NATURAL QUERY JALUR BUS BERBASIS SMS MENGGUNAKAN TEKNOLOGI JAVA

TUGAS AKHIR

Diajukan sebagai Salah Satu Syarat
Untuk Memperoleh Gelar Sarjana
Program Studi Teknik Informatika

OLEH :

Nama : **Agung Wahyu Dewantoro**

No. Mhs : **00 523 177**

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INDUSTRI
UNIVERSITAS ISLAM INDONESIA
YOGYAKARTA
2007

LEMBAR PENGESAHAN DOSEN PEMBIMBING

NATURAL QUERY JALUR BUS

BERBASIS SMS

MENGGUNAKAN TEKNOLOGI JAVA

TUGAS AKHIR

Oleh:

Nama : Agung Wahyu Dewantoro

No mahasiswa : 00 523 177

Yogyakarta, 15 Mei 2007

Dosen Pembimbing

Taufiq Hidayat, ST, MCS.

LEMBAR PENGESAHAN PENGUJI

NATURAL QUERY JALUR BUS BERBASIS SMS MENGGUNAKAN TEKNOLOGI JAVA

TUGAS AKHIR

Oleh :

Nama : **Agung Wahyu Dewantoro**

No. Mahasiswa : **00 523 177**

Telah Dipertahankan di Depan Sidang Penguji Sebagai Salah Satu Syarat

Untuk Memperoleh Gelar Sarjana Program Studi Teknik Informatika

Fakultas Teknologi Industri Universitas Islam Indonesia

Yogyakarta, Juni 2007

Tim Penguji.

(Taufiq Hidayat, ST., MCS.)
Ketua

(Sri Kusumadewi, S.Si., MT.)
Anggota I

(Hendrik, ST.)
Anggota II

Mengetahui,

Ketua Program Studi Teknik Informatika

Universitas Islam Indonesia

Yudi Prayudi, S.Si., M.Kom.

LEMBAR PERNYATAAN KEASLIAN

HASIL TUGAS AKHIR

Saya yang bertandatangan di bawah ini,

Nama : Agung Wahyu Dewantoro

No. Mahasiswa : 00 523 177

Menyatakan bahwa seluruh komponen dan isi dalam Laporan Tugas Akhir ini adalah hasil karya saya sendiri. Apabila dikemudian hari terbukti bahwa ada beberapa bagian dari karya ini adalah bukan hasil karya sendiri, maka saya siap menanggung resiko dan konsekuensi apapun.

Demikian pernyataan ini saya buat, semoga dapat dipergunakan sebagaimana mestinya.

Yogyakarta, **Juni 2007**

(Agung Wahyu Dewantoro)

HALAMAN PERSEMBAHAN

Keempat Orang Tuaku...
Kelima Adik-Adikku...
Diah Rahmawati.

Seluruh Guru Dalam Hidupku...
Semua Orang Yang Percaya Dan Mendukungku...

MOTTO

hidup adalah perjuangan !!!

success is a journey, not destination !!!
bila kita masih bernafas, kehidupan bukan suatu titik,
melainkan suatu koma....

" Aku tak ingin seperti pohon bambu, bila terkena angin goyah
kemana angin bertiup....
Aku ingin seperti pohon Oak, menjulang tinggi menantang angin .... "
(Gie)

KATA PENGANTAR

بِسْمِ اللَّهِ الرَّحْمَٰنِ الرَّحِيمِ

Assalaamu'alaikum Wr. Wb.

Alhamdulillah, puji syukur kehadirat Allah SWT, yang karena limpahan rahmatnya, sehingga Laporan Tugas Akhir *Natural Query* Jalur Bus Berbasis SMS Menggunakan Teknologi Java, ini dapat diselesaikan.

Dalam Laporan Tugas Akhir ini, penulis banyak mendapat dukungan oleh berbagai pihak, ucapan terima kasih penulis haturkan dengan ikhlas kepada :

1. Bapak Fathul Wahid, ST., M.Sc., selaku Dekan FTI.
2. Bapak Yudi Prayudi, S.Si., M.Kom., selaku Ketua Jurusan Teknik Informatika.
3. Bapak Taufiq Hidayat, ST., MCS., selaku dosen pembimbing yang telah memberikan banyak bimbingan dan arahan sehingga Tugas Akhir ini dapat diselesaikan.
4. Seluruh staff dosen Teknik Informatika Universitas Islam Indonesia.
5. Bapak-bapak Sopir dan Kondektur seluruh jalur bus, yang banyak direpotkan oleh penulis.
6. Ibuku atas do'a dan dukungannya yang tanpa putus, serta Pak Koco yang setia mengawal kami selama ini.
7. The Cindils Brotherhood, adik-adikku semua (Evi, Dewi, Risa, Arief dan Wisnya) kalian harus sukses dunia-akhirat.

8. Diah Rahmawati, SIP atas menunggu dan dukungannya selama ini.
9. Teman – teman di Teknik Informatika angkatan 00 dan semuanya (Fero "Strez Inc", Nanda blink, Fuad Trans, Angga Boim, Yasin DNA, Q-mpull, Ivan BRI, Anam Rembang *Romance,* Arie Kazao, Sigit '01 Siemens, Wapon) terima kasih atas kerjasama, *sharing* ilmunya. Semua pihak yang telah membantu secara langsung dan tidak langsung dalam penulisan Laporan Tugas Akhir ini yang tidak bisa disebutkan satu persatu.
10. Teman-teman IMMP 03, CBNP UH, Pabhama Corp XIII, terima kasih atas semprot dan *support* kalian.
11. Motor Legenda-ku, akhirnya.

Penulis menyadari bahwa penulisan Tugas Akhir ini belum sempurna dan masih banyak kekurangan , oleh karena itu diharapkan kritik dan saran agar laporan ini lebih baiksehingga dapat berguna untuk menambah pengalaman dan pengetahuan dimasa yang akan datang.

Wassalaamualaikum Wr. Wb

Yogyakarta, 15 Mei 2007

(Agung Wahyu Dewantoro)

SARI

Kemajuan teknologi komunikasi berkembang sangat pesat seiring dengan bayaknya permintaan masyarakat akan teknologi tersebut. Salah satunya adalah *Global System for Mobile Communication* (GSM) atau yang sering disebut dengan telepon digital. *Short Messaging Service* (SMS) merupakan salah satu fitur dari GSM yang merupakan media komunikasi jarak jauh yang paling bayak digunakan sekarang ini dikarenakan biaya yang murah dan proses cepat.

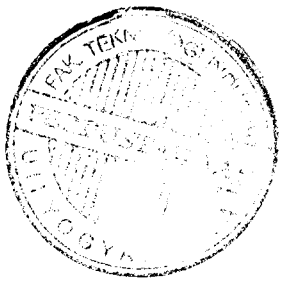
Tidak terkecuali pada kehidupan sehari-hari, masyarakat membutuhkan suatu sistem yang dapat memandu *user* (pengirim SMS) dengan mengirimkan *Short Mesaging Service* (SMS) ke sistem untuk memandu *user* tersebut mengenai informasi jalur bus.

Implementasi sistem *Natural Query* Jalur Bus ini, menggunakan teknologi *Natural Language Processing* (NLP) yang merupakan teknologi yang memungkinkan untuk melakukan berbagai macam pemrosesan terhadap bahasa alami yang biasa digunakan oleh manusia *(human)*.

Key word : *Short Mesaging Service* (SMS), *Natural Language Processing* (NLP), Bahasa Alami, Jalur Bus, *Request*.

TAKARIR

Admin	pengurus, yang mempunyai hak akses penuh
Artificial intelligence	keerdasan buatan
Baudrate	laju, laju baut
Client	klien
Concatenation	penyambungan, merangkai
Compiler	kompiler, penyusun, perakit
Context Diagram	mendefinisikan ruang lingkup dan menggambarkan aliran data dengan entitas-entitas di luar sistem
Context Free Grammar	Tata Bahasa Bebas Konteks
Cross platform application	aplikasi bebas *platform*
Database	sekumpulan berkas yang saling terkait dan membentuk suatu bangun data
Database server	sever pangkalan data, server pelayanan data
DFD	*Data Flow Diagram*, gambaran sistem sebagai jaringan kerja fungsi yang berhubungan satu sama lain dengan aliran dan penyimpanan data
Error	kesalahan, kekeliruan
Finite automata	automata terbatas
Flowchart	diagram alir
Form	bentuk/formulir
Graphic User Interface	antarmuka pengguna grafis



Hand Phone	telepon genggam, telepon selular
Hard disk	cakram keras, tempat penyimpanan data
Hardware	perangkat keras
Hostname	tuan rumah, nama hos, nama induk
ID	*ID*, identitas
Identifier	pengenali, pengidentifikasi
Input	masukan
Interface	anatarmuka
Keyword	kata kunci
Login	proses masuk ke sistem
Logout	proses keluar dari sistem
Natural Language Processing	Pengolahan Bahasa Alami
Natural Query	permintaan, pertanyaan alami
Network Programming	pemrograman jaringan
OOP	*Object Oriented Programming*, pemrograman berorientasi objek
Output	keluaran
Password	sandi, kata sandi, sandi lewat
Plat form	teras, bidang
Port	port, sarana penghubung
Primary Key	kunci primer, kunci utama
Processing	pemrosesan
Protocol Data Unit	Unit Data Protokol

Query	permintaan, pertanyaan
Reply	saji ulang, main ulang, balasan
Request	meminta
Server	pelayan, pembantu, *server*
Software	perangkat lunak
Source code	kode sumber
State	keadaan
Store and forward	simpan dan meneruskan
String	sirkuitan, *string*
Switching Circuit	sirkuit pindah, sirkuit alih
User	pengguna
User friendly	memudahkan pengguna
User Name	nama pengguna
Validity	kesahihan, validitas

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB I

PENDAHULUAN

1.1 Latar Belakang Masalah

Teknologi komunikasi berkembang sangat pesat seiring dengan bayaknya permintaan masyarakat akan teknologi tersebut. Salah satunya adalah *Global System for Mobile Communication* (GSM) atau yang sering disebut dengan telepon digital.

Short Messaging Service (SMS) merupakan salah satu fitur dari GSM yang merupakan media komunikasi jarak jauh yang dikembangkan oleh *Europan Telecomunication Standart Institute* (ETSI) yang paling bayak digunakan sekarang ini dikarenakan biaya yang murah dan proses cepat. Meski telah banyak pula fitur-fitur dari hand phone lainnya seperti MMS, EMS dan lain-lain, tetapi keberadaan SMS tidak ditinggalkan begitu saja. [WAH05]

Banyaknya masyarakat yang melakukan transfer data, maka media SMS sangat baik digunakan untuk transfer data atau informasi dalam kapasitas kecil. Dengan media ini, para pengguna khususnya *hand phone* bisa mengetahui informasi yang dibutuhkan dari server, misalnya Sistem Informasi Akademik yang berbasis SMS. Para mahasiswa mengirim SMS dengan kode tertentu ke *server* yang kemudian secara otomatis di-*reply* oleh server tersebut mengenai informasi yang diinginkan oleh mahasiswa yang mengirimkan SMS sesuai dengan kode-kode yang diterima oleh *server*.

Dalam tugas akhir ini, akan dicoba suatu sistem, dimana akan memberikan suatu *output* berupa informasi kepada *user* dalam hal ini pengguna handphone mengenai informasi jalur-jalur bus via SMS yang dikirimkan ke server, dengan menggunakan teknologi *Natural Language Processing* (NLP) yang merupakan

teknologi yang memungkinkan untuk melakukan berbagai macam pemrosesan terhadap bahasa alami yang biasa digunakan oleh manusia *(human)*.

1.2 Rumusan Masalah

Dari penjelasan Latar Belakang Masalah di atas adalah bagaimana merancang suatu sistem yang memberikan *output* kepada *client* (pengirim SMS) berupa informasi jalur-jalur bus yang akan ditumpangi oleh *client*, dengan menggunakan teknologi *Natural Language Processing* (NLP).

1.3 Batasan Masalah

Dalam membangun sistem ini, batasan masalah yang diberikan adalah :

1. Dalam sistem *Natural Query* Jalur Bus ini, hanya memberikan keluaran berupa informasi jalur-jalur bus yang harus ditumpangi dari posisi pengirim SMS *(client)* berada, sampai tujuan yang diinginkan.
2. Bahasa yang digunakan adalah Bahasa Indonesia.
3. Jalur Bus hanya dibatasi untuk wilayah kota Yogyakarta.
4. Koneksi antara PC dan *handphone* menggunakan kabel data.
5. Data disimpan di *database*.
6. Aplikasi dibangun dengan bahasa J2SE sebagai *compiler*nya, dan menggunakan MySQL sebagi *database*-nya.
7. Aplikasi sistem ini digunakan pada *handphone* bermerk Siemens.

1.4 Tujuan Penelitian

Tujuan dari Tugas Akhir ini adalah :

1. Membangun suatu aplikasi dalam hal ini suatu sistem yang memberikan *output* berupa informasi kepada pengirim SMS (*client*) berupa informasi jalur-jalaur bus kota yang akan ditumpangi menuju tempat yang diinginkan via SMS.
2. Mempelajari serta memanfaatkan teknologi Java, khususnya J2SE.
3. Mengenal dan memahami cara kerja SMS.

1.5 Metode Penelitian

Metode penelitian yang dipakai dalam tugas akhir ini adalah :

1.5.1 Metode Pengumpulan Data

Metode pengumpulan data ini adalah sebagai berikut :

1. Metode *Observasi*

   Metode pengamatan, pengumpulan data yang dilakukan dengan melakukan pengamatan secara langsung ke lapangan, pengumpulan data secara langsung berdasarkan sumber-sumber yang ada di lapangan.

2. Metode *Interview* (wawancara)

   Metode wawancara, pengumpulan data dilakukan dengan mengadakan wawancara secara langsung dengan pihak-pihak yang terkait untuk memperoleh data yang tepat.

3. Metode *Library Research* (Kepustakaan)

   Metode kepustakaan, pengumpulan data yang dilakukan dengan mengumpulkan data lewat buku-buku dan *website* yang relevan dengan permasalahan yang dihadapi.

1.5.2 Metode Pengembangan Perangkat Lunak

Dalam pengembangan sistem informasi jalur bus via SMS ini menggunakan metode struktur data, dengan alat dan teknik yang dibutuhkan dalam pengembangannya sehingga sistem hasil analisis menghasilkan sistem yang strukturnya dapat didefinisaikan, meliputi :

a. Analisis Kebutuhan

Analisis kebutuhan adalah analisis yang dilakukan untuk mengetahui kebutuhan perangkat keras dan perangkat lunak dalam proses penelitian.

b. Perancangan Sistem

Perancangan sistem merupakan tahapan yang dilakukan untuk membuat sebuah perangkat lunak untuk mengetahui *input* dan *output* yang dibutuhkan agar sesuai dengan apa yang diinginkan. Perancangan ini terdiri dari beberapa tahapan, yaitu :

1. Pembuatan diagram alir data, menjelaskan mulai dari konteks diagram sampai dengan DFD turunan sebagai penjelasan dari rancangan program atau sistem yang akan dibuat.
2. Perancangan *database* yang akan digunakan untuk menyimpan data atau informasi yang sesuai dengan DFD yang telah dibuat.
3. Perancangan *interface* (antar muka) yang berguna untuk memberikan gambaran terhadap *software* yang akan dibuat.
4. Pemodelan sistem yang digunakan untuk menampilkan data atau informasi yang diinginkan.

c. Implementasi Sistem

Setelah pembuatan perancangan maka dapat dipresentasikan hasil perancangan yang telah dibuat. *Software* dibuat berdasarkan perancangan yang telah disetujui.

d. Pengujian Sistem

Pengujian dilakukan setelah implementasi sistem tersebut selesai untuk mengetahui relasi dari *software* yang dibuat.

e. Analisis Hasil Implementasi dan Pengujian Sistem

Diperoleh setelah implementasi dan pengujian sistem dilakukan untuk mengetahui hasil dari implementasi tersebut untuk kemudian disempurnakan.

1.6 Manfaat Penelitian

Manfaat dari penelitian ini, diharapkan sistem dapat memberikan kemudahan bagi *user* dalam hal ini *client* atau pengirim SMS berupa *output* berupa informasi jalur-jalur bus yang harus ditumpangi sampai tujuan yang diinginkan.

1.7 Sistematika Penulisan

Untuk mempermudah dalam penulisan tugas akhir ini maka dalam penyusunannya penulis memberikan sistematika penulisan berdasarkan bab demi bab yang berurutan berdasarkan pokok-pokok permasalahan yang terbagi menjadi tujuh bab, yaitu :

BAB I PENDAHULUAN

Bab ini merupakan pengantar terhadap masalah-masalah yang akan dibahas seperti latar belakang masalah, perumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, metode penelitian dan sistematika penulisan.

BAB II LANDASAN TEORI

Bab ini merupakan bagian yang menjadi landasan teori yang digunakan dalam membahas dan memecahkan masalah yang akan dipecahkan dengan memuat landasan teori atau metode yang berhubungan dengan sistem yang akan dibuat, yang meliputi teori – teori mengenai perkembangan JAVA, khususnya J2SE, NetBeans, AT Commmands, Cara Kerja SMS, Teori Bahasa Otomata (TBO), serta mengenai teknologi *Natural Language Processing* (NLP). Dimana semuanya menjadi landasan dalam melakukan penelitian.

BAB III ANALISIS KEBTUHAN PERANGKAT LUNAK

Mengemukakan analisis kebtuhan perangkat lunak yang meliputi: metode analisis, hasil analisis, fungsi-fungsi yang digunakan, analisis kebutuhan *input*, kebutuhan *output*, kebutuhan antar muka, analisis kebutuhan perangkat keras, serta kebutuhan bahasa pemrograman.

BAB IV PERANCANGAN PERANGKAT LUNAK

Tahap ini memuat tentang tahapan perancangan sistem yang dapat digunakan untuk aplikasi *Natural Query* jalur bus kota berdasar hasil analisis kebutuhan. Perangkat lunak. Pada tahap ini meliputi metode perancangan yang meliputi DFD, hubungan dan perancangan tabel, dan hasil perancangan serta perancangan antar muka (*interface*).

BAB V IMPLEMENTASI PERANGKAT LUNAK

Tahap ini memuat batasan- batasan implementasi perangkat lunak, meliputi *form* tampilan dari sistem dan bagian ini juga memuat keterangan cara pemakaian serta penggunaaan perangkat lunak tersebut.

BAB VI ANALISIS KERJA PERANGKAT LUNAK

Tahap ini memuat pengujian terhadap sistem, yang merupakan dokumentasi hasil pengujian normal, pengujian tidak normal, termasuk penanganan kesalahan algoritma dan fungsinya sehingga dapat diperbaiki.

BAB VII PENUTUP

Memuat kesimpulan dari proses pengembangan perangkat lunak, baik pada tahap analisis kebutuhan, perancangan, implementasi, dan analisis kerja perangkat lunak, serta berisi saran yang perlu diperhatiakan berdasar keterbatasan yang diketemukan dalam penelitian.

