan sebutan *text message* atau SMS) antar ponsel, perangkat *portable* lainnya, maupun telepon [4]. Penggunaannya tidak hanya sebatas mengirim pesan, memesan *ringtone* maupun *wallpaper*, namun banyak dikembangkan kegunaan lain dari SMS. Perkembangan teknologi informasi memungkinkan banyak *user* yang memanfaatkan layanan SMS untuk pertukaran informasi, mengingat SMS merupakan media yang murah namun cepat dan akurat. SMS memiliki jaringan yang lebih luas dibandingkan dengan pemanfaatan *e-mail* sebagai media informasi.

SMS merupakan salah satu fitur *messaging* yang ditetapkan oleh standar ETSI (www.etsi.org), pada dokumentasi GSM 03.40 dan GSM 03.38. Pada saat kita mengirim SMS dari ponsel, pesan tersebut tidak langsung dikirimkan ke ponsel tujuan. Pesan tersebut akan dikirimkan terlebih dahulu ke SMS Center (SMSC), kemudian pesan tersebut diteruskan ke ponsel tujuan. Secara singkat, proses yang terjadi dapat digambarkan dalam skema di bawah ini :

Gambar 2. Skema cara kerja SMS

Pesan dikirim dengan mekanisme *store and forward* melalui SMSC, yang akan mencoba untuk mengirimkan pesan ke penerima. Bila penerima tidak terjangkau pada waktu pengiriman, SMSC akan menyimpan pesan tersebut. Pada saat penerima kembali terjangkau, SMSC akan mencoba kembali proses pengiriman tersebut.

Seiring dengan kemajuan teknologi informasi, beberapa layanan SMS dapat dikembangkan. Beberapa layanan SMS yang potensial untuk dikembangkan adalah :

a) *Notification Services* , merupakan layanan yang memberikan pemberitahuan atas sebuah kejadian tertentu. Contoh penggunaan layanan ini adalah sebagai indikator atas masuknya *e-mail*, *fax*, atau *voice*

message. Layanan ini juga dapat digunakan untuk kejadian-kejadian yang terjadwal, dalam hal ini berfungsi sebagai *reminder* maupun eksekutor *action* tertentu.

b) *E-mail Interworking* , merupakan pengintegrasian layanan *e-mail* dengan SMS yang memungkinkan akses *e-mail* melalui SMS.

c) *Information Services* , merupakan jenis layanan yang memberikan begitu banyak informasi kepada pengguna melalui SMS. Informasi ini antara lain informasi trafik, hiburan, keuangan, maupun petunjuk tertentu. Layanan ini menggunakan metode *delivery on demand*, atau respons terhadap *request*.

d) *WAP Integration* , layanan SMS yang terintegrasi dengan WAP yang memungkinkan SMS menjadi mekanisme transport pada pesan WAP. Pesan ini dapat memuat berbagi informasi dari berbagai sumber, seperti *database*, *World Wide Web*, dan *e-mail server*.

Aplikasi yang akan dibuat akan mengadopsi jenis layanan yang pertama, yaitu *Notification Services*, dimana *user* akan menerima SMS pemberitahuan apabila sebuah kejadian tertentu terjadi (dalam hal ini terpenuhinya suatu indikasi tertentu).

Untuk dapat melakukan pengiriman dan penerimaan SMS dari PC, diperlukan koneksi antara PC dengan GSM *modem* atau *handphone*. Pemilihan antara menggunakan GSM *modem* atau *handphone* dapat disesuaikan dengan kebutuhan masing-masing dengan pertimbangan perbedaan harga dan kualitas sinyal. Kualitas sinyal yang dimiliki oleh GSM *modem* lebih baik dibandingkan *handphone* karena GSM *modem* dedesain khusus untuk mengirim dan menerima SMS. Namun dari sisi harga, penggunaan *handphone* dirasa lebih menarik. Beberapa GSM *modem* yang diproduksi oleh Siemens antara lain TC45, TC35, dan M20. Beberapa GSM *modem* terbaru bahkan telah mendukung MMS (*Multimedia Message Service*) dan GPRS.

Untuk dapat melakukan koneksi ke PC, dapat dilakukan dengan melakukan koneksi terminal pada PC. Terdapat berbagai macam program yang dapat digunakan untuk melakukan koneksi ke terminal, diantaranya adalah Hyper Terminal dari Microsoft Windows. Untuk dapat menggunakan program ini, kita terlebih dahulu menghubungkan terminal pada *serial port* yang terdapat pada PC (misal COM1 atau COM2). Pada tahap awal dapat dilakukan uji koneksi dengan mengetikkan perintah "AT" pada Hyper Terminal, dan bila koneksi berhasil maka akan muncul respons "OK". Sedangkan untuk mengetahui nama *vendor* terminal yang digunakan dapat dilakukan dengan perintah "AT+CMGI". Perintah AT (AT *Command*) lain dapat digunakan untuk melakukan pengiriman maupun penerimaan pesan dan fungsi lainnya. Beberapa perintah yang sering digunakan antara lain terdapat pada tabel di bawah ini :

Tabel Daftar Perintah AT

Perintah AT	Fungsi
AT+CMGS	Mengirim Pesan
AT+CMGR	Membaca pesan
AT+CMGD	Menghapus pesan

Dalam penggunaannya, perintah AT tersebut dapat dilakukan di luar Hyper Terminal dengan menggunakan Java dan paket komunikasinya, Java Communication Port.

