APLIKASI CHATTING MULTIPLE CLIENT/SERVER DENGAN MENGGUNAKAN JAVA

TUGAS AKHIR

Diajukan Sebagai Salah Satu Syarat untuk
Memperoleh Gelar Sarjana Teknik Informatika

disusun oleh :

Nama : Demiyanti Anugraheny
No. Mahasiswa : 02 523 112

JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INDUSTRI
UNIVERSITAS ISLAM INDONESIA
YOGYAKARTA
2007

LEMBAR PERNYATAAN KEASLIAN
HASIL TUGAS AKHIR

Saya yang bertandatangan di bawah ini,

Nama : Demiyanti Anugraheny
No.Mahasiswa : 02 523 112

Menyatakan bahwa seluruh komponen dan isi dalam Laporan Tugas Akhir dengan judul **APLIKASI CHATTING MULTIPLE CLIENT/SERVER DENGAN MENGGUNAKAN JAVA** yang diajukan untuk diuji pada tanggal 25 September 2007 adalah hasil karya saya sendiri.

Apabila di kemudian hari terbukti bahwa ada beberapa bagian dari karya ini adalah bukan hasil karya saya sendiri, maka saya siap menanggung resiko dan konsekuensi apapun.

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

Yogyakarta,14 September 2007

Demiyanti Anugraheny

(NIM 02 523 112)

LEMBAR PENGESAHAN PEMBIMBING

APLIKASI CHATTING MULTIPLE CLIENT/SERVER DENGAN MENGGUNAKAN JAVA

TUGAS AKHIR

oleh :

Nama Mhs : Demiyanti Anugraheny

No. Mhs : 02 523 112

Yogyakarta, 14 September 2007

Telah Diterima dan Disetujui dengan baik oleh

Dosen Pembimbing,

Taufiq Hidayat, ST., M.C.S.

LEMBAR PENGESAHAN PENGUJI

APLIKASI CHATTING MULTIPLE CLIENT/SERVER DENGAN MENGGUNAKAN JAVA

TUGAS AKHIR

Oleh :

Nama : Demiyanti Anugraheny

No. Mahasiswa : 00 523 112

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

Yogyakarta, 26 Oktober 2007

Tim Penguji,

Taufiq Hidayat, ST, MCS
Ketua

Hendrik, ST
Anggota I

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

Mengetahui,
Ketua Jurusan Teknik Informatika
Universitas Islam Indonesia

Yudi Prayudi, S.Si, M.Kom.

Demiyanti Anugraheny would say thanks to :

- Mr. Taufiq Hidayat, ST.,MCS , dosen pembimbingku yang paling baek makasih ya Pak...atas segala bimbingan, petunjuk dan kemudahan yang diberikan.
- Mama n Papa, terima kasih telah senantiasa berdo'a untukku dan selalu memaafkan semua kesalahan yang kuperbuat. Semoga Allah swt selalu mengasihi dan menyayangi kalian.

Serta orang-orang yang selama ini telah mendengarkan, mengerti, membantu, dan selalu mengasihi aku :

- Mas Bagus , my lovely brother....do'a dan bantuan materimu sangat berarti bagiku.....Cepet pulang y masq. Semoga cepet dapet jodoh yang selama ini kau cari...
- Mas Ade , my lovely brother too...terimakasih atas bantuan dan perhatian yang kau berikan. Dan selamat atas kelahiran anak pertama...tunggu aku di Makasar y bro...
- Mbak Ayu' , kakak iparku yang baek...terimakasih atas semua info dietnya, it works 4 me...luv u dech...
- Ella a.k.a Tata , keponakan pertamaku...Welcome to the real world... buka matamu dan tersenyumlah pada dunia...
- Mbak Ayu , kakak sepupuku tercinta...terimakasih telah menemaniku berjuang selama ini, akhirnya aku selesai juga....
- Uci , Sahabatku tercinta...terimakasih telah menjadi pendengar setiaku. Keyakinanmu telah menginspirasiku. Ayo kerjain skripsinya, jangan biarkan cita-citamu menunggu. Love u sis....
- Rachma "mama" Yunita, sahabat dan tetangga kamarku,serta teman "jjm"ku. Thanx 4 being such a nice friend....
- Adia "Arindia" Karina , my new best friend....Finally, we did it....