BAB II

LANDASAN TEORI

2.1 Sekilas Tentang Pemrograman Java

2.1.1 Java Secara Umum

Java dikembangkan oleh Sun Microsystem pada Agustus 1991, dengan nama **Oak.** Konon Oak adalah pohon sejenis jati yang terlihat dari jendela pembuatnya, James Gosling. Ada yang mengatakan bahwa Oak adalah singkatan dari *"Object Application Kernel"*, tetapi ada yang menyatakan nama itu muncul setelah nama Oak diberikan. Pada Januari 1995, karena nama Oak dianggap kurang komersil, maka diganti menjadi Java.

Program Java bersifat tidak bergantung pada *platform,* artinya Java dapat dijalankan pada sembarang Sistem Operasi (OS). Ketidakbergantungan terhadap *platform* sering dinyatakan dengan istilah portabilitas. Yang menarik, tingkat portabilitas Java tidak hanya sebatas pada program sumber *(Source Code),* melainkan juga pada tingkat kode biner yang disebut *bytecode.* Dengan demikian bila Anda telah mengkompilasi program Java pada komputer bersistem operasi Windows, Anda dapat menjalankan hasil kompilasinya pada Mancintosh secara langsung, tanpa perlu mengkompilasi ulang. [KAD04]

Sun membagi arsitektur Java menjadi 3 bagian :

1. *Enterprise Java* (J2EE) untuk apliaksi web, aplikasi sistem tersebar denangan beraneka ragam *client* dengan kompleksitas yang tinggi. Merupakan super set dari standart java.
2. *Standard Java* (J2SE), biasa dikenal dengan bahasa Java, digunakan untuk menjalankan dan mengembangkan aplikasi Java pada level PC.

3. *Micro Java* (J2ME), biasa dijalankan pada aplikasi ber-*device* kecil, yang pengaplikasiannya biasa digunakan pada level semisal *Handphone* dan PDA.

2.1.2 J2SE

J2SDK *Standard Edition* atau dikenal dengan J2SE, menyediakan lingkungan pengembangan yang kaya fitur, stabil, aman, *cross platform aplication*. Fitur ini mendukung konektefitas basis data, rancangan antar muka (*user interface*), *Input*, *Output*, dan pemrograman jaringan (*network programming*) termasuk sebagai paket-paket dasar bahasa Java.

2.2 SMS

Short Message Service (SMS), merupakan salah satu fasilitas yang diberikan oleh operator dan juga merupakan layanan teknologi nirkabel digital yang popular. SMS mempunyai kemampuan mengirim dan menerima pesan dari/ ke *handphone*. Teks dapat terdiri dari kata-kata atau bilangan atau kombinasi *numeric*. Masing-masing pesan pendek terdiri dari 160 karakter dengan mengguankan huruf latin, sedang hanya mampu 70 karakter, jika menggunakan huruf Cina atau Arab.

SMS merupakan salah satu fitur dari GSM yang dikembangkan dan distandarisasi oleh ETSI. Pada saat kita mengirim pesan SMS dari *handphone*, maka pesan SMS itu tidak langsung dikirim ke HP tujuan, akan tetapi terlebih dahulu ke SMS *Center* (SMSC) dengan prinsip *Store and Forward*, setelah itu baru dikirim ke *handphone* yang dituju. [WAH05]

2.3 SMS Center (SMSC)

Pada saat pesan SMS dikirim dari *handphone (Mobile Orginated)* pesan tersebut tidak langsung dikirim ke *hanphone* tujuan *(Mobile Terminated)* , tetapi terlebih dulu dikirim ke SMS *Center* (SMSC), baru kemudian pesan tersebut dikirimkan ke *handphone* tujuan. [GUN03]

Melalui keberadaan SMSC, kita dapat mengetahui status dari SMS yang dikirim, apakah telah sampai atau gagal diterima oleh *handphone* tujuan. Apabila *handphone* tujuan dalam keadaan aktif dan menerima SMS yang dikirim, ia akan mengirim kembali pesan konfirmasi ke SMSC yang menyatakan bahwa SMS telah diterima. Kemudian SMSC mengirim kembali status tersebut kepada si pengirim. Tetapi bila *handphone* tujuan mati atau diluar jangkauan, SMS yang dikirim akan disimpan di SMSC sampai periode validitas terpenuhi, dari *handphone* tujuan. Di samping itu, SMSC juga akan mengirim pesan informasi ke nomor pengirim yang menyatakan pesan yang dikirim belum diterima atau gagal. [WAH05]

2.4 Protocol Data Unit (PDU)

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 asli yang dituliskam pada saat akan mengirim pesan. Sesungguhnya mode teks ini adalah pengkodean dari mode PDU. Sedangkan mode PDU adalah format pesan dalam bentuk octet heksadesimal dan oktet semidesimal dengan panjang mencapai 160 (7 bit) atau 140 (8bit) karakter. Di Indonesia tidak semua operator GSM maupun terminal yang mendukung mode teks, sehingga mode yang digunakan adalah mode PDU. Dalam pengiriman pesan terdapat dua jenis *mobile,* yaitu ***Mobile***

Terminated (*Handphone* Penerima) dan ***Mobile Originated*** (*Handphone* Pengirim). [WAH05]

2.4.1 SMS PDU Pengirim (*Mobile Originated*)

SMS PDU Pengirim adalah pesan yang dikirim dari *handphone* ke terminal yang kemudian dikirimkan ke SMSC. Pada prinsipnya apabila kita mengirim pesan ke nomor tujuan, nomor itu akan melalui SMSC.

Pesan yang akan dikirimkan ke terminal masih dalam bentuk teks, sedangkan dalam pengiriman ke SMSC harus dalam bentuk PDU. Untuk itu sebelum dikirim, terminal atau HP akan melakukan perubahan dari format teks menjadi format PDU, proses ini sering disebut ***encoded*** . Adapun skema dari format PDU Pengirim telah diatur dan ditetapkan oleh ETSI seperti pada gambar 2.1.

SCA	PDU Type	MR	DA	PID	DCS	VP	UDL	UD

Gambar 2.1 Skema Format SMS PDU Pengirim

Misalnya kita mengirim pesan SMS ke nomor 628122898840 dengan isi pesan "Pesan Pendek" dengan batas waktu pengiriman 5 hari (waktu penyimpanan pesan di SMSC, jika nomor tujuan tidak bisa menerima pesan) 5 hari, maka format PDU adalah :

0010111C912618229888040000AB0CD0F23CEC06C1CB6E72790D

2.4.1.1 Service Center Addres (SCA)

SCA adalah informasi dari alamat (nomor) SMSC. SCA memiliki tiga komponen utama, yaitu *len, type of number,* dan *service number.* Dalam pengirim pesan SMS, nomor SMSC tidak dicantumkan terlihat pada tabel 2.1.

Tabel 2.1 Service Center Address

Oktet	Keterangan	Hasil
Len	Panjang informasi SMSC dalam octet	00
Type of Number	Format nomer dari SMSC 81 hexa = format lokal 91 hexa = format internasional	<none>
Service center number	Nomer SMSC dari operatorpengirim. Jika panjangnya ganjil maka pada karakter terahir ditambah 0F hexa.	<none>

2.4.1.2 PDU Type

Nilai *default* pada PDU Type untuk SMS pengirim adalah 11 hexa, yang memiliki arti bahwa 11 hexa = 00000100. Nilai *default* tersebut dapat kita lihat pada tabel 2.2 berikut ini.

Tabel 2.2 PDU Type

Bit no	7	6	5	4	3	2	1	0
Nama	RP	UDHI	SRR	VPF	VPF	RD	MTI	MTI
Nilai	0	0	0	1	0	0	0	1

Keterangan :

RP : *Reply Path*. Parameter yang menunjukkan bahwa alur jawaban ada.

UDHI : *User Data Header Indicator*. Bit ini bernilai 1 jika data pengirim dimulai dari suatu judul /tema.

SRR : *Status Report Request*. Bit ini bernilai 1 jika status laporan pengiriman diminta.

VPF : *Validity Period Format*. Format dari batas pengiriman jika pesan gagal diterima.

00 → Jika pesan tidak disimpan di SMSC.

10 → Format relative (satu oktet).

01 → Format Enhanced (tujuh oktet).

11 → Format absolute (tujuh oktet).

RD : *Reject Duplicates*. Parameter yang menandakan ya atau tidaknya *Service Center* akan menerima suatu pegiriman pesan SMS untuk suatu pesan yang masih disimpan di Service Center tersebut. Ia mempunyai MR dan DA yang sama sebagai pesan dikirimkan dari OA yang sama.

MTI : *Message Type Indicator*. Bit bernilai 0 untuk menunjukkan bahwa PDU ini adalah suatu SMS-DELIVER.

2.4.1.3 Message Reference (MR)

Message Reference adalah acuan dari pengatauran SMS. Untuk membiarkan pengaturan SMS dilakukan sendirir ole HP tujuan, maka nilai yang diberikan adalah "00". Jadi pada MR hasilnya adalah 00.

2.4.1.4 Destination Address (DA)

DA adalah alamat (nomor) tujuan yang terdiri dari panjangnya nomor tujuan *(Len)*, format dari nomor tujuan *(Type Number)*, dan nomor tujuan *(Destination Number)*, dapat dilihat pada tabel 2.3.

Tabel 2.3 Destination Address

Oktet	Nilai	Hasil
Len	12	OC
Type Of Number	Format Internasioanal	91
Destination Number	628122898840	261822988804

Jadi pada DA hasinya adalah OC91261822988804

2.4.1.5 Prortocol Identifer (PID)

Protocol Identifer adalah tipe atau format dari cara pengiriman pesan, yang biasanya diatur dari HP pengirim. Misalnya tipe data Standard Text, Fax, E-mail, Telex, X400, dan lain-lainnya.

Nilai *default* dari PID adalah 00 = "Standard Text". Pada contoh ini, pesan SMS yang akan dikirim menggunakan format teks standard, jadi pada *Protocol Identifer* hasilnya adalah 00.

2.4.1.6 Data Coding Scheme (DCS)

Data Coding Scheme adalah rencana dari pengkodean data untuk menentukkan kelas dari pesan tersebut adalah berupa SMS teks standard, Flash SMS, atau Blinking SMS. Pada contoh ini pesan SMS yang dikirim berupa teks standard, jadi pada DCS hasilnya adalah 00.

2.4.1.7 Valididtiy Period (VP)

Validity Period adalah waktu pesan SMS disimpan di SMSC apabila pesan tersebut gagal diterima di HP penerima, dapat kita lihat pada tabel 2.4.

Tabel 2.4 Validity Period

Waktu VP	Nilai VP
5 menit – 720 menit (12 jam)	(Waktu VP / 5) – 1
12,5 jam – 24 jam	143+ ((Waktu VP – 12)*2)
2-30 hari	166 + Waktu VP
Lebih dari 4 minggu	192 + Waktu VP

Pada contoh di atas, waktu VP- nya 5 hari, maka nilai VP adalah 166 + 5 = 171 d = AB h. Jadi pada *Validity Period* hasilnya adalah AB.

2.4.1.8 User Data Length (UDL)

User Data Length adalah panjangnya pesan SMS yang akan dikirim dalam bentuk teks standard. Pada contoh ini SMS yang dikirim adalah "Pesan Pendek", yang memiliki 12 karakter (0Ch). Jadi pada *User Data Length* hasilnya adalah 0C.

2.4.1.9 User Data (UD)

UD adalah isi pesan yang akan dikirim dalam format heksadesimal. Pengkodean dari nilai teks standard menjadi heksadesimal dilakukan dengan bantuan *Default Alfabeth* yang dibakukan oleh ETSI GSM 03.3. [WAH05]

2.4.2 SMS PDU Penerima (Mobile Terminated)

SMS PDU Penerima adalah terminal penerima pesan yang datang atau masuk dari SMSC ke HP dalam format PDU. Pada prinsipnya pesan yang kita terima dari SMSC masih dalam format PDU setelah itu terminal HP yang menerima pesan akan melakukan pengkodean menjadi teks, proses ini sering disebut proses ***decoded***. Cara pengkodean format PDU sudah diatur dan distandarkan oleh ETSI. Format PDU dari SMS Penerima, seperti pada gambar 2.2.

SCA	PDU Type	OA	PID	DCS	SCTS	UDL	UD

Gambar 2.2 Skema Format SMS PDU Penerima

Contoh: Kita menerima pesan dari 62811888374 dengan isi pesan SMS adalah "hellohello" pada tanggal 6 Januari 2004 pukul 16.22WIB. Maka format PDU adalah: 06912618010000040C91261822883847000040106061220282 0AE83299BFD4697D9EC37

2.4.2.1 Service Center Address (SCA)

SCA adalah alamat (nomor) dari SMSC. SCA memiliki tiga komponen utama, yaitu *len, type of number, dan service center number*. Pada contoh nilai dari SCA diatas adalah 06912618010000, dapat kita lihat pada tabel 2.5.

Tabel 2.5 Service Center Address

Oktet	Keterangan	Nilai
Len	Panjang informasi SMSC dalam octet	06
Type of Number	Format nomor dari SMSC 81 hexa = format lokal 91 hexa = format internasional	91
Service Center Number	Nomor SMSC dari operator pengirim. Jika panjangnya ganjil maka pada karakter terakhir	2618010000

	ditambahkan 0F hexa. Satelindo = 62816124 (PDU = 26181642) Telkomsel = 6281100000 (PDU = 261801000) Excelcom = 62818445009 (PDU = 2618455400F9) IM3 = 62855000000 (PDU = 2658050000F0)	

2.4.2.2 PDU Type

Nilai *default* dari PDU Type untuk SMS adalah 04 hexa, yang memiliki arti 04 hexa= 00000100, dapat kita lihat di tabel 2.6.

Tabel 2.6 PDU Type

Bit No	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Nama	RP	UDHI	SRI	<nn>	<nn>	MMS	MTI	MTI
Nilai	*0*	*0*	*0*	*0*	*0*	1	*0*	*0*

Keterangan :

RP : *Reply Path*. Parameter yang menunjukkan bahwa alur jawaban ada.

UDHI :*User Data Header Indicator*. Bit ini bernilai 1 jika data pengirim dimulai dengan suatu judul / tema.

SRI : *Status Report Indicator*. Bit ini bernilai 1 jika status laporan akan dikembalikan ke SME.

MMS : *More Messages to Send*. Bit ini bernilai 0 jika ada pesan lebih yang akan dikirim.

MTI : *Message Type Indicator*. Bit bernilai 0 unuk menunjukan bahwa PDU ini adalah suatu SMS-Deliver.

2.4.2.3 Originator Address (OA)

OA adalah alamat (nomor) dari si pengirim, yang terdiri atas panjangnya nomor pengirim (*Len*), format dari nomor pengirim (*Type Number*), dan nomor pengirim (*Originator Number*). Nilai dari OA pada contoh di atas adalah 0C91261822883847. Dan dapat kita lihat pada tabel 2.7.

Tabel 2.7 Orginator Address

Oktet	Keterangan	Nilai
Len	Panjang nomor pengirim	0C
Type of Number	Format dari nomor pengirim 81 hexa = format lokal 91 hexa = format internasioanal	91
Orinator number	Nomor pengirim dari operator pengirim. Jika panjangnya ganjil maka pada karkter terkhir ditambah 0F hexa.	261822883847

2.4.2.4 Protocol Identifier (PID)

Protol Identifier adalah tipe atau format dari cara pengiriman pesan, yang biasanya diatur dari HP pengirim. Misalnya tipe standar teks, Fax, email, Telex, X400, dan lain-lainnya.

Nilai *default* dari PID adalah 00 = "Standad Text". Untuk contoh diatas nilai dari PID adalah 00, sehingga pesan yang diterima adalah teks standard

2.4.2.5 Data Coding Scheme (DCS)

Data Coding Scheme adalah rencana dari pengkodean data untuk menentukan kelas dari pesan tersebut apakah berupa SMS teks standard, Flash SMS, atau Blinking SMS. Pada contoh di atas DCS adalah 00 yang berarti bahwa pesan yang diterima merupakan pesan teks standard.

2.4.2.6 Service Center Time Stamp (SCTS)

STCS adalah waktu dari penerimaan pesan oleh SMSC penerima. SCTS terdiri atas tahun, bulan, tanggal, jam, menit dan detik, serta zona waktu. Nilai SCTS pada contoh diatas adalah 40106061220282. Dan dapat kita lihat pada tabel 2.8.

Tabel 2.8 Service Center Time Stamp

Nama	Nilai	Hasil
Year	40	04 (2004)
Month	10	05 (Mei)
Date	60	06
Hour	61	16
Minute	22	22
Second	02	20
Time Zone	82	28, dimana 1 unit = 15 menit. Jadi (15x28)/ 60= 7 jam. Sehingga menjadi GMT + 07.00 = WIB.

Dari tabel di atas terlihat bahwa pesan diterima oleh SMSC pada tanggal 16 Mei 2004 pukul 16:22':20" WIB.

2.4.2.7 User Data Length (UDL)

User Data Length adalah panjang dari pesan yang diterima dalam bentuk teks standard. Pada contoh nilai dari UDL adalah 0A, yang berarti pesan yang diterima adalah sebanyak 10 karakter.

2.4.2.8 User Data (UD)

User Data adalah pesan yang diterima dalam format hexadecimal. Pada contoh diatas nilainya adalah E8329BFD4697D9EC37. pengkodean dari nilai hexadecimal menjadi teks standard dengan bantuan tabel ASCII yang dapat dilihat pada tabel 2.9.

Tabel 2.9 User Data

Nilai	Oktet (8 bit)	Septet (7 bit)	Dec	Hasil
E8	**1** 1101000	1101000	104	H
32	**00** 110010	110010 **1**	101	E
9B	**100** 11011	11011 **00**	108	L
FD	**1111** 1101	1101 **100**	108	L

46	01000110	110 1111	111	O
97	10010111	1101000	104	H
D9	11011001	1100101	101	E
EC	1 1101100	1101100	108	L
37	00 110111	1101100	108	L
		110111 1	111	O

Dari tabel diatas terlihat bahwa hasil nilai heksadesimal dari EB329BFD4697D9EC37 adalah "hellohello". Ini berarti pesan yang diterima adalah "hellohello". [WAH05]

Tabel 2.10 Kode ASCII

Desimal	0	1	2	3	4	5	6	7	8	9
0										
1	LF			CR						
2										
3			SP	!	"	#	$	%	&	'
4	(	)	*	+	,	-	.	/	0	1
5	2	3	4	5	6	7	8	9	:	;
6	<	=	>	?	@	A	B	C	D	E
7	F	G	H	I	J	K	L	M	N	O
8	P	Q	R	S	T	U	V	W	X	Y
9	Z	[	\	]	^	_	`	A	b	C
10	d	E	F	G	H	i	J	k	l	M
11	n	O	P	Q	R	s	T	u	v	W
12	x	Y	Z	{	|	}	~	DEL		

2.4.3 AT Command

HP seperti Nokia, Siemens, atau Sonny Ericson memiliki modem *(Modulation Demodulation)*, seperti komputer yang memiliki modem untuk mengirimkan data melalui jaringan telepon. Jadi HP yang memiliki modem akan dapat dikoneksikan ke komputer melalui kabel data, sehingga dapat berfungsi sebagai pengirim data, misalnya fax. Untuk menggunakan mikrokontroler juga dikoneksikan melalui kabel data dan mikrokontroler akan berkomunikasi dengan kabel ini. Perintah standard modem dekenal dengan *AT Command*, perintah ini dapat digunakan untuk mengirim,

menerima/membaca dan menghapus SMS, dan dapat digunakan untuk mengambil informasi lainnya. Beberapa jenis HP memiliki extended *AT Command* yang bisa digunakan untuk mengambil informasi HP, model HP, nomor IMEI, SIM IMSI, status baterai, kekuatan sinyal, nama operator, lokasi, dan cell ID. [ATC02]

2.4.4 NetBeans IDE

Netbeans adalah sebuah proyek *open source* yang didukung oleh banyak orang di dunia, sebuah komunitas *open source* yang tumbuh dengan pesat dengan hampir 100 patner di seluruh belahan dunia. Sun Microsystems mendirikan proyek *open source* ini di bulan Juni pada tahun 2000 dan hingga sekarang masih terus aktif menjadi sponsor utama. Hingga sekarang, terdapat 2 produk aktif proyek ini, yaitu : **NetBeans IDE** dan **NetBeans Platform**.