C. Metode Penelitian

Tahapan yang dilakukan dalam penelitian ini adalah sebagai berikut :

a. Studi Literatur
   Studi literatur dimaksudkan untuk mempelajari berbagai sumber referensi

atau teori (buku dan internet) yang terkait dengan penelitian.

b. Pengembangan Perangkat Lunak berorientasi objek.
   1. Analisis, dimulai dengan menganalisis spesifikasi kebutuhan sistem, definisi *use case* yang menggambarkan bagaimana sistem berjalan, kemudian membuat *CRC Modelling* untuk menganalisis *class diagram* yang diperlukan.
   2. Desain, mendetilkan class diagram yang diperoleh pada tahapan analisis, mendesain prilaku sistem menggunakan sequence diagram, deployment diagram dan database
   3. Implementasi, melakukan pengkodean berdasarkan desain yang telah dibuat.
   4. Pengujian, melakukan *unit testing*, *integration testing dan system testing,* terhadap sistem yang telah dibangun.

D. Hasil Pembahasan

Aplikasi *Monitoring Server* Siakad Unila

Tahapan pembuatan aplikasi dilakukan dengan pendekatan berorientasi objek, dan dijabarkan sebagai berikut :

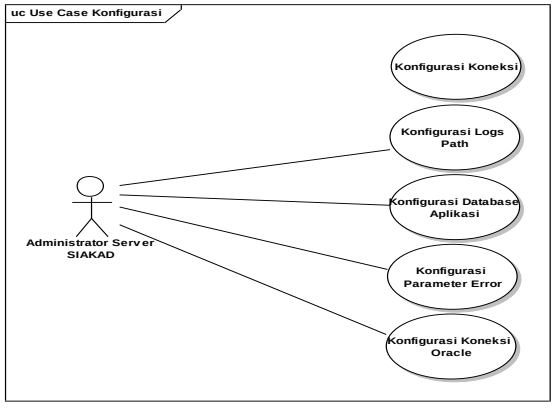
1. Analisis

a. Spesifikasi Kebutuhan

Pada tahap ini kita harus memahami konteks dari permasalahan yang ada, meliputi pemahaman kebutuhan sistem (*system requirements*) dan pemodelan sistem. *System requirements* dapat dikembangkan dari permasalahan yang menyebabkan perlunya dibuat aplikasi ini. Sehingga dapat dirangkum kebutuhan-kebutuhan sebagai berikut :

1. Aplikasi yang dibuat harus dapat mengambil informasi-informasi penting yang terdapat pada *log file*, baik *access log* pada *web server* (Apache) maupun *listener log* pada *database server* (Oracle).
2. Aplikasi dapat menganalisa informasi-informasi tersebut dengan menggunakan acuan yang telah dibuat sehingga dapat segera diketahui kondisi *web server* yang kritis dan usaha akses illegal ke *database server*.
3. Aplikasi dapat mengirimkan SMS yang berisi pesan kesalahan yang terjadi kepada administrator *server* apabila kondisi-kondisi tersebut terpenuhi.
4. Aplikasi ini dijalankan hanya pada saat administrator *server* sedang tidak berada di tempat sehingga aplikasi ini diharapkan dapat menggantikan sementara fungsi pengawasan yang dilakukan oleh administrator.

Dengan mengacu pada analisa kebutuhan diatas, maka dapat dibuat diagram *Use-Case* sebagai berikut :

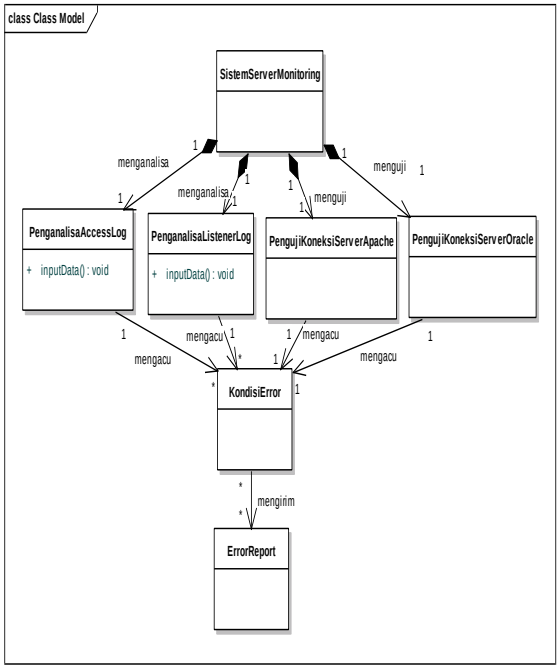


Gambar 3. *Use Case* Diagram

Dari diagram *use-case* di atas, dapat terlihat bahwa secara umum sistem ini dapat digunakan oleh Administrator *server* Siakad untuk melakukan *monitoring* terhadap kondisi *server* Siakad. Sistem akan melakukan *monitoring* terhadap beberapa parameter yang telah dikonfigurasi terlebih dahulu oleh Administrator sebelum menjalankan aplikasi *monitoring*. Saat sistem menemukan indikasi kesalahan, sistem akan mengirimkan sebuah *report* kepada Administrator (berbentuk SMS).