- Sari, ayo semangat, semoga sukses pendadaran nanti...
- Danang, Semangat nang, trust me kesalahan yang sama tidak akan terulang lagi. Semoga sukses pendadaran nanti....
- Dany_why, terimakasih atas perjuanganmu selama ini...Thanx 4 all.....
- Mas Kampo , meskipun kita baru aja kenalan, kata-kata bijakmu telah merasuk ke dalam hatiku...Thanx 4 everything, kutunggu kau di Yogya...
- Teman-teman SMAku : Surya "Pak Ketua", Nanux, Maria, Dini, Astri, Retno, Rochmadi, Tri M, Puji ...thanx 4 everything. Semoga persahabatan kita selalu abadi...
- Seluruh oknum yang menghuni "Kos Ijo" : Pak kos, Eno, Penyox, Ella, Dhani Pratita, Mahendra, Catur, Hana_aw, Diah "Beo", Rachma "mama", Rani, Elli (new member)...Makasih telah buatku betah...Kalian akan selalu kuingat...
- "Kos Ijo" , my second home...terimakasih telah menjadi tempatku bernaung selama ini...I'm gonna miss u...
- Sahabat-sahabat VOIP Informatika 02, makasih banget atas kekompakan dan persahabatan kita. I Love You all..
- Bagian Pengajaran FTI UII, Perpustakaan fakultas FTI UII, Jurusan FTI UII ... terimakasih atas kerjasamanya.
- Semua communitas INFORMATIKA FTI UII semua angkatan dan FTI UII ... kampusku yang nyaman dan membanggakan ...
- Jogja KU ...My second hometown...It's a lovely city...It's a fact...
- Dan semua orang yang telah banyak berperan dalam hidupku.
- Mengenal kalian membuatku mengerti arti kehidupan yang sesungguhnya, dan mengenal kerasnya dunia. This is the real world my friend. Face it....

MOTTO

"Apabila tergelincir orang berilmu maka tergelincirlah seluruh alam dan makhluk."

(Umar bin Khaththab)

Orang-orang yang berhasil di dunia ini adalah orang-orang yang bangkit dan mencari keadaan yang mereka inginkan,dan jika tidak menemukannya, mereka akan membuatnya sendiri.

(George Bernard Shaw)

Pada dasarnya, hidup mandiri adalah soal mengambil keputusan, tak selalu keputusan yang benar, tapi mudah-mudahan yang salah tak begitu banyak sampai kita kehilangan peluang.

(Unknown)

Setiap kali ada hal yang sulit dan menantang menimpaku, itu menandai awal era baru dalam hidupku

(Kimberly Kirberger)

Kegagalan adalah sukses yang tertunda...., so enjoy aja lagi

(Unknown)

Anyone who has never made a mistake has never tried anything new

(Albert Einstein)

KATA PENGANTAR

بسم الله الرحمن الرحيم

Assalamu'alaikum Wr.Wb.

Alhamdulillahirabbil'alamin, puji syukur kehadirat Allah SWT, atas limpahan hidayah, taufiq serta 'inayah-Nya, sehingga laporan tugas akhir dengan judul "**Aplikasi Chatting Multiple Client/Server Dengan Menggunakan Java**" ini dapat terselesaikan dengan baik. Shalawat serta salam semoga senantiasa tercurah atas Nabi Muhammad SAW, para kerabat, serta pengikutnya hingga hari kiamat nanti. Amin.

Laporan tugas akhir ini disusun untuk melengkapi salah satu syarat guna memperoleh gelar Sarjana Jurusan Teknik Informatika pada Universitas Islam Indonesia dan atas apa yang telah diajarkan selama perkuliahan baik teori maupun praktek, di samping laporan sendiri yang merupakan rangkaian kegiatan yang harus dilakukan setelah tugas akhir ini selesai.

Pada kesempatan ini penyusun ingin mengucapkan banyak terima kasih kepada pihak-pihak yang mempunyai andil besar dalam pelaksanaan dan penyelesaian laporan tugas akhir ini, antara lain :

1. Bapak Fathul Wahid S.T., M.Sc sebagai Dekan Fakultas Teknologi Industri Universitas Islam Indonesia.
2. Bapak Yudi Prayudi, S.Si, M.Kom selaku Ketua Jurusan Informatika Fakultas Teknologi Industri Universitas Islam Indonesia.
3. Bapak Taufiq Hidayat, ST, MCS. selaku Dosen Pembimbing Tugas Akhir. Terima kasih atas segala kesabarannya, bantuan, dukungan, pengetahuan dan semua kemudahan yang selalu diberikan.
4. Bapak, Ibu dosen Teknik Informatika dan dosen-dosen Universitas Islam Indonesia. Terimakasih atas semua ilmu yang diberikan.