Dalam NetBeans, pemrograman dilakukan dilakukan berbasis visual dan *event-driven*. Persis seperti Borland Delphi dan Microsoft Visual Studio.

Untuk membuat dialog atau *user-interface*, kita tidak perlu membuat teks program secara manual baris per baris, tetapi cukup klik pada *component pallete*. Program akan dihasilkan secara otomatis.

NetBeans mencakup *compiller* atau *builder*, dan *debuger* internal. Hal ini sangat memudahkan paskaperancangan program. Proses *deployment* dan atau tes dapat dilakukan dari dalam NetBeans. [SRI07]

2.4.5 Natural Language Processing

Pada prinsipnya bahasa alami adalah suatu bentuk representasi dari suatu pesan yang ingin dikomunikasikan antar manusia. Bentuk utama representasinya adalah berupa suara/ucapan (*spoken language*), tetapi sering pula dinyatakan dalam bentuk tulisan.

Bahasa dapat dibedakan menjadi (1) Bahasa Alami, dan (2) Bahasa Buatan. Bahasa alami adalah bahasa yang biasa digunakan untuk berkomunikasi antar manusia, misalnya bahasa Indonesia, Sunda, Jawa, Inggris, Jepang, dan sebagainya. Bahasa buatan adalah bahasa yang dibuat secara khusus untuk memenuhi kebutuhan tertentu, misalnya bahasa pemodelan atau bahasa pemrograman komputer.

Chomsky adalah orang yang pertama kali merepresentasikan bahasa sebagai rangkaian simbol. Chomsky berhasil memperlihatkan bahwa bahasa apapun dapat direpresentasikan dengan suatu cara yang universal. Pemikiran Chomsky yang merepresentasikan bahasa sebagai kumpulan simbol-simbol dan aturan yang mengatur susunan simbol-simbol tersebut telah membuka peluang untuk melakukan pemrosesan bahasa secara simbolik dengan teknologi komputer, sehingga melahirkan bidang ilmu ***Natural Language Processing*** **(NLP)**.

Dipandang dari sisi implementasi teknologinya, pemrosesan bahasa lisan dan tulisan adalah sangat berbeda. Bahasa lisan lebih banyak melakukan pemrosesan bunyi atau suara, sedangkan bahasa tulisan lebih banyak melakukan pemrosesan simbol-simbol tertulis. Sebagai akibatnya, penelitian di bidang bahasa lisan pada awalnya lebih banyak dilakukan dalam bidang "*signal processing*", sedangkan penelitian-penelitian untuk pemrosesan bahasa tulisan lebih banyak dilakukan dalam bidang "*artificial intelligence*" atau kecerdasan buatan yang pada prinsipnya melakukan *symbolic processing*. Saat ini, teknologi yang berkaitan dengan pemrosesan bahasa alami ini sering disebut sebagai "*speech and language technology*", "*natural language processing technology*", "*human language technology*", atau dalam beberapa pertemuan ilmiah para peneliti di bidang ini

di Indonesia, menyepakati penggunaan istilah "teknologi bahasa" untuk menyebut teknologi ini. Dari segi keilmuan, bidang ini dikenal sebagai bidang "natural language processing" atau" *computational linguistic*". [ERW05]

2.4.6 Teori Bahasa Otomata

Dikemukakan pertama kali oleh **McCulloch** dan **Pits** pada sebuah mesin abstraks sederhana, bernama *finite automata* untuk memodelkan *neuron nets*. Finite automata juga digunakan untuk merancang *switching circuit*. Studi mengenai teori otomata terkait bidang-bidang lain di ilmu komputer. Kemudian, ekivalensi antara *finite automata* dan ekspresi regular *(regular ekspression)* dikemukakan Sthepen Kleene. Sejak saat itu, teori bahasa otomata dikaitkan secara erat dengan teori bahasa formal. Hubungan erat antara teori otomata dengan teori koding *(coding theory)* juga banyak diteliti.

Saat ini *finite automata* digunakan untuk menyelesaikan beragam persoalan perangkat lunak terutama pada perancangan kompilator serta pengolahan *teks/string*. *Finite automata* dapat dinyatakan sebagai pengenal bahasa. Bahasa yang dikenali oleh *finite automata* adalah bahasa sederhana namun mempunyai aspek penerapan sangat penting di ilmu informatika/komputer. Seperti *Turing Machine* seperti komputer modern saat ini. *Turing machine* dapat mengolah masukan (simbol-simbol di *tape*) dan menghasilkan keluaran (simbol-simbol yang berada di *tape*-nya setelah berakhirnya sebarisan pergerakan).

Karena banyak yang berperan dalam pengembangannya, bidang teori ini diberi aneka ragam nama, yaitu teori otomata *(theory of otomata)* teori bahasa formal *(theory of formal language)*, atau teori mesin Turing *(theory of Turing machine)*. [BAM04]

2.4.6.1 Terminologi di Teori Bahasa

Terminologi dasar yang penting dalam memahami teori bahasa adalah alpabet, penyambungan (*Concatenation*) dan *string* pada alpabet V. alpabet adalah himpunan simbol (karakter) tak kosong yang berhingga. Alpabet digunakan untuk membentuk kata-kata (*string-string*) di bahasa. Ketika kita membicarakan bahasa, kita harus memulainya dengan alpabet. Pada beberapa buku, alpabet dilambagkan dengan Σ.

Dan bila kita membandingkan istilah "bahasa" seperti yang telah didefinisikan terhadap bahasa-bahasa (alami – bahasa manusia) secara tertulis yang telah dikenal. Kita dapat menyatakan bahwa bahasa Inggris tidak sekedar kumpulan kata-kata, namun dapat dipandang sebagai kumpulan kalimat-kalimat yang legal. Jika kita mengadopsi pendekatan ini, maka alpabet tidak hanya berisi 26 huruf, tapi juga termasuk simbol, spasi dan tanda baca. [BAM04]

2.4.6.2 Finite State Automata (FSA)

Bahasa formal dapat dipandang sebagai entitas abstrak, yaitu sekumpulan *string-string* simbol alpabet tertentu. Namun kita juga dapat memandang bahasa sebagai entitas-entitas abstrak yang dapat dikenali atau dibangkitkan melalui suatu mesin komputasi. Istilah mengolah berarti mengenali atau membagkitkan. Bahasa paling sederhana adalah bahasa regular. Mesin yang dapat mengenali bahasa kelas ini adalah *finite (state) automata. Finite (state) automata* adalah model komputasi.

Finite automata adalah model matematika sistem dengan masukkan dan keluaran diskrit. Sistem dapat berada disalah satu dari sejumlah berhingga konfigurasi internal yang disebut dengan *state. State* sistem merupakan ringkasan informasi yang berkaitan denagn masukan-masukan berikutnya. *Finite automata* sangat cocok untuk memodelkan sistem dengan jumlah *state* yang berhingga.

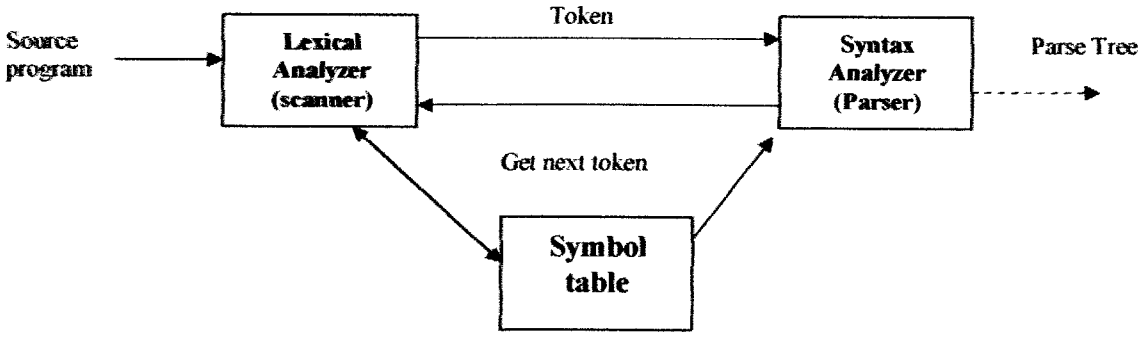
Didalam *finite automata* terdapat klasifikasi yang mampu mengenali himpunan regular secara presisi. Dengan demikian, kedua *finite automata* itu dapat mengenali *string-string* yang ditunjukkan dengan ekspresi regular secara tepat. *Finite automata* dapat berupa:

1. *Deterministic finite automata* (DFA)
2. *Nondeterministic finite automata* (NDFA) yang berarti lebih dari satu transisi dari satu *state* dimungkinkan pada simbol masukan yang sama. [BAM04]

2.4.6.3 Lexical Analyzer (Scanner)

Penganalisis sintaks (*parser*) memperoleh *string token* dari *lexical analyzer* dan memverifikasi string berdasar *grammar* bahasa sumber. Sebelumnya yang perlu diketahui adalah bahwa *lexical analyzer* ini atau biasa yang disebut *scanner* atau *tokenizer*, menerjemahkan menjadi masukan menjadi bentuk yang berguna untuk tahap-tahap kompilasi berikutnya. *Lexical analyzer* merupakan atarmuka antara kode sumber program dan penganalisis sintaks (*parser*). *Scanner* melakukan pemeriksaan karakter per karakter pada teks masukan, memecah program menjadi bagian-bagian yang disebut dengan token. Dan tugas pokok dari *lexical analyzer* adalah mengubah program sumber yang dipandang sebagai barisan *byte*/karakter menjadi barisan *token*. *Token* merupakan unit atau elemen dasar bahasa komputer (seperti 'kata' di bahasa manusia). Dan *token* merupakan unit yang tidak dapat terbagi lagi.

Hubungan anatara *parser* dengan *lexical analyzer* yaitu, *parser* (*syntax analyzer*) berfungsi menghasilkan pohon sintaks program sumber yang didefinisikan *grammar*. Simbol terminal pohon sintaks adalah token-token yang dihasilkan *scanner*. Token-token dihasilkan dengan cara memisahkan program sumber tersebut dilewatkan ke *parser*.



Gambar 2.4 Hubungan Scanner dan Parser

Lexical analyzer merupakan komponen kompilasi yang independen yang berkomunikasi dengan *parser* lewat antarmuka yang terdefinisi dengan bagus dan sederhana. Terlihat pada gambar diatas. [BAM04]

2.4.6.4 Context Free Grammar (CFG)

Context Free Grammar atau Tata Bahasa Bebas Konteks (TBBK) ini seperti halnya pada tata bahasa regular, sebuah tata bahasa bebas konteks ini adalah suatu cara yang menunjukkan bagaimana menghasilkan untai-untai dalam bahasa. Pada saat menurunkan suatu *string*, simbol-simbol variabel akan mewakili bagian-bagian yang belum terturunkan dari *string* tersebut. Pada tata bahasa bebas konteks (CFG), bisa terdapat banyak bagian yang belum terturunkan itu, dan bisa terjadi dimana saja. Dan CFG ini menjadi dasar dalam pembetukan suatu *parser* atau proses analisis sintaks. Bagian sintaks dalam suatu kompilator kebanyakan didefinisikan dalam tata bahasa bebas konteks. [UDI01]

Tata Bahasa Bebas Konteks (TBBK) atau CFG ini memiliki aturan:

1. Terminal atau simbol dasar yang tidak dapat diturunkan lagi dan terminal disebut juga token (dilambangkan dengan huruf kecil: a, b, c, d, dan seterusnya).

2. Non terminal merupakan variabel sintak yang masih dapat diturunkan lagi (dilambangkan dengan huruf besar: A, B, C, D, dan seterusnya).
3. Hanya memiliki satu buah simbol variabel terminal di ruas kiri ($\alpha \rightarrow \beta$).

Contoh aturan produksi yang termasuk TBBK atau CFG terlihat pada aturan produksi dibawah ini :

S => AB

A => aA | a

B => bB | B

Sebuah Parser akan membentuk Pohon Sintaks (*Parse Tree*). Parser merupakan bentuk implementasi dari TBBK. Sebuah *tree* adalah suatu *graph* terhubung yang tidak sirkuler dan memiliki satu buah *root* (akar) dan dari situ memiliki lintasan ke setiap simpul (daun/*leaf*).

Parse Tree berfungsi untuk menggambarkan bagaimana memperoleh suatu *string* dengan cara menurunkan simbol-simbol variabel menjadi simbol-simbol terminal, sampai tidak ada simbol yang belum tergantikan.

Pohon penurunan *Parse Tree* (*Parse Tree Derivation*) berguna menggambarkan bagaimana memperoleh suatu *string* (untai) dengan cara menurunkan simbol-simbol variabel menjadi simbol-simbol terminal, sampai tidak ada yang belum tergantikan. [UDI01]

Contoh Pohon *Parse* pada *Context Free Grammar*, dapat dilihat dibawah ini:

S => aS

S => bT

T => a

Maka misalkan untuk string "**aaba**" maka TBBK atau CFG diatas dapat diturunkan

menjadi :

S –> aS

S => aaS

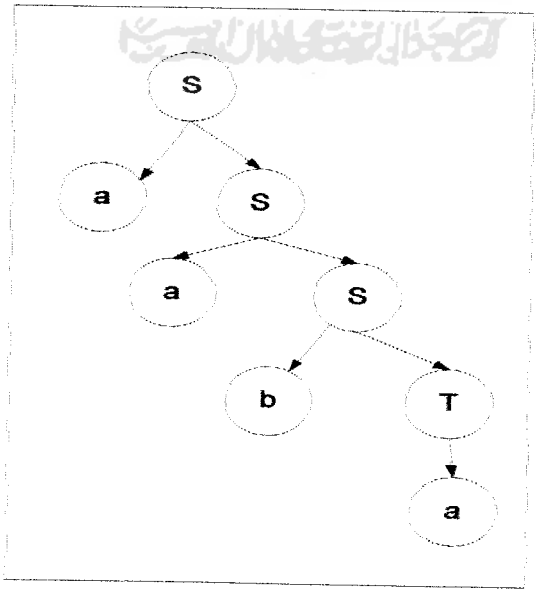
S => aabT

S => aaba

Artinya string "**aaba**" cocok dan diterima oleh TBBK diatas.

Akan kita gambarkan Pohon Sintaksnya, untuk memperoleh untai "aaba". Pada pohon tersebut simbol awal akan menjadi akar (*root*). Setiap kali penurunan dipilih aturan produksi yang menuju ke solusi. Simbol-simbol variabel akan menjadi simpul-simpul yang mempunyai anak. Simpul-simpul yang tidak mempunyai anak akan menjadi simbol terminal. [FIR01]

Kalau kita baca simbol terminal yanag ada pada gambar 2.6 dari kiri ke kanan akan diperoleh untai "aaba".



Gambar 2.6 Pohon Penurunan untuk untai "aaba"

Parse Tree Derivation dapat dilakukan dengan dua cara :

1. Dengan penurunan terkiri : nonterminal terkiri yang disubstitusi.
2. Dengan penurunan terkanan : nonterminal terkanan yang disubstitusi.

Contoh penurunan untuk implemementasi CFG dengan aturan produksi dapat dilihat pada contoh sebagai berikut :

S => **aAS** | **a**

A => SbA | **ba**

Untuk *string* **"aabbaa"** :

1. Dengan penurunan terkiri : S => aAS => aSbAS => aabAS=> aabbaS => **aabbaa**.
2. Dengan penurunan kanan : S => aAS => aAa => aSbAa =>aSbbaa => **aabbaa.**

BAB III

ANALISIS KEBUTUHAN

3.1 Metode Analisis

Analisis suatu sistem merupakan salah satu proses yang harus dilakukan dalam perancangan dan implementasi suatu perangkat lunak, untuk mengidentifikasi dan mengevaluasi permasalahan, kesempatan dan hambatan yang terjadi dan kebutuhan-kebutuhan yang dibutuhkan sehingga dapat diusulkan perbaikan kedepannya, dan untuk memperoleh sistem aplikasi yang lebih baik dikemudian hari. Metode pengembangan sistem yang digunakan dalam analisis ini adalah dengan *oriented methodologies*, metode ini menekankan karakteristik dari data yang akan diproses. Alat yang digunakan adalah *Context Level Data Flow Diagram* (DFD). Metode pengumpulan data menggunakan teknik wawancara, observasi serta *library research*.

3.1.1 Metode Pengumpulan Data

Dalam penelitian ini metode pengumpulan data yang digunakan berupa wawancara, observasi serta *library research*.

3.1.1.1 Metode Wawancara

Wawancara (*interview*) adalah semacam percakapan bertujuan untuk memperoleh informasi. Wawancara dilakukan pada Dinas Perhubungan, Propinsi Daerah Istimewa Yogyakarta serta sopir dan kondektur bus kota dengan tujuan dilakukan wawancara adalah untuk mengetahui jalur bus kota, yang beroperasi di kota Yogyakarta, serta jalan yang dilewati, dari berangkat sampai tujuan akhirnya.

3.1.1.2 Metode Observasi

Metode observasi atau pengamatan merupakan salah satu metode pengumpulan data yang cukup efektif. Observasi merupakan pengamatan langsung yang bertujuan untuk menguji validitas dari data hasil wawancara yang telah didapatkan.

3.1.1.3 Metode Kepustakaan

Metode kepustakaan atau *library research*, merupakan salah satu metode yang digunakan dalam melakukan penelitian ini. Dengan melakukan studi kepustakaan dengan mengumpulkan data-data lewat buku serta situs-situs *website* yang relevan dengan permasalahan yang dihadapi.

3.1.2 Kebutuhan Antarmuka

Antar muka pemakai atau *user interface* adalah bagian penghubung antara program sistem informasi dengan pemakai. Kebutuhan terhadap antar muka yang diinginkan sebaik mungkin bersifat *user friendly*, artinya pengguna dapat menggunakan perangkat lunak yang dibuat dengan mudah dan senyaman mungkin untuk mendapatkan suatu informasi yang diinginkan oleh *user*.