b. *CRC Modelling*

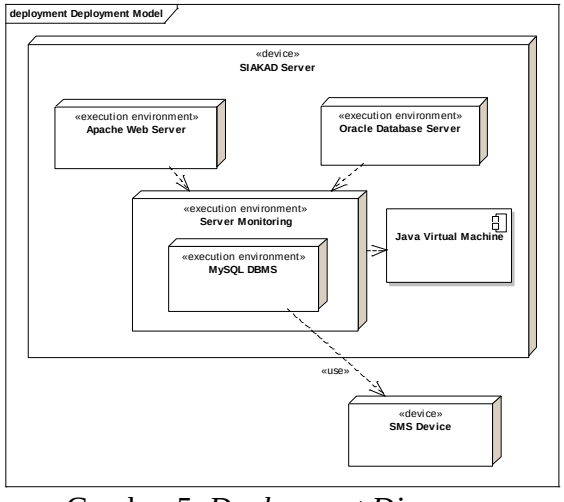
Pada tahap ini dihasilkan *class diagram* yang menggambarkan *class-class* yang perlu disediakan untuk melakukan fungsi yang diharapkan pada sistem.
Dengan menggunakan metode *verb/noun* dan *CRC Modelling* dapat dibuat *class diagram* sebagai berikut :



Gambar 4. *Class Diagram* tahap Analisis

2. Desain

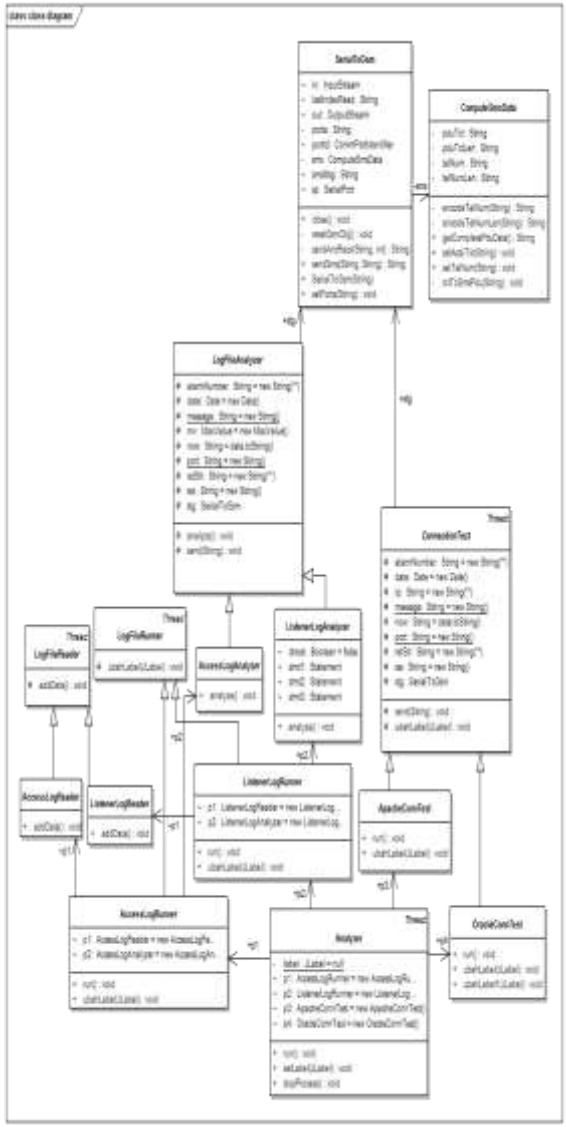
Pada tahap desain ini dihasilkan *class diagram, sequence diagram, deployment diagram* dan skema *database* yang akan digunakan.



Gambar 5. *Deployment Diagram*

Dengan mempertimbangkan hanya dua *device* yang diperlukan, yaitu Siakad *server* yang telah ada ditambah dengan sebuah SMS *device* (dalam hal ini sebuah ponsel) maka *deployment diagram* yang dihasilkan akan terlihat seperti gambar diatas. Aplikasi akan langsung ditempatkan pada Siakad *Server* sebagai sebuah aplikasi yang berdiri sendiri dengan menggunakan *Java Virtual Machine* (JVM) dan mengambil data dari *web server* Apache dan *database server* Oracle.