5. Ayah dan Ibu serta keluargaku tercinta yang telah memberikan banyak semangat dan dorongan,serta do'a yang tiada putusnya.
6. *VOIP Community* yang memberikan semangat, dorongan, bantuan sehingga penyusun dapat meyelesaikan tugas akhir dengan baik.
7. Penghuni "Kos Ijo" yang telah memberikan banyak waktu untuk berbagi.
8. Semua pihak yang tidak bisa disebut satu persatu yang telah membantu sehingga laporan tugas akhir ini dapat terselesaikan.

Di tengah keterbatasan penyusun dalam laporan tugas akhir ini, penyusun berharap kiranya laporan ini bermanfaat bagi pembaca. Semoga Allah SWT membimbing dan menyertai setiap langkah kita. Amiin.

Wassalamu'alaikum Wr.Wb.

Yogyakarta, 14 September 2007

Penyusun

ABSTRAKSI

Salah satu model umum yang diterapkan untuk pemrograman jaringan adalah model *client / server*. Sehingga secara umum jika ingin mengembangkan suatu aplikasi jaringan, maka aplikasi tersebut dikenal juga dengan aplikasi *client / server*.

Sebagai contoh salah satu aplikasi yang menggunakan model *client / server* adalah *chatting*. Aplikasi *chatting* ini menggunakan prinsip dimana *client* atau *user* melakukan permintaan untuk terkoneksi dengan *server* dan memanfaatkan fasilitas-fasilitas yang disediakan oleh *server*.

Setelah melalui pengujian dan analisa, dapat diketahui bahwa sistem tersebut secara keseluruhan telah berjalan sesuai dengan yang diharapkan, dan mempunyai kelebihan yaitu dengan banyaknya *client* yang dapat terhubung ke jaringan dan dapat berkomunikasi satu sama lain melalui *chatting*. Selain itu aplikasi *chatting* ini dapat berjalan di setiap sistem operasi karena dibuat dengan bahasa pemrograman java yang bersifat *multi platform*.

Kata kunci : *jaringan, client, server, chatting, multi platform.*

TAKARIR

bug	kesalahan
byte	bite
bytecode	file class
client	klien
Data Sharing	berupa pengolahan sebuah data atau informasi secara bersama-sama
debugger	pencari kesalahan
device input	perangkat masukan
developer	pengembang
fair	adil
hardware	perangkat keras
Host	Tuan rumah
Interface	antarmuka
IP-Address	alamat IP
memory leaking	masalah di mana memori yang digunakan untuk objek atau variabel yang sudah tidak digunakan tidak didealokasikan sehingga menyebabkan kehabisan memori karena proses alokasi maupun dealokasi yang tidak diatur dengan baik
Message	pesan
Output	*keluaran dari sistem*
peer-to-peer	jaringan antara dua komputer
Platform	perangkat lunak/keras yang dapat menjalankan progam
platform Independent	tidak tergantung mesin atau sistem operasi
port	lokasi komunikasi dari aplikasi yang akan dihubungi
Portable	dapat dibawa-bawa

programmer	orang yang membuat program
programming	pemrograman
Receiver	penerima
Resource Sharing	berupa penggunaan perangkat keras bersama-sama
request	permintaan
Router	perangkat antara yang dapat digunakan untuk menghubungkan jaringan lokal
server	pelayan
service	pelayanan
transmitter	pengirim
Unified Modelling Language	Bahasa untuk menspesifikasikan, memvisualisasi, membangun dan mendokumentasikan *artifact* (bagian dari informasi yang digunakan atau dihasilkan oleh proses pembuatan perangkat lunak, *artifact* tersebut dapat berupa model, deskripsi atau perangkat lunak) dari sistem perangkat lunak, seperti pada pemodelan bisnis dan sistem non perangkat lunak lainnya.
User friendly	Mudah digunakan
user	pengguna
view	tampilan

DAFTAR ISI

DAFTAR GAMBAR

BAB I
PENDAHULUAN

1.1 Latar Belakang

Dalam beberapa tahun terakhir ini, teknologi komputer telah berkembang sangat pesat. Akibat perkembangan teknologi yang sangat pesat ini, maka teknologi-teknologi menjadi sangat terkait. Perbedaan-perbedaan yang terjadi dalam pengumpulan, pengiriman, penyimpanan dan pengolahan informasi telah dapat diatasi. Dalam hal ini memungkinkan pengguna dapat memperoleh informasi secara cepat dan akurat.