3.1.3 Kebutuhan Keamanan Data

Keamanan data merupakan salah satu unsur yang penting yang harus dipertimbangkan dalam proses desain suatu sistem. Karena suatu sistem tanpa keamanan data yang baik akan merugikan sistem itu sendiri, sebab data akan bebas diakses oleh pihak-pihak yang tidak bertanggung jawab.

3.2 Hasil Analisis Kebutuhan

Dari hasil analisis yang diperoleh dari Sistem *Natural Query* Jalur Bus Kota dengan Teknologi Java ini, ada beberapa proses masukan data dan proses keluaran data.

3.2.1 Data Masukan (*Input*)

Data *input* atau masukan dalam Sistem *Natural Query* Jalur Bus Kota, adalah sebagai berikut :

a. Data armada jalur bus yang beroperasi di kota Yogyakarta.

   Contoh : Jalur 1, Jalur 2, Jalur 3, dan seterusnya.

b. Data nama jalan-jalan di kota Yogyakarta.

c. Kata kunci atau *key word* dari nama jalan-jalan di kota Yogyakarta.

   Contoh : Jl. Malioboro *key word*-nya yaitu : Hotel Garuda, Malioboro Mall, Gedung DPRD, KFC, Hotel Mutiara, Toko Liman.

d. Data daftar jalur operasi bus kota yang beroperasi di Yogyakarta.

   Contoh : Jalur 1 melewati : Terminal Giwangan – Lingkar Selatan – Jl. Wonosari – Jl. Gedong Kuning – Jl. Ngeksigondo – Jl. M. Supeno – Jl. Kol. Sugiono – Jl. MT. Haryono – Jl. Sugeng Jeroni - Jl. Tendean – Jl. HOS. Cokroaminoto – Jl. Kyai Mojo – Jl. Sudirman – Jl. Cik Ditiro – Bunderan UGM.

e. Tempat awal (dari) dan akhir (tujuan) dari *user.*

3.2.2 Kebutuhan Proses

Kebutuhan proses adalah kebutuhan pengolahan data dari *input* data yang diberikan kepada sistem. Pengolahan data tersebut adalah :

1. Proses pemasukan data armada jalur bus yang beoperasi di kota Yogyakarta

   Pada proses ini, data yang yang dimasukkan kemudian diproses adalah data jalur apa saja yang beroperasi di kota Yogyakarta. Data armada tersebut adalah Jalur 1 samapai Jalur 16.

2. Proses pemasukan data nama jalan-jalan kota Yogyakarta

   Proses *ini* adalah dimana sistem menerima proses input berupa nama jalan-jalan yang berada di wilayah kota Yogyakarta.

3. Proses pemasukan kata kunci (*key word*) jalan-jalan di kota Yogyakarta

   Pada Proses pemasukan kata kunci jalan-jalan yang berada di wilayah kota Yogyakarta ini, dengan cara meng-identifikasi seluruh jalan-jalan tersebut sesuai ciri-ciri yang berada pada jalan yang bersangkutan, untuk memudahkan *user* serta sistem dalam mengolah SMS.

4. Proses pemasukan daftar jalur bus yang beroperasi di Yogyakarta

   Dalam proses pemasukan daftar jalur bus *ini*, harus melewati proses *input* nama jalan-jalan di kota Yogyakarta dan proses *input* data armada yang beroperasi di kota Yogyakarta, dengan memilih jalur bus dan jalan-jalan yang dilewati oleh armada bus yang bersangkutan.

5. Proses pencarian posisi awal dan tujuan dari *user*

   Proses *ini* merupakan proses akhir untuk mencari posisi dari *user* (pengirim SMS), dari posisi awal serta tujuan dari *user*, sesuai dengan data-data yang telah dimasukkan pada proses *input* diatas yang kemudian terhubung dengan *database*.

3.2.3 Kebutuhan Keluaran (*Output*)

Keluaran yang dihasilkan berupa informasi kepada *user* (pengirim SMS) tentang informasi jalur bus yang harus ditumpangi oleh pengirim SMS dari dan sampai ke tujuan yang diinginkan.

3.2.4 Kebutuhan Antar Muka (*Interface*)

Kebutuhan antarmuka (*Interface*) untuk suatu aplikasi yang akan dibuat didapatkan dari hasil wawancara dengan pihak-pihak yang terkait atau dari hasil studi lapangan, dan aplikasi lain yang sejenis. Kebutuhan antar muka harus bersifat *user friendly*, artinya bahwa pengguna aplikasi yang akan dihasilkan bisa mudah untuk mengoperasikannya.

3.2.5 Kebutuhan Keamanan Data

Keamanan data dapat dilakukan dengan pembuatan tabel pengguna (*user*) yang disertai dengan sandi (*password*) sehingga hanya admin yang namanya tercantum dalam tabel tersebut dan yang mengetahui *password* saja yang dapat memasukkan data-data yang baru atau merubah data.

3.2.6 Kebutuhan Perangkat Lunak

Aplikasi pada penelitian tugas akhir ini dikembangkan dengan perangkat lunak yaitu, Java JDK 1.5.0_06, NetBeans 5.0, Misrosoft Visio, Siemens Data Suite, *Database* MySQL yang berjalan pada sistem operasi Microsoft Windows XP Professional.

3.2.7 Kebutuhan Perangkat Keras

Perangkat keras yang digunakan dalam pengembangan aplikasi penelitian tugas akhir ini adalah satu unit laptop dengan spesifikasi sebagai berikut:

1. Prosessor Intel Pentium M processor 1,7GHz
2. 40 GB hard drive
3. Memory 256 MB DDR2
4. VGA 64 MB
5. DVD/CD-RW Combo

BAB IV

PERANCANGAN PERANGKAT LUNAK

4.1 Perancangan Perangkat Lunak

Metode perancangan yang digunakan dalam perangkat lunak ini adalah metode *Data Flow Diagram (DFD)*.

4.2 Hasil Perancangan

Berdasarkan perancangan yang dibuat setelah melakukan analisis terhadap kebutuhan perangkat lunak, dapat diketahui apa saja yang menjadi masukan perangkat lunak, keluaran perangkat lunak, dan proses yang terjadi dalam perangkat lunak. Proses-proses tersebut dapat dilihat dengan menggunakan DFD (*Data Flow Diagram*).

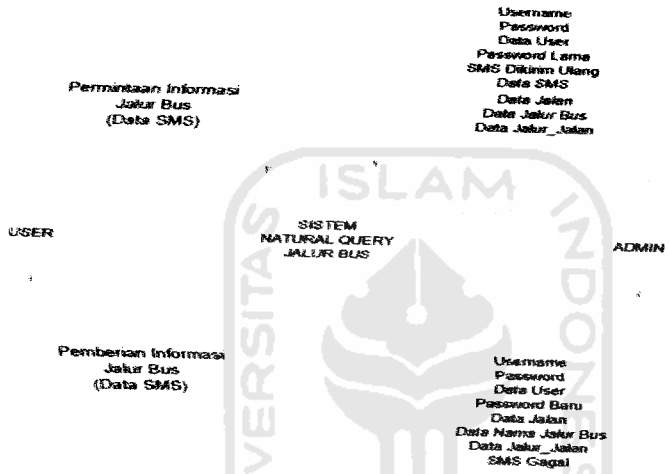
4.2.1 Perancangan Data Flow Diagram

Data Flow Diagram (DFD) merupakan diagram yang menggunakan notasi – notasi untuk menggambarkan arus data dari sistem secara logika. DFD sering digunakan untuk menggambarkan lingkungan fisik dimana data tersebut mengalir atau lingkungan fisik dimana data tersebut disimpan. DFD merupakan alat yang digunakan pada metodologi pengembangan sistem yang terstruktur. DFD merupakan alat yang cukup popular sekarang ini karena dapat menggambarkan arus data di dalam sistem dengan terstruktur dan jelas.

Dari perancangan DFD ini dapat diketahui proses apa saja yang terjadi di dalam sistem, sehingga dapat diperoleh gambaran atau langkah-langkah kerja yang dapat digunakan untuk membuat sistem yang terstruktur.

4.2.1.1 Diagram Konteks

Diagram konteks mengandung suatu proses dan beberapa entitas yang mewakili proses dari seluruh sistem. Dari analisa yang dilakukan, diperoleh diagram konteks seperti terlihat pada gambar 4.1. Diagram ini menerangkan mengenai gambaran sistem secara umum, yang berisi informasi jalur bus kota yang berada di wilayah Yogyakarta yang dibutuhkan pengguna atau *user*.

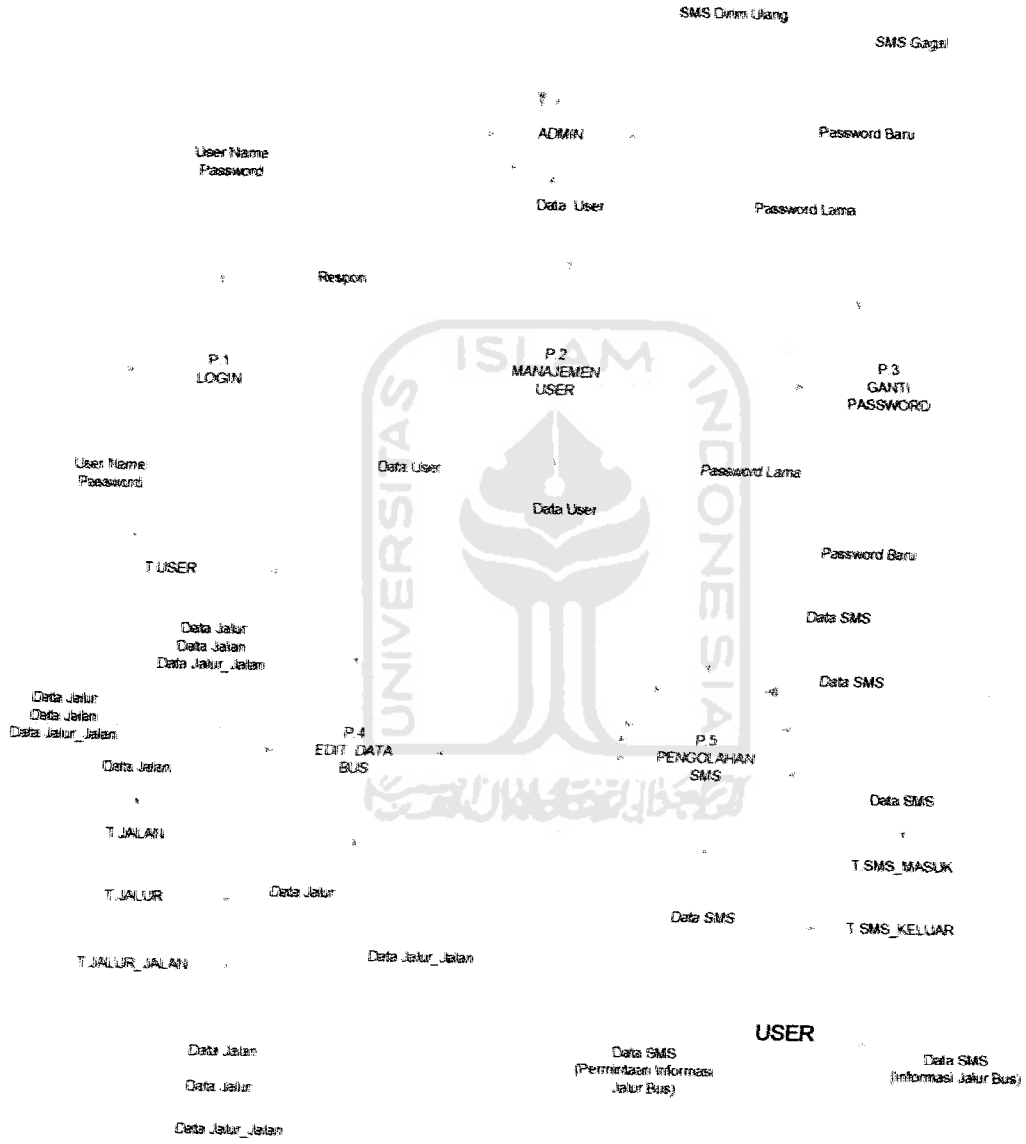


Gambar 4.1 Diagram Konteks Sistem *Natural Query* Jalur Bus Dengan Pengolahan Bahasa Alami.

Diagram konteks mendefinisikan ruang lingkup dan menggambarkan aliran data dengan entitas-entitas di luar sistem yang berhubungan dengan Sistem Informasi Jalur Bus Kota dengan Bahasa Alami, terlihat pada gambar 4.1. Ada 2 entitas luar yang berhubungan dengan sistem ini, yaitu: admin dan *user*. *User* memberikan masukan berupa data informasi jalan yang akan dituju dan posisi *user* berada, kemudian menerima balikan dari *sistem* berupa data atau informasi yang diolah oleh sistem, selanjutnya sistem menyediakan data hasil berupa Informasi jalur Bus Kota yang akan dinaiki oleh *user*.

4.2.1.2 Diagram Arus Data Level 1 (*Data Flow Diagram*)

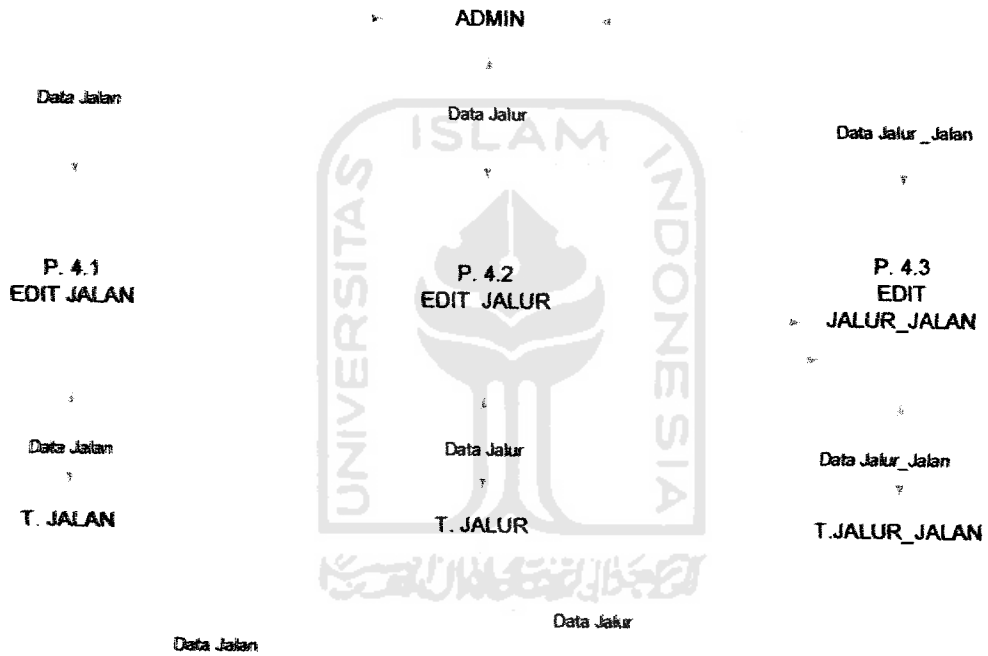
Diagram aliran data menggambarkan sistem sebagai suatu jaringan kerja antara fungsi yang berhubungan dengan aliran data dari sistem.. Aliran data level 1 terlihat pada gambar 4.2 dibawah ini.



Gambar 4.2 DFD Level 1 Sistem *Natural Query* Jalur Bus

Pada proses ini, *user* meminta informasi kepada Sistem Informasi Jalur Bus Kota dengan cara mengirimkan SMS kepada sistem. Kemudian sistem akan me-*reply* berupa jalur bus kota yang akan ditumpangi oleh *user*. Yang oleh admin data-data yang berasal dari Tabel Jalan, Tabel Jalur dan Tabel Jalur_Jalan diolah oleh sistem untuk di-*reply* kepada *user* melalui Kirim SMS.

4.2.1.3 Diagram Arus Data Level 2 Proses Edit Data Bus



Gambar 4.3 DFD Level 2 Proses Edit Data Bus

Pada gambar 4.3 diatas, adalah tentang proses edit data Jalan, Jalur, dan Jalur_Jalan. Dimana proses edit ini, adalah proses identifikasi jalan dan jalur yang dilewati oleh bus yang melewati kota Yogyakarta. Dalam proses Edit Jalan ini, admin memasukkan Id_Jalan, Nama Jalan, dan Kata Kunci. Untuk Kata Kunci (*key word*) disini

merupakan *representative* dari Nama Jalan yang telah dimasukkan, sebagai identifikasi dari sistem ini untuk Nama Jalan, dengan cara meng-*input*-kan variabel-variabel yang berhubungan dengan jalan yang dimaksud.

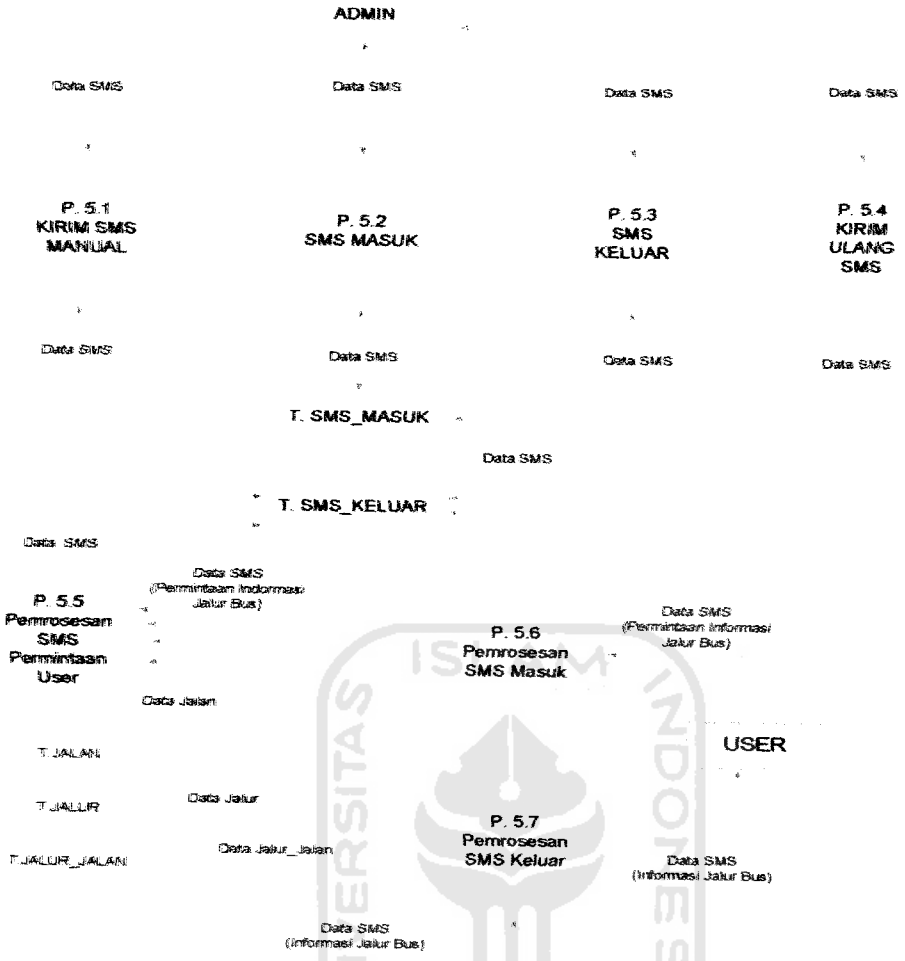
Sedangkan proses Edit Jalur pada sistem ini, admin memasukan data-data Jalur Bus yang melewati kota Yogyakarta.