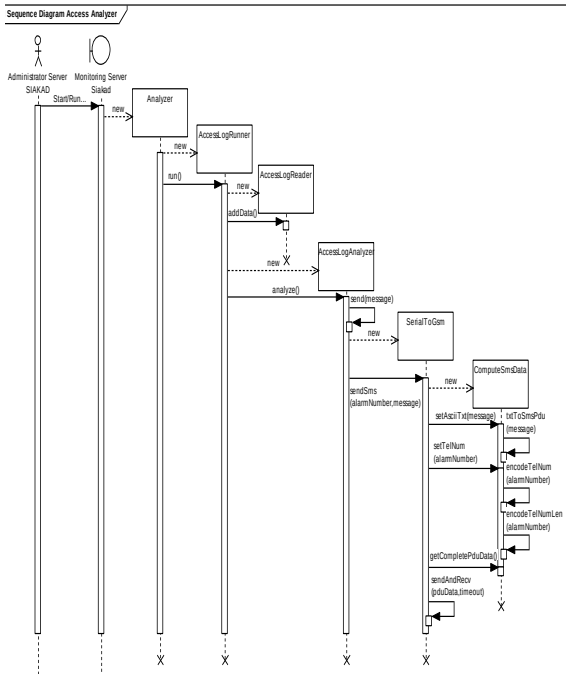
Tahap selanjutnya adalah melakukan desain *class* berdasarkan kandidat-kandidat *class* yang telah dibuat pada proses analisis.



Gambar 6. *Class Diagram* Tahap Desain

Class KondisiError dieliminasi karena tiap proses *monitoring* memiliki kondisi yang berbeda yang dinyatakan sebagai *error* dan akan lebih baik jika dibuat sebagai kondisi pada *method* analisa maupun tes koneksi. Untuk mengirimkan pesan kesalahan berupa SMS digunakan dua buah *class* yang telah dibuat oleh developer lain (Marco Tozzini) yaitu *class* SerialToGsm dan ComputeSmsData.

Setelah *class diagram*, selanjutnya dibuat *sequence diagram* yang menunjukkan interaksi antar *object*. Pada *sequence diagram* dibawah ini ditunjukkan interaksi antar *class* yang telah dibuat dalam setiap proses yang terjadi. Proses menganalisa access.log, listener.log, uji koneksi ke web server, dan uji koneksi ke database server dimodelkan melalui interaksi yang terjadi antara *class-class* yang terlibat dalam proses melalui masing-masing *method*.

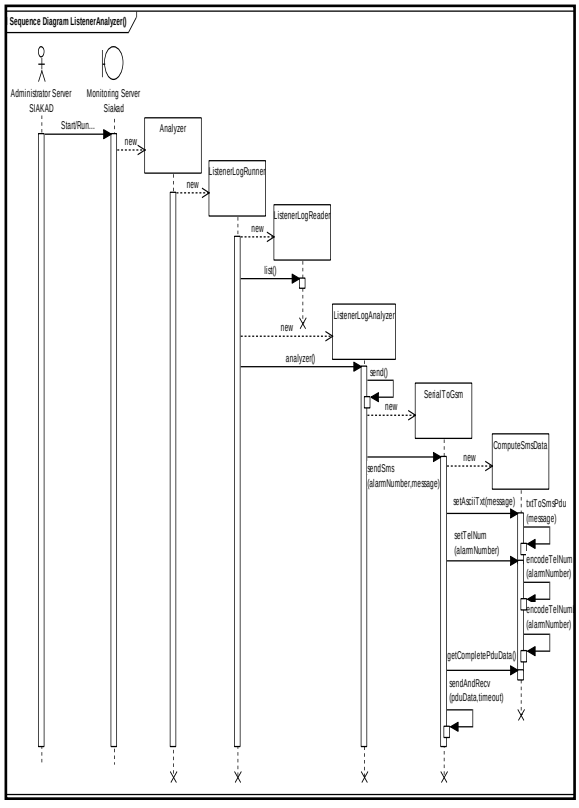


Gambar 7. *Sequence diagram* access.log

Pada *sequence diagram* di atas, dapat dijelaskan interaksi yang terjadi pada proses analisa access.log. *Class* pertama yang terlibat adalah Analyzer yang akan menjalankan AccessLogRunner yang memanggil *method* addData() pada AccessLogReader dan *method* analyze() pada AccessLogAnalyzer.