Sampai saat ini, teknologi dari jenis *personal computer* hingga *super computer* telah mengalami perkembangan, sehingga meningkatkan kapasitas dan pengolahan data. Model komputer tunggal yang melayani seluruh tugas-tugas komputasi suatu organisasi telah diganti oleh sekumpulan komputer yang berjumlah banyak dan terpisah tetapi masih saling berhubungan dalam melaksanakan tugasnya. Sistem ini disebut sebagai jaringan komputer.

Jaringan komputer adalah sebuah sistem yang terdiri atas komputer dan perangkat jaringan lainnya yang bekerja bersama-sama untuk mencapai suatu tujuan yang sama.

Dalam sebuah jaringan komputer biasanya terdapat banyak komputer yang terhubung ke sebuah atau beberapa *server*. Administrator adalah orang yang bertindak sebagai pengelola jaringan dan biasanya berada pada *server* jaringan. Apabila jumlah komputer yang terhubung ke jaringan cukup banyak, maka pemantauan terhadap kegiatan yang dilakukan oleh *client* akan menjadi sulit. Jika *client* melakukan kegiatan yang menyimpang dari ketentuan, administrator harus mendatangi *client* untuk mengingatkan. Hal ini tidak efektif terutama jika jumlah *client* cukup banyak. Oleh karena itu diperlukan suatu sistem yang diharapkan dapat memantau kegiatan *client* dan mengingatkan client apabila melakukan kesalahan, serta menghentikan kegiatan yang dilakukan apabila perlu. Pengendalian suatu

terminal/ *client* tanpa bersentuhan secara fisik melalui komputer *server* sangat diperlukan oleh seorang administrator di jaringan yang cukup besar karena begitu banyaknya komputer-komputer yang terhubung untuk dipantau.

1.2 Rumusan Masalah

Berdasarkan latar belakang di atas, maka permasalahan yang akan ditekankan adalah **"Bagaimana merancang suatu aplikasi Chatting Multiple Client/Server dengan menggunakan bahasa pemrograman Java"**.

1.3 Batasan Masalah

Dalam hal ini, batasan masalah yang harus diperhatikan adalah :

1. Aplikasi terdiri dari dua bagian yang terpisah, yaitu aplikasi untuk *server* dan aplikasi *client*
2. Aplikasi ini dibuat menggunakan pemrograman *socket.*
3. Aplikasi ini dijalankan di jaringan lokal.

1.4 Tujuan Penelitian

Penelitian ini dilakukan dengan beberapa tujuan yang dapat dijelaskan sebagai berikut :

1. Membuat suatu aplikasi untuk komunikasi antar sesama *client* yang diatur oleh *server*
2. Mempermudah komunikasi dalam jaringan.

1.5 Manfaat Penelitian

Adapun manfaat dari penelitian adalah :

1. Memudahkan pengelolaan komputer yang terhubung ke jaringan
2. Memudahkan proses pemantauan pengguna komputer yang terhubung ke jaringan.

3. Memudahkan *client* untuk berkomunikasi dengan *client* lainnya

1.6 Metodologi Penelitian

1.6.1 Metode Pengumpulan Data

Metode pengumpulan data adalah metode yang digunakan untuk mengumpulkan data yang diperlukan dalam penelitian. Metode ini meliputi : studi pustaka, yaitu pengumpulan data dengan cara melakukan studi, analisis dan dokumentasi literatur, dan sumber catatan lain yang berkaitan dengan permasalahan yang dibahas.

1.6.2 Metode Pengembangan Sistem

Metode pengembangan sistem disusun berdasarkan hasil dari data yang sudah diperoleh. Metode ini meliputi :

1. Analisis Kebutuhan

   Analisis ini dilakukan untuk mengolah data yang sudah diperoleh dan mengelompokkan data sesuai dengan kebutuhan perancangan.

2. Perancangan

   Tahap ini merupakan tahap perancangan sistem, yaitu mendefinisikan kebutuhan yang ada, menggambarkan bagaimana sistem dibentuk dan persiapan untuk membangun aplikasi.

3. Implementasi

   Tahap ini adalah penerjemahan rancangan dalam tahap desain ke dalam bahasa pemrograman komputer yang telah ditentukan sebelumnya.