Untuk proses Edit Jalur_Jalan atau Edit Rute bus ini, admin harus memilih salah satu Jalur Bus yang telah tersedia dari data yang dimasukkan pada saat Edit Jalur. Kemudian admin, harus memasukkan data jalan-jalan yang telah tersedia dari Edit Jalan yang disesuaikan dengan rute jalur bus yang sebenarnya.

4.2.1.4 Diagram Arus Data Level 2 Proses SMS

Pada proses gambar 4.4 yakni *Data Flow Diagram* Level 2 menerangkan proses SMS. Yang terdiri atas proses SMS Masuk, yang berisi *request* dari pengirim SMS mengenai jalur yang akan ditumpangi sesuai dengan posisi awal dan tujuan akhir pengirim SMS.

Kemudian proses SMS Keluar, Kirim SMS Manual serta Kirim Ulang SMS ini, merupakan proses dimana sistem akan merekamnya melalui *data base* yang terdiri atas: Kepada, Pesan, Status, Coba, dan Tanggal Pengiriman. Yang berupa *output* jalur yang dikehendaki dari *user* atau pengirim SMS. Antara *user* (pengirim SMS) dengan sistem terhubung via kabel data dengan Modem GSM yang terkoneksikan dengan SMSC.



Gambar 4.4 DFD Level 2 Proses SMS.

4.2.2 Perancangan Basis Data

Suatu sistem yang terintegrasi dengan basis data harus mempunyai struktur basis data yang teratur dan terorganisasi dengan baik. Basis data merupakan bagian terpenting dari suatu sistem informasi yang dibuat. Basis Data adalah sistem terkomputerisasi yang tujuan utamanya adalah memilihara informasi dan membuat informasi tersedia saat dibutuhkan. Secara Praktis basis data dapat dianggap sebagai suatu penyusunan data yang terstruktur yang disimpan dalam media pengingat (*hard disk*) yang tujuannya adalah agar data tersebut dapat diakses dengan mudah dan cepat. [KAD02]

Struktur ini berisi rancangan basis data dari variable-variabel yang digunakan di dalam pembuatan Sistem *Natural Query* Jalur Bus, adalah sebagai berikut:

Tabel 4.1 Data User ini, berfungsi sebagai data *user* dan *password* untuk hak akses suatu halaman tertentu didalam Sistem *Natural Query* Jalur Bus Kota Dengan Pengolahan Bahasa Alami ini.

Tabel 4.1 Data User

Field	Type	Ukuran	Keterangan
id_user	Char	3	Primary Key
Uname	Varchar	20	
Passwd	Varchar	32	
nm_user	Varchar	30	

Pada tabel 4.2 Data Jalan, berisi nama-nama jalan yang berada di Yogyakarta yang dilewati jalur bus kota serta berisi kata kunci dari nama-nama jalan tersebut.

Tabel 4.2 Data Jalan

Field	Type	Ukuran	Keterangan
id_jalan	Char	5	Primary Key
nm_jalan	Varchar	50	
kata_kunci	Text	225	Kata Kunci

Pada 4.3 Tabel Jalur, terdiri dari dua (2) item yang berisi nama jalur-jalur bus kota yang beroperasi di Yogyakarta serta berisi id_jalur armada yang beroperasi.

Tabel 4.3 Data Jalur

Field	Type	Ukuran	Keterangan
id_jalur	Char	3	Priamary Key
nm_jalur	Varchar	30	

Pada Tabel 4.4 Jalur dan Jalan ini, merupakan perpaduan antara data jalan-jalan dan jalur-jalur atau rute jalur bus yang berada di kota Yogyakarta. Terelasi antar tabel Jalan dan Jalur.

Tabel 4.4 Data Jalur dan Jalan

Field	Type	Ukuran	Keterangan
id_jalur	Char	3	Primary Key
id_jalan	Char	5	Primary Key
No	Tinyint	4	No urut input jalan

Pada tabel 4.5 ini, merupakan tabel yang ber-isi data-data yang akan ditampilkan oleh Sistem melalui pada *form* SMS Keluar.

Tabel 4.5 Data SMS Keluar

Field	Type	Ukuran	Keterangan
id_sms	Bigint	20	Primary Key
Kepada	Varchar	20	
Pesan	Varchar	160	
Status	Char	1	
Coba	Tinyint	4	
Tanggal	Date		
Jam	Time		

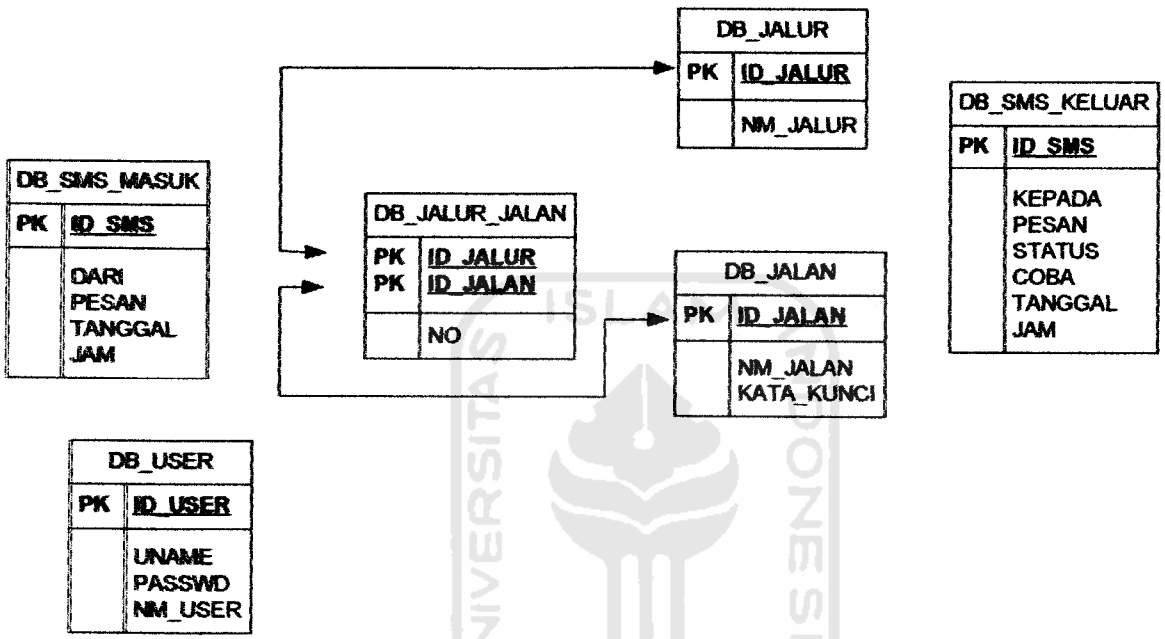
Data pada Tabel 4.6 SMS Masuk ini berisikan informasi apa saja mengenai SMS yang masuk ke dalam sistem. Yang berisikan id_sms, Dari, Pesan, tanggal dan Jam, serta pesan (*request*) yang masuk ke dalam Sistem *Natural Query* Jalur Bus Kota ini.

Tabel 4.6 Data SMS Masuk

Field	Type	Ukuran	Keterangan
id_sms	Bigint	20	Primary Key
Dari	Varchar	20	
Pesan	Varchar	160	
Tanggal	Date		
Jam	Time		

4.2.3 Relasi Antar Tabel

Relasi antar tabel dibuat dengan tujuan membuat suatu hubungan antar *database* sehingga dapat saling memanggil data yang satu dengan yang lainnya agar saling berhubungan. Gambar 4.5 dibawah ini merupakan relasi antar tabel yang ada.



Gambar 4.5 Relasi Antar Tabel

Keterangan :

PK : *Primary Key*

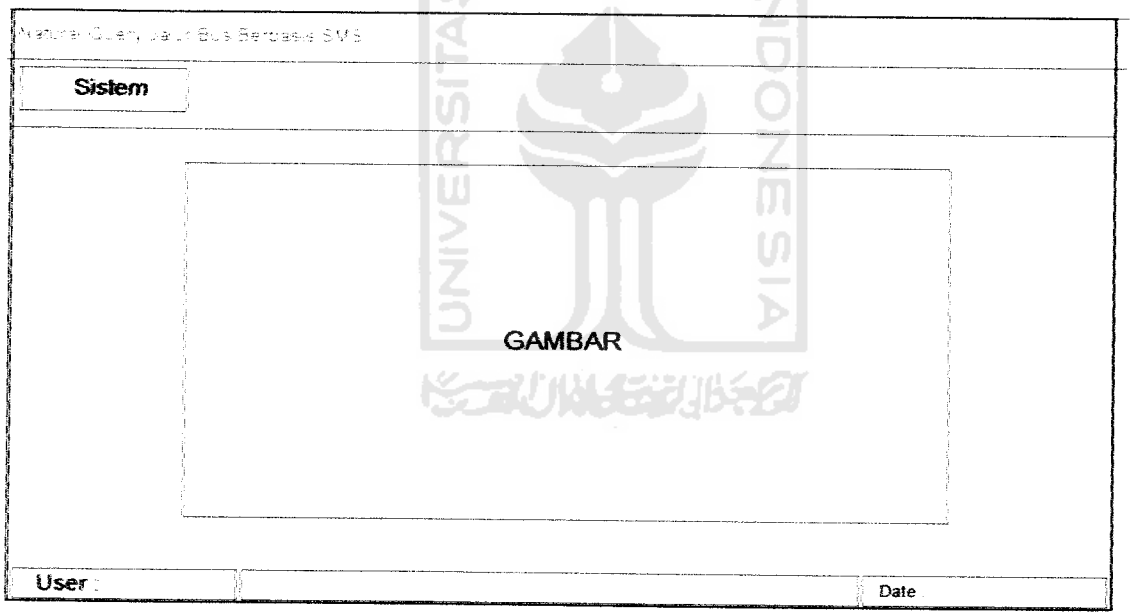
↔ : Relasi *one to one*

↤↦ : Relasi *one to many*

4.2.4 Perancangan Interface

Perancangan antar muka dibuat dengan tujuan agar sistem yang akan dibuat sudah user *friendly* atau belum, terkadang sistem yang dibuat berbasis GUI (*Graphic User Interface*) yang tidak melalui perancangan yang baik dalam aplikasinya akan menyulitkan user untuk mengoperasikannya. Oleh karena itu kebutuhan antar muka dirancang sedemikian rupa guna memudahkan pengguna dapat berinteraksi dengan baik. Tampilan antar muka menu utamanya terdiri atas Sistem, Data Bus, SMS dan Konfigurasi.

Untuk dapat masuk kedalam Menu Utama, *user* harus memasukkan *Username* dan *Password* terlebih dahulu, bila tidak memasukkan *Username* dan *Password*, Sistem hanya akan berisi Login, Informasi dan Tutup saja. Terlihat pada gambar 4.6.



Gambar 4.6 Perancangan *Interface* Menu Utama

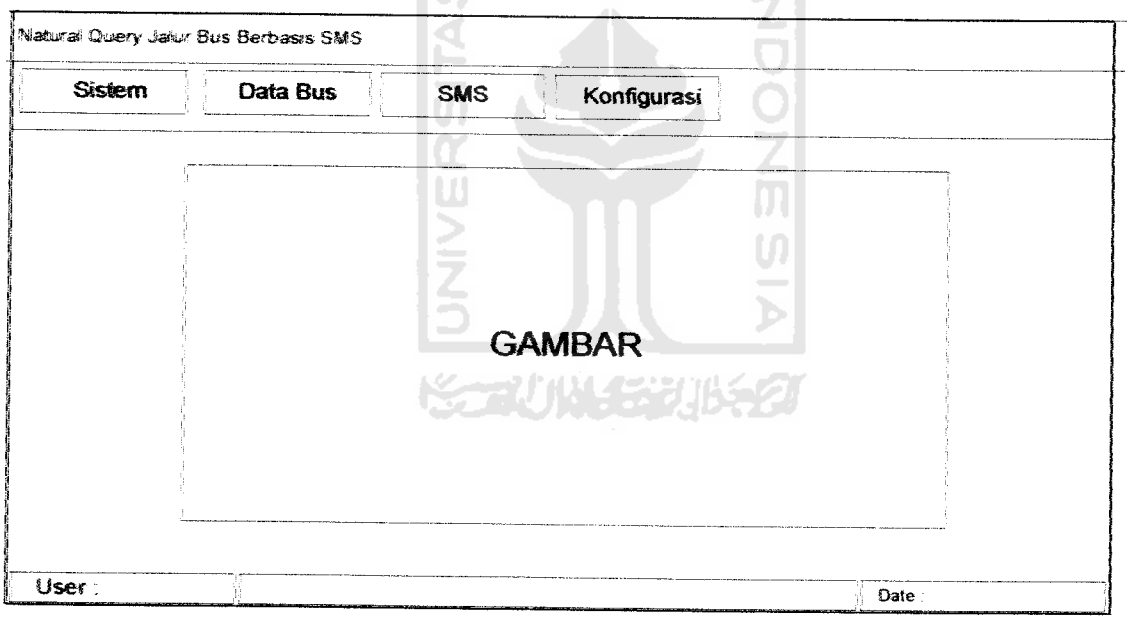
Menu **Sistem** setelah memasukkan *Username* dan *Password* dengan benar, terdiri atas *Login, Logout,* Manajemen *User,* Ganti *Password,* Informasi, dan Tutup.

Sedang untuk menu **Data Bus**, terdiri atas Edit Jalan, Edit Jalur dan Edit Rute (Jalur_Jalan). Yang datanya disesuaikan dengan data riil rute bus yang berlaku di Yogyakarta.

Pada menu **SMS** di tampilan antar mukanya, terdapat 3 sub menu yang terdiri atas: SMS Masuk, SMS Keluar, dan Kirim SMS.

Kemudian untuk menu sistem yang ketiga adalah, menu **Kofigurasi**, dimana menu ini untuk melihat konfigurasi atau setting *database* dari *server* sistem ini.

Kemudian untuk rancangan tampilan *interface* Menu Utama bila telah mengisikan *Username* dan *Password* dengan benar, akan terlihat pada gambar 4.7 dibawah ini:



Gambar 4.7 Perancangan *Interface* Menu Utama Setelah Memasukkan *Username* dan *Password* Dengan Benar.

BAB V

IMPLEMENTASI PERANGKAT LUNAK

5.1 Batasan Masalah

Aplikasi *Natural Query* Jalur Bus ini, dirancang untuk memandu serta memberikan kemudahan bagi *user* atau pengirim SMS, untuk mengakses jalur-jalur bus yang beroperasi di kota Yogyakarta. Sistem tersebut mempunyai batasan implementasi sebagi berikut :

1. Jalur bus yang dapat diakses melalui SMS oleh *user*, hanya yang beroperasi di daerah kota Yogyakrta.
2. *User* (pengirim SMS) harus memberikan informasi mengenai posisi awal dan posisi tujuan dari *user*.
3. Aplikasi dibangun dengan bahasa J2SE sebagai *compiler*nya, dan menggunakan MySQL sebagi *database*-nya.
4. Bahasa yang digunakan dalam sistem ini menggunakan bahasa Indonesia.

5.2 Tahap Pembuatan Perangkat Lunak

Pembuatan aplikasi *Natural Query* Jalur Bus ini melalui tiga tahap, yaitu :

1. Implementasi antarmuka. Pada tahap ini, yang dilakukan adalah merancang antarmuka yang akan digunakan.
2. Tahap pembuatan basis data (*database*) dengan relasinya.
3. Tahap penulisan kode program. Pada tahap ini dilakukan penulisan kode-kode program yang diinputkan pada tiap-tiap tombol yang digunakan dalam tiap proses.

5.3 Perangkat Lunak dan Perangkat Keras yang Dibutuhkan

Perangkat keras (*hard ware*) minimum yang digunakan dalam aplikasi adalah sebagai berikut :

1. CPU : Berbasis prosessor 32-bit
2. RAM : Minimal 256 MB.
3. VGA Card : 32 MB
4. Harddisk dengan ruang kosong minimal 5 GB
5. Monitor
6. Keybord & Mouse
7. CD ROM

Perangkat lunak (*soft ware*) yang digunakan dalam pembuatan aplikasi *Natural Query* Jalur Bus adalah sebagai berikut :

1. Sistem Operasi Windows XP
2. Microsoft Visio 2002
3. Java JDK 1.5.0_06
4. NetBeans 5.0
5. Siemens Data Suite
6. MySQL

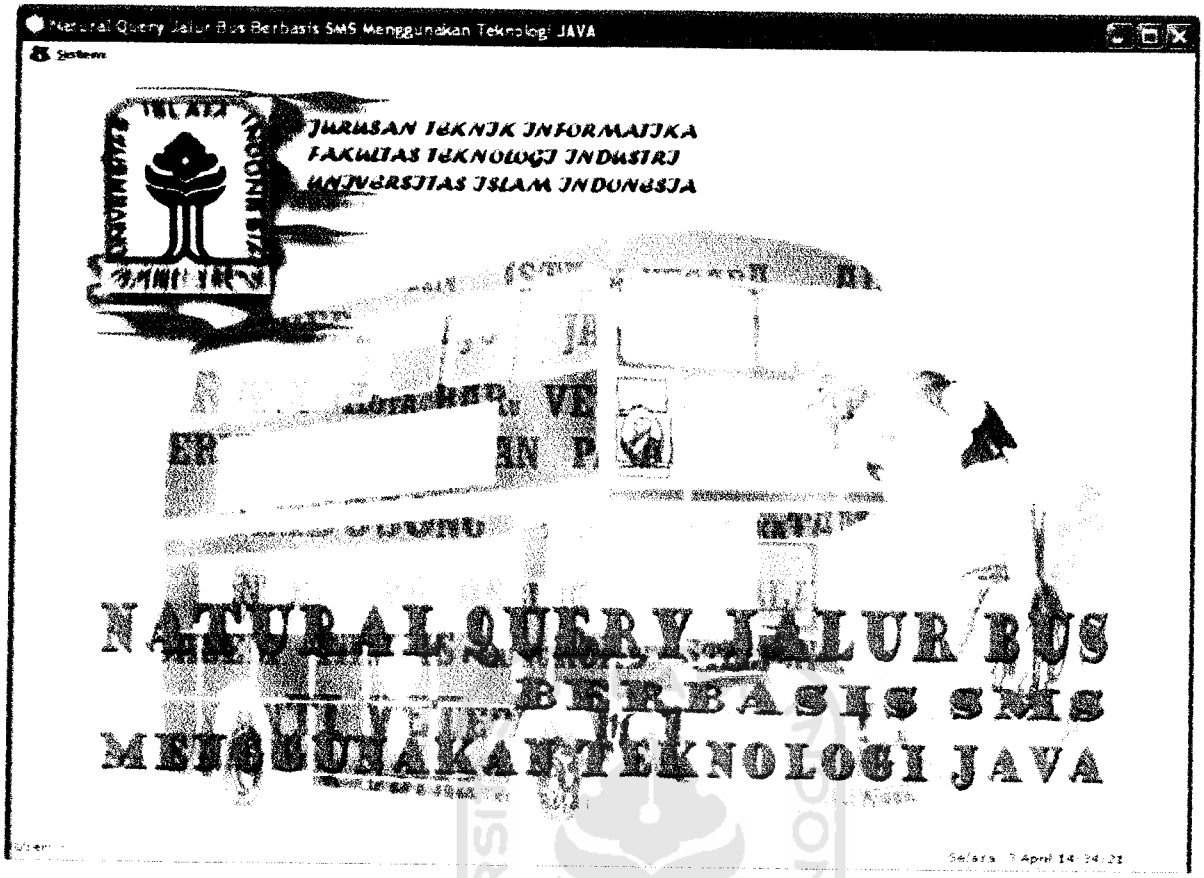
5.4 Implementasi Perangkat Lunak

Implementasi merupakan tahapan dimana sistem siap dioperasikan pada keadaan yang sebenarnya sehingga sistem yang dibuat benar-benar dapat menghasikan tujuan yang diinginkan. Sebelum program diterapkan dan diimplementasikan dalam keadaan sebenarnya dilapangan, maka program harus *error free* (bebas kesalahan). Kesalahan pemrograman yang mungkin terjadi antara lain kesalahan penulisan bahasa, kesalahan sewaktu proses, atau kesalahan logika.