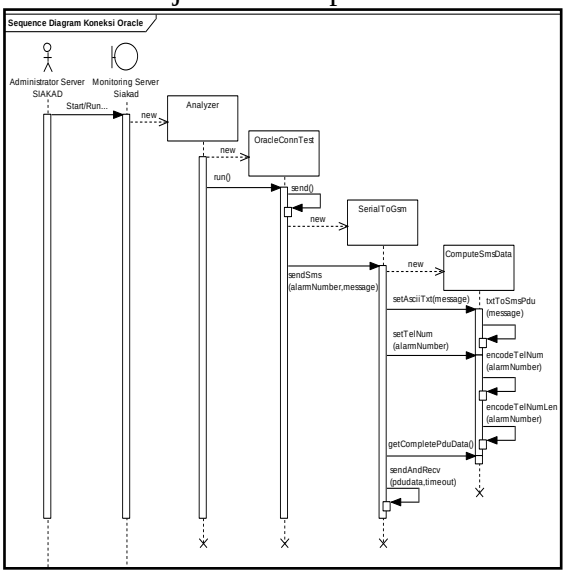
Saat ditemukan indikasi kesalahan pada *method* analyze(), AccessLogAnalyzer akan mengeksekusi *method* send(message) akan memanggil *method* sendSMS(alarmNumber,message) pada *class* SerialToGsm dengan mengirimkan *string* pesan yang akan dikirimkan dan nomor ponsel tujuan.

Class ini melalui *method* sendSms-nya tersebut akan mengeksekusi *method* setAsciiTxt(message) dan setTelNum(alarmNumber) pada *class* ComputeSmsData untuk mendapatkan hasil *encoding* isi pesan dan nomor ponsel dalam format PDU. Setelah mendapatkan format PDU, SerialToGsm melalui *method* sendAndRecv(pduData,timeout) memberikan *AT Command* yang akan mengirimkan SMS ke ponsel tujuan.



Gambar 8. *Sequence diagram*

uji koneksi Apache

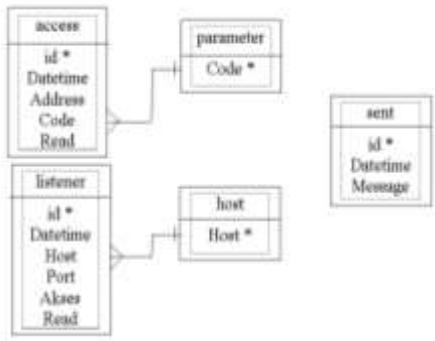


Gambar 9. *Sequence diagram* uji koneksi Oracle

Perbedaan *sequence diagram* uji koneksi Oracle dengan uji koneksi Apache hanya pada *class* yang dipanggil oleh *class* Analyzer melalui *method* run(). Pada uji koneksi Oracle, *class* Analyzer akan memanggil *method* run pada *class* OracleConnTest. Pada saat OracleConnTest menemukan indikasi kesalahan, akan dikirim SMS dengan cara yang sama dengan uji koneksi Apache.

Dengan dihasilkannya *class diagram* yang menggambarkan *class-class* yang akan dibuat dan menggambarkan interaksi yang terjadi melalui *sequence diagram*, langkah berikutnya dalam tahap desain adalah membuat desain database aplikasi. *Database* yang digunakan dalam aplikasi ini adalah *database* MySQL dengan lima buah tabel yang dibuat. Tabel- tabel tersebut adalah :

a) access : id (bigint(5), auto_increment, PRI) , Datetime (datetime) , Address (varchar(50)) , Code (varchar(3)) , Read (varchar(1)).
b) listener : id (bigint(5), auto_increment, PRI) , Datetime (datetime) , Host (varchar(15)) , Port (varchar(10)) , Akses (varchar(8)) , Read (varchar(1)).
c) sent : id (bigint(3), auto_increment, PRI) , Datetime (varchar(34)) , Message (varchar(160)).
d) host : Host (varchar(15), PRI).
e) parameter:Code (varchar(3), PRI).



Gambar 10. Relasi Antar Tabel

Relasi antar tabel diatas menunjukkan adanya relasi antara tabel access dengan parameter, dan antara tabel listener dengan host, sedangkan tabel sent tidak memiliki relasi dengan tabel manapun. *Field* Code pada tabel access memiliki relasi langsung dengan *field* Code pada tabel parameter. Sedangkan antara tabel listener dengan tabel host terdapat relasi melalui *field* Host.

3. Implementasi

Tahap implementasi merupakan tahap penerjemahan desain ke dalam baris-baris kode yang akan difungsikan membentuk subsistem-subsistem yang akan berkolaborasi dalam keseluruhan sistem yang akan dibuat. Pada tahap ini, *class-class* yang telah dibuat pada tahap analisis akan direalisasikan ke dalam fungsi-fungsi di dalam baris kode.

Berikut penjelasan user interface yang dihasilkan :

Secara umum aplikasi ini menjalankan fungsi pengawasan *server* Siakad Unila dengan cara menjalankan beberapa *monitoring* yang telah ditentukan sejak

awal pembuatan aplikasi, yaitu pada tahap analisa kebutuhan.

Halaman Utama

Halaman utama pada aplikasi ini merupakan sebuah *window* yang terdiri dari tiga buah menu utama, yaitu *File, Configuration,* dan *Help*. Pada halaman utama terdapat indikator proses yang sedang berjalan di pojok kiri bawah *window*. Untuk masuk ke halaman utama dapat dilakukan dengan dua cara, yaitu dengan mengeksekusi *executable file* yang dihasilkan oleh *installer* (ServMon.exe) atau dengan melakukan pengeksekusian *class* utama aplikasi ini (com.sms.StartHere) untuk mendapatkan detil proses pada konsol.

Menu File

Menu yang pertama pada aplikasi ini adalah menu *File* yang terdiri atas submenu *Start/Run..., Stop and Exit,* dan *Exit*. Untuk menjalankan *monitoring* menggunakan aplikasi ini dapat dilakukan dengan menggunakan submenu *Start/Run...* sedangkan untuk menghentikan proses *monitoring* dapat menggunakan submenu *Stop and Exit*. Submenu *Exit* digunakan untuk keluar dari aplikasi ini.

Menu Configuration

Menu kedua adalah menu yang digunakan untuk melakukan konfigurasi terhadap aplikasi ini, yaitu menu *Configuration*. Ada beberapa hal yang perlu dikonfigurasi dalam aplikasi ini, yaitu : parameter koneksi, letak access.log dan listener.log, parameter koneksi *database* aplikasi, parameter kesalahan, dan parameter koneksi ke *database* Oracle.

Menu Help

Menu terakhir yaitu menu *Help* merupakan menu yang berisi penjelasan tentang aplikasi ini dan cara ,menggunakannya. Pada menu *Help* terdapat dua buah submenu, yaitu *Manual* dan *About* .

Submenu Start, Start and Exit, dan Exit

Pada menu *File* terdapat submenu *Start/Run..., Start and Exit,* dan *Exit* yang digunakan untuk menjalankan dan menghentikan *monitoring* serta untuk keluar dari aplikasi. Saat kita menjalankan *monitoring*, status aplikasi akan berubah menjadi *running* dan indikator proses akan menunjukkan proses yang sedang berjalan. Untuk dapat mengetahui detil proses yang berjalan dapat dilakukan dengan cara menjalankan aplikasi ini melalui konsol. Administrator *server* Siakad Unila dapat menghentikan *monitoring* oleh aplikasi ini pada saat *monitoring* dapat dilakukan secara manual.

Pada saat proses *monitoring* dilakukan, ada beberapa proses yang berjalan pada aplikasi, yaitu menyimpan data dari access.log ke *database*, menganalisa data access.log tersebut dari *database*, menyimpan data dari listener.log ke *database*, menganalisa data listener.log dari *database*, melakukan koneksi ke *web server* Apache, melakukan koneksi ke *database server* Oracle, dan *idle* selama *delay* yang ditentukan sebelum mengulangi proses pertama.

Apabila aplikasi menemukan kondisi yang dianggap sebagai indikasi *error*, maka aplikasi akan mengirimkan pesan dalam bentuk SMS yang terlihat pada gambar-gambar di bawah ini :

Gambar 11. Pesan kesalahan pada access.log

Bila terdapat *status code* yang dianggap sebagai indikasi *error*, maka akan dikirimkan SMS pesan kesalahan dengan format : *Server Error Has Occured at* (*datetime YYYY-MM-DD HH:MM:SS*) *from IP*: (*IP address*) *with status code* (*status code*)

Gambar 12. Pesan kesalahan pada listener.log

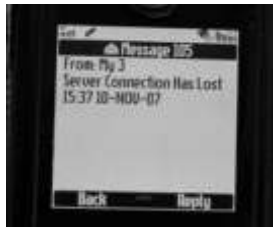
Untuk indikasi akses ilegal pada *database server*, dihasilkan pesan kesalahan berupa SMS dengan format : *Illegal Access Has Occured at* (*datetime YYYY-MM-DD HH:MM:SS*) *from IP :* (*IP address*).

Gambar 13. Pesan kesalahan *web server overloaded*

Pada saat indikasi *web server* yang *overloaded*, aplikasi akan mengirimkan SMS dengan isi pesan : *Web Server Overloaded*.

Gambar 14. Pesan kesalahan database server overloaded

Ketika *database server* berada pada kondisi *overloaded*, maka aplikasi ini akan mengirimkan pesan kesalahan : *Database Server Overloaded*.



Gambar 15. Pesan putusnya koneksi ke *web server*

Pada saat aplikasi menguji koneksi ke *web server* Apache yang terputus, maka aplikasi ini akan mengirimkan SMS berisi pesan kesalahan : *Server Connection Has Lost*.

Gambar 16. Pesan putusnya koneksi ke *database server*

Bila ditemukan kondisi terputusnya koneksi ke *database server* Oracle, maka pesan kesalahan : *Connection to Oracle Has* Lost yang akan dikirimkan.