5.4.1 Hasil Antarmuka

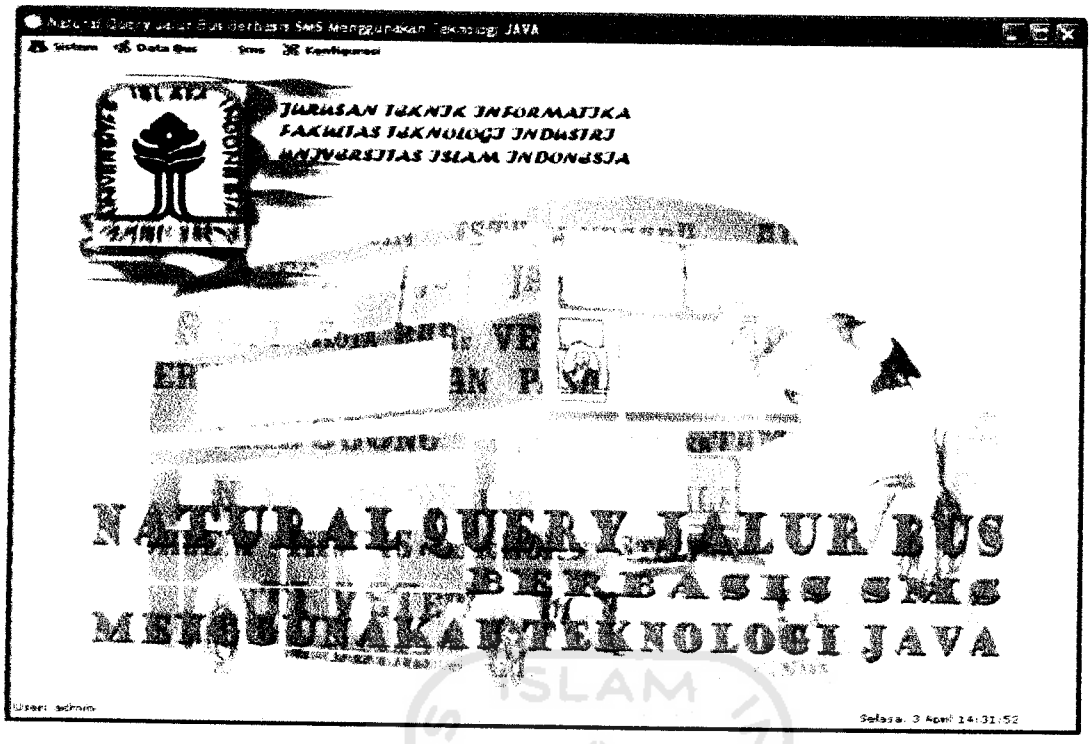
Hasil Implementasi Antarmuka ini, merupakan hasil dari perancangan *interface* yang dibahas pada Bab IV, kemudian hasil dari rancangan tersebut di implementasikan kedalam pembuatan perangkat lunak ini.

Pada tahap awal sistem ini, ketika *user* membuka implementasi, *interface* yang pertama kali dilihat adalah *form* awal. Pada *form* ini terdapat menu Sistem yang berisikan *Login*, dimana *user* dapat mengakses Sistem *Natural Query* ini dengan memasukkan *username* serta *password* yang sesuai. Kemudian pada *form* ini terdapat sub menu Informasi, untuk mengetahui modem yang digunakan, apakah telah tersambung atau belum. Serta sub menu Tutup, untuk menutup (*close*) sistem ini secara keseluruhan. Tampilannya dapat dilihat pada gambar 5.1.



Gambar 5.1 Tampilan Menu Awal

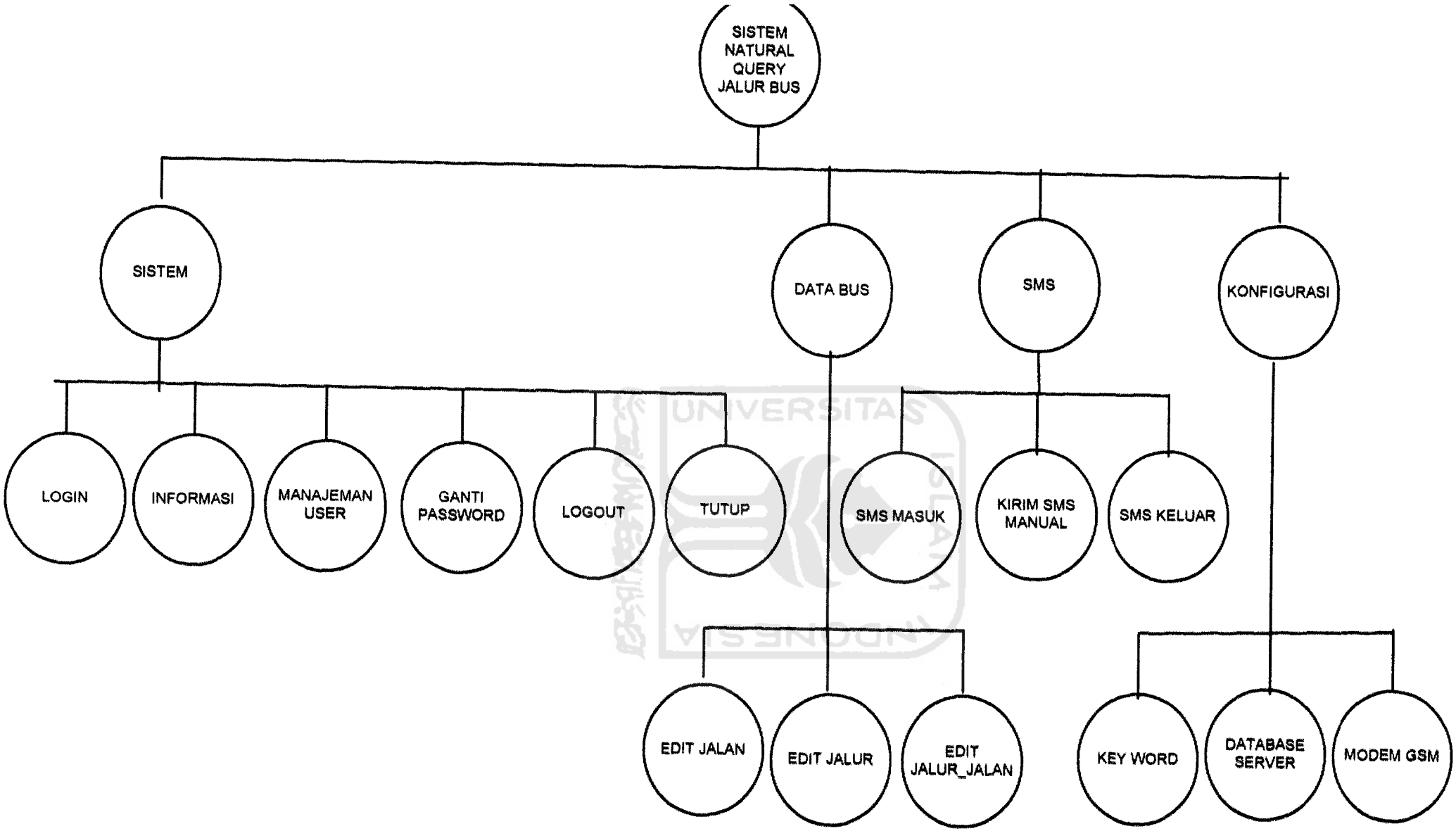
Kemudian hasil perancangan *interface* selanjutnya adalah, *Form* Utama ini adalah *form* yang ditampilkan setelah admin atau operator memasukkan *Username* dan *Password*, dimana pada *form* ini admin dapat meng-edit data atau memasukkan data baru, atau melakukan setting konfigurasi dari sistem ini. Dimana pada *form* utama ini, terdapat menu Sistem, Data Bus, SMS, Konfigurasi. Tampilan *interface Form* Utama dapat kita lihat pada gambar 5.2 di bawah ini.



Gambar 5.2 Tampilan *interface* Menu Utama

5.4.2 Hasil Menu

Pada hasil implementasi ini, menu-menu yang akan diakses dapat terlihat *tool-tool*-nya pada *interface* Menu Utama. Terdapat empat menu utama dalam sistem *Natural Query* Jalur Bus ini, yang didalamnya kemudian terbagi menjadi sub-sub menu lagi. Diantaranya adalah menu: Sistem, Data Bus, SMS, dan Konfigurasi. Untuk gambar struktur dari hasil implementasi menu, dapat dilihat struktur keseluruhannya pada gambar 5.3.



Gambar 5.3 Struktur Implementasi Menu

5.4.2.1 Hasil Sistem

Pada *tool* menu Sistem ini, terdapat 6 buah sub menu, yaitu *Login*, Informasi, Manajemen User, *Logout*, Ganti *Password* dan Tutup. Penjabarannya sebagi berikut:

a. *Login*, *user* harus memasukkan *username* dan *password* bila ingin mengakses sistem ini.

b. Informasi, berisi informasi siapa pembuat program ini dan berisi informasi dari modem yang digunakan.

c. Manajemen *User*, bila sistem ingn menambah *user* baru selain admin.

d. *Logout*, dimana *user* ingin menutup sebagian menu dari sistem, dan kembali ke menu awal.

e. Ganti *Password*, dimana *user* ingin mengganti *password* yang lama dengan *password* yang baru.

f. Tutup, bilamana *user* ingin menutup (*close*) seluruh aplikasi ini, dapat dilakukan dengan sub menu ini.

5.4.2.2 Hasil Data Bus

Pada hasil implementasi ini, terdapat 3 buah sub menu. Dimana ketiganya merupakan saling berhubungan dalam relasi *database*. Dapat diterangkan sebagai berikut:

a. Edit Jalan, *user* dalam hal ini operator ataupun admin dapat melakukan penambahan, *delete* atau *input* data jalan-jalan, beserta *key word*-nya pada sub menu ini.

b. Edit Jalur, sama halnya pada *Edit* Jalan, *user* dapat meng-edit jalur bus pada sistem ini.

c. Edit Rute, berbeda dengan 2 sub menu diatas, pada sub menu *Edit* Rute ini, *user* memilih jalur yang tersedia dan jalan yang tersedia oleh sub menu *Edit* Jalan & Jalur, kemudian disesuaikan dengan data riil.

5.4.2.3 Hasil SMS

Pada Menu SMS ini, terdapat tiga sub menu, yang ketiganya berhubungan dengan *feed back* yang akan diberikan oleh pengirim SMS. Ketiganya yaitu:

a. SMS Masuk, merupakan SMS atau pesan yang diterima oleh sistem ini yang kemudian disimpan dalam *database*, untuk di-*identifier*.
b. Kirim SMS Manual, bila terjadi *error* pada sistem dan tidak bisa mengirimkan SMS secara otomatis, maka admin atau operator dapat mengirim *reply* dengan cara manual pada sub menu ini.
c. SMS Keluar, seluruh SMS yang keluar dari sistem ini akan di-*record* oleh sistem, dan tersimpan dalan *database*, dapat dilihat pada sub menu ini.

5.4.2.4 Hasil Konfigurasi

Menu ini memiliki dua sub menu, dimana sub menu ini untuk melakukan *setup* terhadap sistem *Natural Query* Jalur Bus ini. Keduanya yakni:

a. *Database Server*, dimana sub menu ini untuk melakukan *setting* atau *setup* terhadap *database* yang digunakan oleh sistem.
b. Modem GSM, dimana *setup port modem* ponsel dapat dilakukan pada sub menu ini. Berfungsi untuk mengkoneksikan PC dan ponsel.

c. *Key Word*, merupakan bagian dari sistem ini, untuk melakukan penambahan unsur kata-kata "dari" dan "ke" yang ada kemungkinan dalam SMS masuk diketikkan oleh *user* (pengirim SMS) dalam melakukan *request*, seperti: "dari, dr, dr." Atau "ke, menuju".

5.4.3 Hasil Prosedural

Hasil implementasi ini, merupakan hasil dari pembuatan perangkat lunak dari tahap penulisan kode-kode program (*source code*), dari Sistem *Natural Query* Jalur Bus yang dibangun. Hasil Implementasi Prosedural ini, yang terbagi atas beberapa *package* (paket) yang mempunyai fungsi-fungsi tertentu dalam pembangunan sistem ini.

Didalam *package-package* ini, terbagi lagi atas kelas-kelas (*class*) yang mempunyai fungsi-fungsi berbeda pula. Sistem *Naural Query* Jalur Bus ini memiliki *package-package* (paket-paket) sebagai berikut:

5.4.3.1 Package database

Pada paket *database* ini, terdapat kelas-kelas untuk mengakses *database* yang digunakan oleh sistem ini, yaitu:

1. Connection.class

   Pada *class* ini digunakan untuk mengoneksikan *database*.

2. Query.class

   Untuk menjalankan *query* SQL.

5.4.3.2 Package smsbuskota

Pada *package* smsbuskota ini, merupakan penopang utama dari sistem yang dibangun, dimana seluruh proses implementasi yang telah disebutkan diatas *source code*-nya dapat dilihat disini. Paket ini terbagi atas kelas-kelas sebagi berikut:

1. Database.class

   Pada *class* ini digunakan *setting* koneksi dengan *database*.

2. Informasi.class

   Pada *class* ini, digunakan pada sistem untuk memberikan informasi berupa modem dan *about* dari sistem.

3. Login.class

   Pada *class* ini, difungsikan untuk menampilkan menu *login* pada sistem dimana *user* memasukkan *username* dan *password*.

4. Main.class

   Pada Main.class ini digunakan ketika sistem di-*run*.

5. ManajemenUser.class

   Pada *class* ini, digunakan untuk menambahkan *user-user* baru untuk mengoperasikan sistem *Natural Query* Jalur Bus ini.

6. Password.class

   Pada *class* ini, digunakan oleh *user* untuk mengganti *password* lama menjadi *password* yang baru.

7. Modem.class

   Pada Modem.class digunakan pada sistem untuk melakukan konfigursi pada sistem.

8. SetupJalan.class

   Pada *class* ini, berfungsi sebagai edit pada sistem, pada sub menu Edit Jalan.

9. SetupJalur.class

   Pada *class* ini, berfungsi sebagai edit pada sistem, pada sub menu Edit Jalur.

10. SetupRute.class

    Untuk *class* ini, digunakan untuk melakukan edit pada sub menu Edit Rute.

11. SmsKeluar.class

    Seluruh SMS yang keluar dilakukan pada sistem, *source code*-nya pada *class* ini.

12. SmsMasuk.class

    Sedangkan SMS yang masuk, pemrosesan kedalam sistem diolah pada *class* ini.

13. SmsKirim.class

    Pada *class* ini, untuk melakukan proses kirim SMS secara manual, jika SMS yang dikirim secara otomatis oleh sistem tidak berjalan semestinya.

14. SmsProcesor.class

    Pada SmsProcesor.class ini, *source code*-nya digunakan sebagai proses parsing terhadap SMS yang masuk.

15. Utama.class

    Pada *class* ini, tampilan utama dari sistem *souce code* ada pada class ini.

5.4.3.3 Package utils

Pada package utils ini, terdapat kelas-kelas yang berfungsi sebagai alat bantu (*utilitas*) terhadap sistem *Natural Query* Jalur Bus.

1. Date.class

Pada *class* ini, akan menampilkan data pada sistem berupa: hari, tanggal, dan jam.

2. Dialog.class

Pada kelas Dialog ini, akan menampilkan peringatan atau *message* terhadap *user* bila menjalankan sistem ini.

3. File.class

Pada File.class ini, berfungsi untuk membuka *file-file* yang ada pada sistem.

4. IniFile.class

Kemudian untuk kelas ini, berfungsi sebagai pengaturan konfigurasi dari sistem.

5. Util.class

Pada Util.class ini, digunakan sebagai alat bantu sistem jika dijalankan.

5.4.3.4 Procedure Program

Procedure program dari sistem ini, yang berupa *source code* untuk beberapa prosedur-prosedur utama dari sistem *Natural Query* Jalur Bus, adalah sebagai berikut:

1. Kirim SMS

Method ini terdapat pada *package* smsbuskota di kelas SmsKirim.class. *Source Code*-nya dapat dilihat pada halaman Lampiran A.

2. Proses *Parsing* SMS Keluar.

Method ini terdapat pada *package* smsbuskota di kelas SmsProcesor.class. Sedangkan untuk *source code*-nya dapat kita lihat pada halaman Lampiran B.

BAB VI

ANALISIS KINERJA PERANGKAT LUNAK

6.1 Pengujian Sistem

Tahap selanjutnya adalah tahap pengujian dan analisis perangkat lunak. Sebelum sistem atau program diaplikasikan pada *real activity*, maka diperlukan evaluasi atau pengujian dan pengetesan terhadap berbagai aspek. Pengujian aplikasi ini perlu dilakukan sebelum program tersebut diterapkan ke dalam lingkungan sebenarnya.

Tujuan pengujian sistem antara lain untuk membandingkan kebenaran dan kesesuaian dengan kebutuhan perangkat lunak. Pengujian ini dilakukan untuk menemukan kesalahan-kesalahan yang mungkin terjadi. Pada tahapan ini, perangkat lunak akan diuji apakah masih ditemukan *error* pada sistem yang dibuat.

Caranya dengan mengirimkan data SMS kepada sistem, yang berisikan posisi awal dan posisi tujuan pengirim SMS. Pengujian ini dilakukan terhadap sistem dengan kondisi normal, yaitu apabila posisi awal (dari) dan posisi tujuan (ke) terpenuhi, serta data SMS yang dikirimkan sesuai dengan *database*.