Submenu Conn. Parameter

Menu *Configuration* digunakan untuk melakukan konfigurasi terhadap beberapa parameter agar aplikasi dapat berjalan seperti yang diharapkan. Parameter pertama yang perlu dikonfurasi adalah parameter koneksi pada submenu *Conn. Parameter*. Submenu ini mengatur s*erial port* yang digunakan, *baud rate* ponsel, nomor ponsel administrator, *server IP*, dan *delay* tiap eksekusi pada proses *monitoring*. Pada pengaturan *serial port*, aplikasi akan mendeteksi *serial port* yang tersedia pada komputer. Sedangkan *baud rate* ditentukan sendiri oleh administrator berdasarkan ponsel yang digunakan. Nomor ponsel administrator yang diisikan pada pengaturan nomor ponsel administrator harus mengikuti format yang diberikan agar dapat dikenali dengan baik oleh sistem.

Gambar 17. Submenu Conn. Parameter

Pada pengaturan lainnya, ditentukan alamat IP *server* yang akan diakses oleh aplikasi dan pengaturan waktu *delay* yang akan digunakan oleh aplikasi untuk berada dalam keadaan *idle* sebelum aplikasi melakukan *monitoring* kembali. Untuk pengaturan *delay*, nilai yang dimasukkan harus dalam satuan *millisecond*.

Submenu Logs Path

Submenu *Logs Path* ini digunakan untuk menentukan *path* dari *log file* yang akan dianalisa oleh aplikasi ini, baik access.log maupun listener.log dan jumlah akses maksimal yang diijinkan pada kedua *log file* tersebut. Untuk menentukan *path* dapat dilakukan dengan melakukan *browse* pada *dialog box* yang telah tersedia. Sedangkan pengaturan jumlah akses maksimal dimaksudkan untuk menjadikan hal tersebut sebagai salah satu parameter *error* pada *server*. Hal ini berkaitan dengan kehandalan *server* dilihat dari sisi beban. Diharapkan dengan adanya pengaturan ini, dapat ditentukan batas kritis *server* sebelum *server* berada pada kondisi *overloaded*.

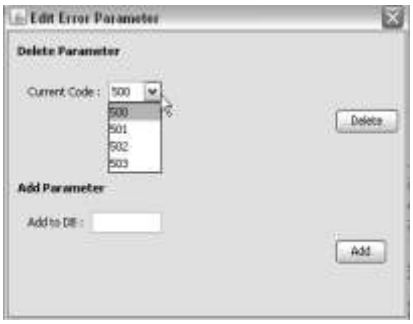
Gambar 18. Submenu Logs Path

Submenu DB Connection

Submenu selanjutnya dari menu *Configuration* adalah *DB Connection* yang mengatur *setting* koneksi aplikasi ke *database* aplikasi ini sendiri, yaitu *database* MySQL. URL, *username*, dan *password* dapat dikonfigurasi disini agar aplikasi dapat mengakses *database* untuk melakukan pengambilan data dan proses *input, update, delete* sesuai kebutuhan.

Submenu Error Code

Submenu *Error Code* merupakan fitur tambahan yang memungkinkan administrator sebagai *user* menentukan sendiri *status code* pada access.log yang dianggap sebagai *error code*. Pada submenu ini, administrator dapat melakukan *input* dan *update* terhadap *status code* yang telah ditentukan sebelumnya.



Gambar 19. Submenu Error Code

Submenu Oracle Home
Serupa dengan submenu *Error Code*, submenu ini juga merupakan fitur tambahan dari hasil pengujian pertama aplikasi ini. Submenu ini digunakan dalam proses *monitoring*, dalam hal ini koneksi ke *database server*, yaitu Oracle. Pada submenu ini ditentukan parameter-parameter yang dibutuhkan untuk melakukan koneksi ke Oracle. Secara sederhana, proses yang terjadi saat *monitoring* koneksi ke Oracle aplikasi akan mencoba melakukan koneksi ke *database* Oracle dan akan mengirimkan *error report* apabila koneksi gagal.

3. Pengujian
Tahap pengujian sebenarnya telah dilakukan pada tahap implementasi, yaitu *small testing*. *Small testing* dilakukan untuk menguji *class-class* yang dihasilkan selama proses *coding*. Saat baris-baris kode dalam setiap *class* dibuat, selalu dilakukan pengujian untuk mengetahui setiap fungsi yang dibuat telah sesuai dengan tujuannya. Pengujian keseluruhan (*system testing*) baru dilakukan setelah seluruh proses *coding* telah dilakukan. Pengujian ini dilakukan untuk menguji keberhasilan seluruh fungsi pada aplikasi. *Small testing* terbagi menjadi dua, yaitu *unit testing* dan *integration testing*.

a. Unit testing
Unit testing merupakan pengujian terhadap setiap unit dari *software* yang dibuat, dalam hal ini setiap *class*. Pengujian tiap *class* dilakukan pada saat implementasi sehingga setiap *class* yang dibuat akan diuji terlebih dahulu, baik *class* yang digunakan untuk memasukkan data ke dalam *database* maupun *class* untuk menganalisa data tersebut dan menemukan indikasi kesalahan. Dalam pengujian ini ditemukan beberapa permasalahan yang terjadi, diantaranya adalah pada *class* AccessLogReader dan ListenerLogReader. Pada AccessLogReader terjadi beberapa kali kesalahan *coding* untuk melakukan pengambilan data dari access.log ke *database*. Salah satu kesalahan yang terjadi adalah kesalahan penggunaan *delimiter* untuk mendapatkan data dari access.log. Kesalahan lain terjadi karena perbedaan pola pada beberapa access.log.