Kemudian pengujian ini dilakukan terhadap sistem dengan kondisi tidak normal, yaitu apabila posisi awal (dari) dan posisi tujuan/akhir (ke) tidak terpenuhi, serta data SMS yang dikirimkan tidak sesuai dengan *database*.

6.2 Hasil Pengujian Sistem

Pada Pengujian Sistem ini, dilakukan terhadap sistem dengan keadaan normal, data SMS yang dikirimkan memenuhi unsur-unsur posisi awal (dari) dan posisi akhir (awal) serta data SMS yang dikirimkan sesuai dengan *database*. Sedangkan untuk pengujian yang tidak normal, adalah data SMS posisi awal dan tujuan tidak benar atau kurang salah satunya kemudian data SMS yang dikirimkan tidak ada atau tidak sesuai dengan *database*.

6.2.1 Pengujian Secara Normal

Pengujian program adalah memasukan data sesuai dengan sistem yang ada. Selain itu, pengujian program memiliki arti bahwa di dalam pengujian ini akan dilakukan sesuai dengan data yang sesungguhnya, yaitu dengan memperhatikan tipe data, panjang karakter, maupun kesesuaian dengan sumber data. Dalam pengujian SMS Masuk ini, yang kemudian akan diproses oleh sistem secara otomatis dengan pembacaan teks SMS yang masuk yang telah terpenuhi unsur-unsur posisi awal (dari) dan posisi akhir (ke) kemudian data SMS sesuai juga dengan *database*.

SMS yang masuk dalam meminta *request* jalur bus yang terpenuhi unsur-unsur posisi awal (dari) dan posisi awal (ke), serta data SMS yang masuk sesuai dengan *database*. Pengujian dilakukan dengan cara antara lain:

1. *Request* data SMS yang dikirimkan memenuhi unsur posisi awal (dari) dan posisi akhir (ke), dengan megetikkan posisi awal dahulu, kemudian posisi tujuan. Contoh SMS sesuai keterangan diatas adalah sebagai berikut:

   "Saya dari kusumanegara mau ke Jl. Cokroaminoto naik jalur mana ya boss?? "

2. *Request* data SMS yang dikirimkan ke sistem memenuhi unsur posisi awal (dari) dan posisi tujuan, dengan mengetikkan posisi tujuan dahulu (ke), kemudian baru megetikkan posisi awalnya (dari). Contoh format SMS dari ketentuan diatas dapat dilihat pada kalimat dibawah ini:

   "coy.. aku mau ke Jl. malioboro dari jl. Jenderal Sudirman enaknya lewat mana yah..."

3. Mengirimkan *request* berupa SMS ke sistem, dengan megetikkan nama jalan atau *key word*-nya saja, dengan asumsi suku kata pertama sebagai posisi awal, dan suku kata kedua sebagai posisi tujuan yang dibedakan dengan tanda spasi diantaranya. Dengan catatan posisi yang diketikkan (awal dan tujuan) hanya terdiri atas satu suku kata untuk masing-masing posisi. Contoh format *request* SMS yang dimaksud adalah sebagai berikut:

   " Malioboro Gembiraloka"

6.2.2 Pengujian Secara Tidak Normal

Dalam Pengujian sistem ini, apabila SMS yang masuk dalam pengiriman SMS tidak mencantumkan unsur posisi awal atau posisi akhir hanya salah satunya. Berbeda apabila tidak mencantumkan posisi awal dan tujuan, sistem dapat membacanya yang diketikan pertama sebagai posisi awal dan yang kemudian sebagai posisi tujuan. Serta apabila *user* (Pengirim SMS) salah mengetikkan posisi awal atau tujuan serta pengetikkan *key word* yang tidak sesuai dengan *database*. Pengujian dilakukan dengan cara antara lain:

1. *Request* data SMS yang dikirimkan ke sistem tidak memenuhi unsur posisi awal dan posisi akhir (tujuan), dengan hanya mengetikkan unsur "dari" atau "ke" saja. Contoh *request* SMS yang tidak memenuhi unsur-unsur yang dimaksud adalah sebagai berikut:

   "Saya dari Giwangan mau UGM enaknya naik jalur apa yuk??"

2. Mengirimkan SMS dalam me-*request* jalur bus yang dinginkannya, dengan mengetikkan nama jalan ataupun *key word*, secara salah dalam penulisannya. Contoh pengiriman SMS *request* yang dimaksud adalah sebagai berikut:

   " Saya dari Gedong Tuning mau ke Jl. Taman Siwa "

3. Me-*request* terhadap sistem, hanya mengetikkan nama jalan atau *key word*-nya saja, akan tetapi lebih dari dua suku kata untuk posisi awal atau akhirnya.

   "Katamso Pasar Gamping"

6.2.3 Pengujian Data Jalan Tidak Dilewati Rute Jalur Bus

Pada pengujian SMS yang masuk, jika jalan atau *key word*-nya (posisi awal dan posisi tujuan) ada datanya pada *database*. Akan tetapi jalan atau *key word*-nya yang diketikkan oleh pengirim SMS tidak ada jalur yang melewatinya yang sesuai dengan *database*. Pengujian dilakukan dengan cara sebagai berikut:

1. Mengirimkan data SMS ke sistem, berupa *request* jalur bus dengan mengetikkan posisi awal serta tujuan secara benar posisi awal dan akhirnya, akan tetapi posisi

yang di-*request* tidak dilewati jalur bus. Contoh format SMS *request* yang dimaksud diatas adalah sebagi berikut:

"Saya dari Jl. Veteran mau ke Jl. Suryotomo"

2. Mengirimkan data SMS ke sistem, berupa posisi awal dan tujuan secara tidak benar, akan tetapi tidak dilewati jalur bus. Contoh format SMS *request* yang dimaksud adalah sebagi berikut:

"saya FE-UTY mau ke Melia Purosani enaknya naik jalur berapa neh..."

6.3 Analisis Hasil Pengujian

Dalam analisis hasil pengujian sistem *Natural Query* Jalur Bus ini, yang dilakukan secara normal dan tidak normal, serta data yang tidak tersedia pada database, memberikan hasil *output* sesuai dengan data yang sesungguhnya, yaitu dengan memperhatikan tipe data, panjang karakter, maupun kesesuaian dengan sumber data.

6.3.1 Analisis Hasil Pengujian Secara Normal

1. Untuk *request* data SMS yang dikirimkan memenuhi unsur posisi awal (dari) dan posisi akhir (ke), dengan megetikkan posisi awal dahulu, kemudian posisi tujuan, akan mengeluarkan hasil berupa *alternative* jalur yang harus ditumpangi oleh pengirim SMS dengan menyebutkan jalan yang dilewati jalur yang bersangkutan, dari posisi awal sampai posisi tujuan yang diketikkan. Contoh hasil *reply* dari SMS yang masuk sesuai dengan pengujian secara normal adalah sebagi berikut:

"Rute Bus dari Kusumanegra ke Cokroaminoto **JALUR 12** dengan rute Kusumanegra, Sultan Agung, KHA Ahmad Dahlan, Cokroaminoto, **JALUR 17** dengan rute Kusumanegara, Veteran, Gambiran, Perintis Kemerdekaan, Pramuka, Imogiri Timur, TERMINAL GIWANGAN, Imogiri Timur, Pramuka, Menteri Supeno, Veteran, Kusumanegara, Tamansiswa, Kolonel Sugiono, Letjend. Sutoyo, Mayjend. MT. Haryono, Wahid Hasyim, Letjend. S. Parman, Kapten Tendean, Cokroaminoto"

2. Untuk *request* data SMS yang dikirimkan ke sistem memenuhi unsur posisi awal (dari) dan posisi tujuan, dengan mengetikkan posisi tujuan dahulu (ke), kemudian baru megetikkan posisi awalnya (dari), dengan metode ini, analisa hasil pengujian megeluarkan *output* yang sama. Ini menandakan posisi awal dan akhir jika terpenuhi unsur "ke" dan "dari" tidak menjadi masalah. Contoh hasil *request* SMS masuk sesuai dengan pengujian secara normal sesuai dengan format *request* pengujian secara normal diatas, adalah sebagai berikut:

"Rute bus dari jl. Jenderal Sudirman ke malioboro JALUR 4 dengan rute Jenderal Sudirman, Pangeran Mangkubumi, Malioboro"

3. Pada pengiriman *request* berupa SMS ke sistem, dengan megetikkan nama jalan atau *key word*-nya saja, dengan asumsi suku kata pertama sebagai posisi awal, dan suku kata kedua sebagai posisi tujuan yang dibedakan dengan tanda spasi

diantaranya. Dengan catatan posisi yang diketikkan (awal dan tujuan) hanya terdiri atas satu suku kata untuk masing-masing posisi, dengan memenuhi persyaratan diatas, maka sistem memberikan *reply* berupa jalur bus yang tersedia sesuai dengan *database*. Contoh balasan dari *request* hasil pengujian secara normal sesuai dengan ketentuan diatas, dapat dilihat contoh format *reply*-nya:

"Rute bus dari malioboro ke Gembiraloka (Kusumanegara) JALUR 4 dengan rute Malioboro, Ahmad Yani, Senopati, Sultan Agung, Kusumanegara"

6.3.2 Analisis Hasil Pengujian Secara Tidak Normal

1. *Request* data SMS yang dikirimkan ke sistem tidak memenuhi unsur posisi awal dan posisi akhir (tujuan), dengan hanya mengetikkan unsur "dari" atau "ke" saja.
2. Mengirimkan SMS dalam me-*request* jalur bus yang dinginkannya, dengan mengetikkan nama jalan ataupun *key word*, secara salah dalam penulisannya.
3. Me-*request* terhadap sistem, hanya mengetikkan nama jalan atau *key word*-nya saja, akan tetapi lebih dari dua suku kata untuk posisi awal atau akhirnya.

Untuk analisa hasil pengujian untuk ketiga item diatas, memiliki respon *output* yang sama. Dengan memberikan *output* berupa *request* data SMS yang masuk sesuai ketiga item diatas, tidak dapat diproses karena tidak memenuhi unsur "ke" dan "dari", serta tidak sesuai dengan *database* sistem. Berupa SMS balasan dari sistem yang berupa kalimat sebagai berikut:

"Request tidak dapat diproses, dari dan tujuan tidak diketemukan. Contoh Mirota Malioboro, dari gembiraloka ke ugm, ke Ramai mall dari Jl. Kaliurang".

6.3.3 Analisis Hasil Pengujian Data Jalan Tidak Dilewati Jalur Bus

1. Dengan mengirimkan data SMS ke sistem, berupa *request* jalur bus dengan mengetikkan posisi awal serta tujuan secara benar posisi awal dan akhirnya, akan tetapi posisi yang di-*request* tidak dilewati jalur bus, maka sistem akan memberikan *report* berupa *reply* yang menyatakan jalan yang dimaksud *user* (pengirim SMS) tidak tersedia pada *database*. Contoh hasil *reply* dari *request* jalur bus, tetapi jalan atau *key word* posisi awal dan tujuan yang diketikkan tidak dilewati jalur bus, sesuai dengan hasil pengujian diatas, adalah sebagai berikut:

   "Rute bus dari Veteran ke Mayor Suryotomo
   Tidak ada di database."

2. Dengan mengirimkan data SMS ke sistem, berupa posisi awal dan tujuan secara tidak benar, akan tetapi tidak dilewati jalur bus, sesuai hasil analisa maka sistem akan mengeluarkan *output* berupa *reply* yang menyatakan SMS yang masuk tidak dapat diproses, karena tidak memenuhi posisi awal dan akhir serta tidak sesuai dengan *database* sistem, seperti pada SMS balasan diatas. Sedangkan contoh dari pengujian *request* sesuai pengujian diatas akan menghasilkan *reply* yang sama dengan pengujian secara tidak normal.

6.3.4 Kelebihan Sistem

Sistem *Natural Query* Jalur Bus ini, memiliki kelebihan. Diantaranya adalah *user* (pengirim SMS) dapat mengirimkan SMS kepada sistem dalam memandu mendapatkan jalur bus yang akan ditumpangi dengan mengirimkan SMS berbahasa alami. Seperti halnya berkomunikasi dengan sesama manusia *(human)*, karena sistem ini berteknologi *Natural Language Processing*.

Pengguna sistem ini, akan mendapatkan kemudahan serta efisensi waktu dan yang terpenting adalah murah dalam mengakses sistem dalam memandu jalur bus yang akan ditumpangi, dengan cara mengirim SMS ke sistem.

Selain itu, sistem ini memiliki potensi untuk lebih banyak dikembangkan dan dipakai oleh instansi didaerah manapun, khususnya seperti Dinas Perhubungan yang memiliki kompetensi kearah transportasi.

6.3.5 Kekurangan Sistem

Sesuatu yang ada didunia ini pasti tidak ada yang sempurna, kesempurnaan hanya milik Allah semata. Demikian juga sistem yang dibuat oleh manusia ini.

Sistem *Natural Query* Jalur Bus ini, memilik kekurangan salah satunya adalah memerlukan kebutuhan perangkat keras seperti *hard ware* yang lebih besar kapasitasnya.

Serta sistem belum dapat meng-*cover* serta memberikan solusi terhadap pengirim SMS, bila memunculkan jalan-jalan yang tidak dilalui jalur bus pada saat mengirimkan SMS ke sistem ini.

BAB VII

PENUTUP

7.1 Kesimpulan

Berdasarkan hasil proses pengembangan perangkat lunak yang telah dibuat, baik pada tahap analisis kebutuhan perangkat lunak, perancangan, implementasi dan terutama pada analisis kinerja perangkat lunak, maka dapat diambil beberapa kesimpulan sebagai berikut :

1. Sistem *Natural Query* Jalur Bus ini, mampu memberikan rekomendasi berupa *alternative* jalur yang akan ditumpangi *user* (pengirim SMS) dengan mengirimkan posisi awal dan posisi tujuan terhadap sistem.
2. Dalam meminta *request* jalur bus ini, si pengirim SMS dapat mengirimkan SMS seperti halnya berkomunikasi dengan manusia (*human*), dengan memperhatikan elemen posisi awal dan tujuan.
3. Sistem ini menggunakan teknologi *Natural Language Processing*, dalam mengolah SMS yang masuk dan memberikan *ouput* berupa informasi jalur bus.

7.2 Saran

Beberapa saran yang dapat diberikan untuk pengembangan sistem *Natural Query* Jalur Bus ini mungkin untuk dapat disempurnakan serta bermanfaat bagi masyrakat luas. Adapun saran yang dapat diberikan antara lain :

1. Sistem diharapkan dapat dikembangkan ke arah yang lebih luas dan bersifat umum sehingga dapat diimplementasikan sebagai layanan informasi tidak hanya pada satu alat transportasi saja, dalam hal ini bus.

2. Sistem ini dapat dikembangkan tidak hanya menggunakan bahasa Indonesia, tetapi seluruh bahasa yang ada, dengan memasukkan *key word* sesuai dengan bahasa yang digunakan oleh *user* (pengirim SMS).
3. Untuk lebih mengembangkan sistem ini diharapkan untuk diadakan kerjasama dengan pihak-pihak yang mungkin terkait terutama dengan pihak pemerintah daerah serta Dinas Perhubungan.

DAFTAR PUSTAKA

[ATC02] AT Commands Interface Guide

[BAM04] Bambang, Hariyanto, Ir., MT. *Teori Bahasa, Otomata, dan Komputasi Serta Terapannya.* Bandung: Informatika Bandung, 2004.

[ERW05] Erwin, Muhammad A.H. *Diktat Kuliah Pengolahan Bahasa Alami (version 2.0).* Yogyakarta: 2005.

[GUN03] Gunawan, Ferry. *Membuat Aplikasi SMS Gateway Server dan Client dengan Java dan PHP.* Jakarta: Elex Media Computindo, 2003.

[HAR03] Hariyanto, Bambang. *Esensi-esensi Bahasa Pemrograman Java.* Bandung: Informatika Bandung, 2003.

[KAD02] Kadir, Abdul. *Penuntun Prakktis Belajar SQL.* Yogyakarta : Andi Offset, 2002.

[KAD04] Kadir, Abdul. *Dasar Pemrograman JAVA 2.* Yogyakarta: CV.Andi Offset, 2004.

[KAD03] Kadir, Abdul. *Aplikasi Database Dengan Menggunaka mySQL.* Yogyakarta : Andi Offset, 2003.

[RIC02] Ricyanto, Isak. *Dasar-Dasar Pemrograman Bebasis Objek dengan JAVA 2 (JDK 1,4).* Yogyakarta : Andi Offset, 2002.

[SRI07] Sri G. Hartati, Herry B. Suharto, Soesilo M. Wijono. *Pemrograman GUI Swing Java dengan NetBeans 5.* Yogyakarta: Andi Offset. 2007.

[UDI01] Udirarartatmo, Firrar. *Teori Bahasa dan Otomata.* Yogyakarta : J&J Learning, 2001.

[WAH05] Wahana Komputer, Tim Penelitian dan Pengembangan. *Pengembangan Aplikasi Sistem Informasi Akademik Berbasisi SMS Dengan JAVA.* Jakarta: Salemba Infotek, 2005.

www.elektroindonesia.com

www.ilmukomputer.com

www.nlp.aia.bppt.go.id/glosti/

LAMPIRAN

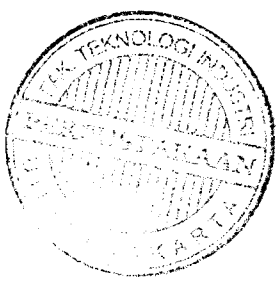
Lampiran A.

Method Proses Kirim SMS.

Terdapat Pada Package smsbuskota, pada kelas SmsKirim.

```
private void cKirimActionPerformed(java.awt.event.ActionEvent evt) {
        if (cekForm()) {
            String pesan = tPesan.getText();
            double jml_char = pesan.length();
            double x = 160;
            int jml_sms = (int) Math.ceil(jml_char/x);
            if (jml_sms == 0) {
                jml_sms = 1;
            }
            for (int i=0; i < jml_sms; i++) {
                String psn = "";
                if (pesan.length() > x) {
                    psn = pesan.substring(0, (int) x);
                    pesan = pesan.substring((int) x);
                } else {
                    psn = pesan;
                }
                m.Q1.exec("INSERT INTO sms_keluar VALUES ('', " +
                          "'"+tKepada.getText()+"', "+
                          "'"+m.util.quote(psn)+"', "+
                          "'A', '0', CURRENT_DATE, CURRENT_TIME)");
            }
            tutup();
        }
    }
```

UNIVERSITAS ISLAM INDONESIA



Lampiran B.

Method Proses Parsing.

Terdapat Pada Package smsbuskota, pada kelas SmsProcesor.

```
public void proses(String no, String text) {
        String balasan = "";
        Vector dari = new Vector();
        Vector ke = new Vector();
        String[] array = m.util.explode(text, " ");
        if (array.length == 2) {
            Q1.select("SELECT * FROM jalan WHERE nm_jalan='"+m.util.quote(array[0])+"' OR
kata_kunci LIKE '%"+array[0]+"%'");
            if (!Q1.isEmpty()) {
                try {
                    dari.add(Q1.SQL().getString("id_jalan"));
                } catch (SQLException ex) {
                    ex.printStackTrace();
                }
            }
            Q1.select("SELECT * FROM jalan WHERE nm_jalan='"+m.util.quote(array[1])+"' OR
kata_kunci LIKE '%"+array[1]+"%'");
            if (!Q1.isEmpty()) {
                try {
                    ke.add(Q1.SQL().getString("id_jalan"));
                } catch (SQLException ex) {
                    ex.printStackTrace();
                }
            }
        } else {
            Vector v = new Vector();
            Q1.select("SELECT * FROM jalan");
            try {
                Q1.SQL().beforeFirst();
                while (Q1.SQL().next()) {
                    int x =
text.toLowerCase().indexOf(Q1.SQL().getString("nm_jalan").toLowerCase());
                    if (x != -1) {
                        if (!inVector(v, Q1.SQL().getString("id_jalan"))) {
                            v.add(new String[] {
                                Q1.SQL().getString("id_jalan"),
                                ""+x,
                                Q1.SQL().getString("nm_jalan")
                            });
                        }
                    } else {
",");                        String[] s = m.util.explode(Q1.SQL().getString("kata_kunci"),

                        for (int i=0; i < s.length; i++) {
                            if (!s[i].trim().equals("")) {
                                x = text.toLowerCase().indexOf(s[i].toLowerCase());
                                if (x != -1) {
                                    if (!inVector(v, Q1.SQL().getString("id_jalan"))) {
                                        v.add(new String[] {
                                            Q1.SQL().getString("id_jalan"),
                                            ""+x,
                                            s[i]
                                        });
                                        break;
                                    }
                                }
                            }
                        }
                    }
                }
            } catch (SQLException ex) {
                ex.printStackTrace();
            }
```

```
                    // urutkan
                    for (int i=0; i < v.size()-1; i++) {
                        for (int j=i+1; j < v.size(); j++) {
                            String[] s1 = (String[]) v.elementAt(i);
                            String[] s2 = (String[]) v.elementAt(j);
                            if (Integer.parseInt(s2[1]) < Integer.parseInt(s1[1])) {
                                v.set(i, s2);
                                v.set(j, s1);
                            }
                        }
                    }
                    boolean ada_dari = false;
                    boolean ada_ke = false;
                    int index_dari = 0;
                    int index_ke = 0;
                    for (int i=0; i < m.vDari.size(); i++) {
                        int x =
text.toLowerCase().indexOf(m.vDari.elementAt(i).toString().toLowerCase());
                        if (x != -1) {
                            ada_dari = true;
                            index_dari = x;
                            break;
                        }
                    }

                    for (int i=0; i < m.vKe.size(); i++) {
                        int x =
text.toLowerCase().indexOf(m.vKe.elementAt(i).toString().toLowerCase());
                        if (x != -1) {
                            ada_ke = true;
                            index_ke = x;
                            break;
                        }
                    }
                    if (ada_dari && ada_ke) {
                        if (index_ke > index_dari) {
                            String s1 = text.substring(0, index_ke);
                            for (int i=0; i < v.size(); i++) {
                                String[] s = (String[]) v.elementAt(i);
                                Q1.select("SELECT * FROM jalan WHERE id_jalan='"+s[0]+"'");
                                try {
                                    if
(s1.toLowerCase().indexOf(Q1.SQL().getString("nm_jalan").toLowerCase()) != -1) {
                                        dari.add(new String[] {s[0],
Q1.SQL().getString("nm_jalan")});
                                    } else {
                                        String[] arr =
m.util.explode(Q1.SQL().getString("kata_kunci"), ",");
                                        for (int j=0; j < arr.length; j++) {
                                            if (!arr[j].trim().equals("")) {
                                                if
(s1.toLowerCase().indexOf(arr[j].toLowerCase()) != -1) {
                                                    dari.add(new String[] {s[0], arr[j]});
                                                    break;
                                                }
                                            }
                                        }
                                    }
                                } catch (SQLException ex) {
                                    ex.printStackTrace();
                                }
                            }
                            String s2 = text.substring(index_ke);
                            for (int i=0; i < v.size(); i++) {
                                String[] s = (String[]) v.elementAt(i);
                                Q1.select("SELECT * FROM jalan WHERE id_jalan='"+s[0]+"'");
                                try {
                                    if
(s2.toLowerCase().indexOf(Q1.SQL().getString("nm_jalan").toLowerCase()) != -1) {
                                        ke.add(new String[] {s[0],
Q1.SQL().getString("nm_jalan")});
```

```
                                } else {
                                    String[] arr =
m.util.explode(Q1.SQL().getString("kata_kunci"), ",");
                                    for (int j=0; j < arr.length; j++) {
                                        if (!arr[j].trim().equals("")) {
                                            if
(s2.toLowerCase().indexOf(arr[j].toLowerCase()) != -1) {
                                                ke.add(new String[] {s[0], arr[j]});
                                                break;
                                            }
                                        }
                                    }
                                }
                            } catch (SQLException ex) {
                                ex.printStackTrace();
                            }
                        }
                    } else {
                        String s1 = text.substring(0, index_dari);
                        for (int i=0; i < v.size(); i++) {
                            String[] s = (String[]) v.elementAt(i);
                            Q1.select("SELECT * FROM jalan WHERE id_jalan='"+s[0]+"'");
                            try {
                                if
(s1.toLowerCase().indexOf(Q1.SQL().getString("nm_jalan").toLowerCase()) != -1) {
                                    ke.add(new String[] {s[0],
Q1.SQL().getString("nm_jalan")});
                                } else {
                                    String[] arr =
m.util.explode(Q1.SQL().getString("kata_kunci"), ",");
                                    for (int j=0; j < arr.length; j++) {
                                        if (!arr[j].trim().equals("")) {
                                            if
(s1.toLowerCase().indexOf(arr[j].toLowerCase()) != -1) {
                                                ke.add(new String[] {s[0], arr[j]});
                                                break;
                                            }
                                        }
                                    }
                                }
                            } catch (SQLException ex) {
                                ex.printStackTrace();
                            }
                        }
                        String s2 = text.substring(index_dari);
                        for (int i=0; i < v.size(); i++) {
                            String[] s = (String[]) v.elementAt(i);
                            Q1.select("SELECT * FROM jalan WHERE id_jalan='"+s[0]+"'");
                            try {
                                if
(s2.toLowerCase().indexOf(Q1.SQL().getString("nm_jalan").toLowerCase()) != -1) {
                                    dari.add(new String[] {s[0],
Q1.SQL().getString("nm_jalan")});
                                } else {
                                    String[] arr =
m.util.explode(Q1.SQL().getString("kata_kunci"), ",");
                                    for (int j=0; j < arr.length; j++) {
                                        if (!arr[j].trim().equals("")) {
                                            if
(s2.toLowerCase().indexOf(arr[j].toLowerCase()) != -1) {
                                                dari.add(new String[] {s[0], arr[j]});
                                                break;
                                            }
                                        }
                                    }
                                }
                            } catch (SQLException ex) {
                                ex.printStackTrace();
                            }
                        }
                    }
                }
```

```
                }
                if (!ada_dari && !ada_ke) {
                    if (v.size() > 0) {
                        String[] s = (String[]) v.elementAt(0);
                        dari.add(new String[] {s[0], s[2]});
                    }
                    if (v.size() > 1) {
                        String[] s = (String[]) v.elementAt(1);
                        ke.add(new String[] {s[0], s[2]});
                    }
                }
            }
            if (dari.size() == 0 || ke.size() == 0) {
                balasan = "Request tidak dapat diproses, dari dan tujuan tidak ditemukan.
    Contoh: Mirota Malioboro, Dari gembiraloka ke ugm, ke ramai mall dari jl. kaliurang";
            } else {
                balasan = "";
                Q1.select("SELECT * FROM jalur");
                try {
                    Q1.SQL().beforeFirst();
                    Vector vJalur = new Vector();
                    while (Q1.SQL().next()) {
                        Q2.select("SELECT * FROM jalur_jalan a " +
                                "LEFT JOIN jalan b ON a.id_jalan=b.id_jalan " +
                                "WHERE id_jalur='"+Q1.SQL().getString("id_jalur")+"' ORDER BY
no");
                        Vector vJalan = new Vector();
                        Q2.SQL().beforeFirst();
                        while (Q2.SQL().next()) {
                            vJalan.add(new String[] {Q2.SQL().getString("id_jalan"),
Q2.SQL().getString("nm_jalan")});
                        }
                        for (int i=0; i < dari.size(); i++) {
                            String[] arr1 = (String[]) dari.elementAt(i);
                            int x = indexOf(vJalan, arr1[0]);
                            if (x != -1) {
                                for (int j=0; j < ke.size(); j++) {
                                    String[] arr2 = (String[]) ke.elementAt(j);
                                    int y = indexOf(vJalan, arr2[0], x);
                                    if (y != -1) {
                                        Vector vHasil = new Vector();
                                        for (int k=x; k <= y; k++) {
                                            String[] s = (String[]) vJalan.elementAt(k);
                                            vHasil.add(s[1]);
                                        }
                                        vJalur.add(" Jalur "+Q1.SQL().getString("nm_jalur")+"
dengan rute "+m.util.implode(vHasil, ", "));
                                    }
                                }
                            }
                        }
                    }
                    if (vJalur.size() > 0) {
                        String[] arr1 = (String[]) dari.elementAt(0);
                        Q3.select("SELECT * FROM jalan WHERE id_jalan='"+arr1[0]+"'");
                        balasan += "Rute bis dari
"+(arr1[1].equals(Q3.SQL().getString("nm_jalan")) ? arr1[1] : arr1[1]+"
("+Q3.SQL().getString("nm_jalan")+")");
                        String[] arr2 = (String[]) ke.elementAt(0);
                        Q3.select("SELECT * FROM jalan WHERE id_jalan='"+arr2[0]+"'");
                        balasan += " ke "+(arr2[1].equals(Q3.SQL().getString("nm_jalan")) ?
arr2[1] : arr2[1]+" ("+Q3.SQL().getString("nm_jalan")+")");
                        balasan += " "+m.util.implode(vJalur, ", ");
                    }
                } catch (SQLException ex) {
                    ex.printStackTrace();
                }
                if (balasan.equals("")) {
                    try {
                        String[] arr1 = (String[]) dari.elementAt(0);
                        Q3.select("SELECT * FROM jalan WHERE id_jalan='"+arr1[0]+"'");
```

```
                    balasan += "Rute bis dari "+Q3.SQL().getString("nm_jalan");
                    String[] arr2 = (String[]) ke.elementAt(0);
                    Q3.select("SELECT * FROM jalan WHERE id_jalan='"+arr2[0]+"'");
                    balasan += " ke "+Q3.SQL().getString("nm_jalan");
                    balasan += " tidak ada di database.";
                } catch (SQLException ex) {
                    ex.printStackTrace();
                }
            }
        }
//          System.out.println(
//                  "Dari: "+dari.size()+"\n" +
//                  "Ke: "+ke.size()+"\n" +
//                  "Balasan: "+balasan);
        if (!balasan.equals("")) {
            String pesan = balasan;
            double jml_char = pesan.length();
            double x = 160;
            int jml_sms = (int) Math.ceil(jml_char/x);
            if (jml_sms == 0) {
                jml_sms = 1;
            }
            for (int i=0; i < jml_sms; i++) {
                String psn = "";
                if (pesan.length() > x) {
                    psn = pesan.substring(0, (int) x);
                    pesan = pesan.substring((int) x);
                } else {
                    psn = pesan;
                }
                Q1.exec("INSERT INTO sms_keluar VALUES ('', " +
                        "'"+no+"', "+
                        "'"+m.util.quote(psn)+"', "+
                        "'A', '0', CURRENT_DATE, CURRENT_TIME)");
            }
        }
    }
```

UNIVERSITAS ISLAM INDONESIA

FAK. TEKNOLOGI INDUSTRI

Lampiran C.

Daftar Jalur Bus Kota Yogyakarta

Jalur 1

Term. Giwangan – RR Selatan – Ngangkruk – Ngeksigondo – M. Supeno – Jokteng Wetan – Jokteng Kulon – Jl. Tendean – HOS Cokroaminoto – Kyai Mojo – Sudirman – Cik Di Tiro – Lingkar UGM – Cik Di Tiro – Suroto – Yos Sudarso – Faridan M Noto – Diponegoro – Kyai Mojo – Cokroaminoto – Tendean – Sugeng Jeroni – Jokteng Kulon – Jokteng Wetan – Supeni – Gedong Kuning – Ngangkruk – RR Selatan – Term. Giwangan

Jalur 2

Term. Giwangan – Pramuka – M. Supeno – Sugiyono – Sisingamangaraja – RR Selatan – Jl. Paris – Katamso – Suryotomo – Abu Bakar Ali – Suroto – Cik Di Tiro – Lingkar UGM

Jalur 3

Term. Giwangan – RR Selatan – Sisingamangaraja – Sugiyono – Tamansiswa – Gayam – Kenari – Tut Harsono – Adisucipto – Gejayan – RR Utara – Jl. Kaliurang – Lingkar UGM

Jalur 4

Giwangan – RR Selatan – Ngangkruk – Gd. Kuning – Kusumanegara – Suryotomo – Suroto – Lingkar UGM – C. Simanjuntak – Sudirman – Mangkubumi – Malioboro – a.yani-Senopati –sultan agung- Kusumanegara – Gd Kuning – RR Selatan - Giwangan

Jalur 5

Giwangan – Pramuka – Joteng Wetan – Jl. Pr Tritis – Mangkuyudan – Panjaitan – letjend. MT Haryono – Wahid Hasyim – Bhayangkara – Tent. Pelajar – Jl. Magelang – Jombor – RR Utara - Lingkar UGM

Jalur 6

Giwangan – Gambiran – Veteran – Kusumanegara – Hayam Wuruk – Yos Sudarso – Wahidin – Yohanes – Lingkar UGM-

Jalur 7

Term. Giwangan – Pramuka – Kemerdekaan – Gambiran – Veteran – Janti Gd Kuning – Janti – Adisucipto – Gejayan – RR Utara – Lingkar UGM

Jalur 8

TIDAK DIJALANKAN

Jalur 9

Term. Giwangan – Pramuka – Ngeksigondo – Gd Kuning – Kusumanegara – Tamansiswo – Kol. Sugiono-Jokteng Wetan – Katamso – Senopati – KHA Dahlan – Wahid Hasyim –S. Parman- Patangpuluhan – IKIP PGRI –sonosewu- – Jl. Wates – RR Selatan – Bugisan – S. Parman – Wahid Hasyim- KHA Dahlan – Senopati- Katamso- Jokteng Wetan – Kol. Sugiono- Tamansisiwa – Suryopranoto – Mangunsarkoro – Kusumanegara – Gd Kuning – Ngeksigondo – Pramuka – Term. Giwangan

Jalur 10

Term. Giwangan – RR Selatan – Rejowinangun – Kebun Raya – SGM – Aipda Tut Harsono – Kenari – Bausasran – Mataram – Abu Bakar Ali – Atmo Sukarto – Kusbini – Langensari – Munggur – Adisucipto – Kledokan – Babarsari – RR Timur – Term. Condong Catur – RR Timur – Babarsari – Kledokan – Adisucipto – Urip Sumoharjo – Suroto – Mataram – Hayam Wuruk – Bausasran – Kenari – Veteran – Gambiran – Term. Giwangan

Jalur 11

Term. Giwangan – RR Selatan – Rejowinangun – Gd Kuning – Kemasan– Pembayun – Tegalgendu – Tegalturi – Menukan – Sisingamangaraja - Pr Tritis – RR Selatan – Jokteng Kulon – S. Parman – Patangpuluhan – RE Martadinata – Suprapto– Jlagran Lor – Ps Kembang – Malioboro – A. Yani - KHA Dahlan – RE Martadinata – SONosewu–

Patangpuluhan – Kap. Tendean - Bugisan – RR Selatan – Jl. Paris – Menukan – Sisingamangaraja – Tri Tunggal – Sorogenen - Tegalturi – JAIM - Term. Giwangan

Jalur 12

Giwangan – Pramuka – Pandean – Glagahsari – Kusumanegara – Sultan Agung – KHA Dahlan – Cokroaminoto – Tent. Pelajar – Borobudur Plaza – Monginsidi – Lingkar UGM – Monginsidi – Borobudur Plaza – Tent. Pelajar – Cokroaminoto – KHA Dahlan – Sultan Agung – Suryopranoto – Mangunsarkoro – Kusumanegara – Glagah Sari – Veteran – Pramuka – Term. Giwangan

Jalur 14

Jombor – Monjali – Jetis – Tugu – Jl. Kaliurang – RR Utara – Term. Concat – Gejayan – Urip Sumoharjo – Wahidin – Langensari – Mojo- sanggarahan - Gondosuli – Cendana – Kusumanegara – Veteran – M. Supeno – Pramuka – Term. Giwangan – Jl. Imogiri – Pramuka – Pandean – Glagahsari – Kusumanegara – Cendana – Gondosuli – Munggur – Gejayan – Term. Concat – RR Utara – Jl. Kaliurang – Jetis – Monjali – Jombor

Jalur 15

Term. Giwangan – Pramuka – Sisingamangaraja – Menukan – Mangkuyudan – Gading – Jokteng Timur – Katamso – Ibu Ruswo – KHA Dahlan – Jl. Wates – Ps. Gamping – Sidoarum – Jl. Godean – Pingit – Monginsidi – Jetis – Lingkar UGM

Jalur 16

Term. Giwangan – Tegalturi – Sisingamangaraja – Menukan – Jl. Paris – Katamso – Suryotomo – Shopping – Senopati – Sultan Agung – Suryotomo – Juminhan – Bausasran- gayam - Sukonandi – Kusumanegara – Cendana – Kenari – Tut Harsono – Adisucipto – Gejayan – Term. Concat

Jalur 17

Term. Concat – Gejayan – Colombo – Yohanes – Sudirman – Tugu – AM. Sangaji- Jetis – Monginsidi- Borobudur Plaza – Pingit – Samsat – Suprapto – Wirobrajan – Tendean – S. Parman – Jokteng Kulon – Jokteng Wetan – Tamansiswa – Suryopranoto – Mangunsarkoro – Kusumanegara – Veteran – P. Kemerdekaan - Pramuka – **Term.**

Giwangan – Pramuka – Veteran – Kusumanegara – Tamansiswa – Jokteng Wetan – Jokteng Kulon – S. Parman – Wirobrajan – Bhayangkara – Pingit – Borobudur Plaza – Jetis – Tugu – C. Simanjuntak – Terban – Colombo – Gejayan – Term. Concat

Jalur 18

TIDAK DIJALANKAN

Jalur 19

Term. Jombor – RR Barat – Jl. Godean – Kyai Mojo – Tugu – P. Mangkubumi – Malioboro – KHA Dahlan – Wirobrajan – HOS Cokroaminoto – Jl. Godean – RR Barat – Term. Jombor