b. **Integration testing**
Pada tahap ini akan diuji bagaimana setiap *class* yang dibuat secara independen dapat berkolaborasi. Pengujian akan dilakukan pada *class-class* yang menggunakan *class* lain dalam *method*-nya. Salah satunya adalah pengujian terhadap interaksi *class* ApacheConnTest yang menguji koneksi ke *web server* dengan dua buah *class* yang digunakan dalam pengiriman SMS yaitu SeriaToGsm dan ComputeSmsData. Pengujian dilakukan dengan cara mengeksekusi *class* ApacheConnTest pada saat *web server* dimatikan. Dari hasil pengujian ini diketahui bahwa ketiga *class* ini telah dapat berkolaborasi, dibuktikan dengan terkirimnya SMS berisi pesan kesalahan koneksi *web server* yang terputus.

c. **System testing**
Setelah seluruh *class* beserta *Graphical User Interface* (GUI) telah selesai dibuat, dilakukan pengujian keseluruhan terhadap sistem. Pengujian terpenting pada tahap ini adalah pengujian *use case*. Terdapat tujuh *use case* yang dibuat pada analisa kebutuhan, yaitu menjalankan *monitoring* dan konfigurasi aplikasi (koneksi, *logs*

path, *database* aplikasi, parameter *error*, dan koneksi Oracle).

E. Kesimpulan dan Saran

Kesimpulan

1. Aplikasi *Monitoring* Server Siakad Unila Berbasis SMS ini dapat melakukan pengawasan terhadap kinerja *server* pada saat administrator server tidak dapat melakukan pengawasan *server*.
2. Administrator akan mendapatkan *report* berupa SMS apabila terdapat indikasi kinerja *server* yang buruk.
3. Terdapat *delay* yang cukup besar antara waktu kejadian dan waktu ditemukannya indikasi kinerja *server* yang buruk karena terdapat beberapa proses yang dilakukan oleh aplikasi, termasuk *sleep time*.
4. Aplikasi akan melakukan *monitoring* terhadap semua aspek yang telah ditentukan dalam analisa kebutuhan sehingga administrator tidak dapat menentukan sendiri pilihan *monitoring* yang perlu dilakukan.

Saran

1. Untuk mencegah *delay* yang semakin besar, diharapkan agar administrator melakukan *backup* data access.log dan listener.log.
2. Diharapkan dapat dilakukan pengembangan terhadap aplikasi ini, terutama untuk mengurangi *delay* dan pembebanan terhadap *server* termasuk dengan membuat opsi *monitoring* yang perlu dilakukan.
3. Diharapkan dapat juga dilakukan pengembangan yang bertujuan untuk meningkatkan fleksibilitas aplikasi ini, diantaranya dengan menambahkan *class* yang bertanggung jawab untuk melakukan analisa terhadap *log file* dengan format yang lain dan *class* yang dapat menguji koneksi ke *database server* lain, misalnya MySQL.

Daftar Pustaka

[1.] http://en.wikipedia.org/wiki/Webserver, akses 14 Agustus 2006.
[2.] http://en.wikipedia.org/wiki/Database_server , akses 14 Agustus 2006.
[3.] Kyte, Thomas, 2005, *Expert Oracle Database Architecture 9i and 10g Programming Techniques and Solutions*, Apress®, California.
[4.] http://en.wikipedia.org/wiki/Short_message_service, akses 16 Agustus 2006.
[5.] http://www.pcmag.com, akses 16 Agustus 2006.
[6.] Suhendar, A. dan Gunadi, H., 2002, *Visual Modelling Menggunakan UML dan Rational Rose*, Penerbit Informatika, Bandung.
[7.] Kolling, Michael, and Barnes, David, 2002, *Object First with Java*, Pearson Education/ Prentice Hall.
[8.] Pressman, Roger S.,Ph.D., 2001, *Software Engineering*, Mc Graw-Hill, New York.
[9.] O' Docherty, Mike , 2005, *Object-Oriented Analysis and Design*, John Wiley & Sons, Ltd., England.
[10.] Gunawan, Ferry, 2003, *Membuat Aplikasi SMS Gateway Server dan Client dengan Java dan PHP*, PT. Elex Media Komputindo, Jakarta.
[11.] Deitel, H.M, and Deitel, P.J, 2004, *Java™ How to Program, sixth edition*, Prentice Hall, New Jersey.
[12.] http://www.funsms.net/sms_tutorial.htm, akses 22 Agustus 2006.
[13.] http://computer.howstuffworks.com/web-server.htm, akses 14 Agustus 2006.
[14.] http://www-group.slac.stanford.edu/wim/logfiles/info.html, akses 13 Jun 2006

.