4. Pengujian

   Setelah aplikasi selesai dibuat, maka pada tahap ini merupakan uji coba terhadap program tersebut. Sehingga analisis hasil implementasi yang didapat dari sistem disesuaikan dengan kebutuhan sistem tersebut. Jika penerapan sistem sudah

berjalan dengan lancar, maka sistem dapat diimplementasikan untuk melakukan perhitungan biaya penugasan dalam penentuan lokasi dan tata letak fasilitas.

1.7 Sistematika Penulisan

Untuk mempermudah pembahasan tugas akhir ini maka dalam penyusunannya penulis membagi pokok-pokok permasalahan kedalam lima bab sebagai berikut :

Bab I Pendahuluan

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

Bab II Landasan Teori

Pada bab ini berisi mengenai teori–teori yang berhubungan dengan penelitian diantaranya mengenai teori tentang jaringan, aplikasi *chatting*, *socket client-server*.

Bab III Metodologi

Bab ini memuat tentang langkah-langkah yang dibutuhkan dalam penelitian, yaitu : analisis kebutuhan, perancangan, dan implementasi sistem.

Bab IV Hasil dan Pembahasan

Bab ini berisi tentang hasil dan pembahasan yang telah diperoleh dari perangkat lunak yang dibuat. Serta memuat tentang pengujian perangkat lunak, pengujian sistem aplikasi dengan data masukan dan analisis kinerja dari sistem.

Bab V Penutup

Berisi kesimpulan dari proses pengembangan perangkat lunak, serta saran-saran yang perlu diperhatikan berdasar keterbatasan – keterbatasan yang ditemukan dan asumsi yang dibuat selama TA.

BAB II
LANDASAN TEORI

2.1. Konsep Jaringan Komputer

Jaringan komputer adalah sebuah sistem yang terdiri atas komputer dan perangkat jaringan lainnya yang bekerja bersama-sama untuk mencapai suatu tujuan yang sama. Agar dapat mencapai tujuan yang sama, setiap bagian dari jaringan komputer meminta dan memberikan layanan (*service*). Pihak yang meminta layanan disebut klien (*client*) dan yang memberikan layanan disebut pelayan (*server*). Arsitektur ini disebut dengan sistem *client-server*, dan digunakan pada hampir seluruh aplikasi jaringan komputer [www.id.wikipedia.org].

Tujuan dibangunnya suatu jaringan komputer adalah membawa informasi secara tepat dan tanpa adanya kesalahan dari sisi pengirim (*transmitter*) menuju ke sisi penerima (*receiver*) melalui media komunikasi.

Secara umum, jaringan komputer mempunyai beberapa manfaat yang lebih dibandingkan dengan komputer yang berdiri sendiri dan dunia usaha telah mengakui bahwa akses ke teknologi moderen selalu memiliki keunggulan kompetitif dibandingkan pesaing yang terbatas dalam bidang teknologi.

Adapun manfaat yang didapat dalam membangun jaringan komputer adalah sebagai berikut :

1. Penggunaan perangkat keras bersama-sama
2. Media komunikasi
3. Integrasi data
4. Pengembangan dan pemeliharaan peralatan
5. Keamanan data
6. Sumber daya lebih efisien dan informasi terkini

Jaringan komputer dapat dibedakan berdasarkan jarak, yaitu :

1. *Local Area Network (LAN)*

   Jaringan ini digunakan untuk menghubungkan komputer-komputer pribadi dan *workstation* dalam suatu perusahaan yang menggunakan peralatan secara bersama-sama dan saling bertukar informasi. Biasanya jaringan ini dimiliki oleh perusahaan tanpa menggunakan fasilitas dari perusahaan telekomunikasi umum [Wahana,2003].

2. *Metropolitan Area Network (MAN)*

   Jaringan ini pada dasarnya merupakan versi LAN yang berukuran lebih besar dan biasanya memakai teknologi yang sama dengan LAN. MAN merupakan pilihan untuk membangun jaringan komputer antar kantor dalam suatu kota. MAN dapat mencakup perusahaan yang memiliki kantor-kantor yang letaknya sangat berdekatan dan MAN mampu menunjang data dan suara, bahkan bisa disambung dengan jaringan televisi kabel. Jaringan ini memiliki jarak dengan radius 10-50 km. Di dalam jaringan MAN hanya memiliki satu atau dua buah kabel yang fungsinya untuk mengatur paket melalui kabel output. [Wahana,2003].

3. *Wide Area Network (WAN)*

   Merupakan sebuah jaringan yang sangat luas, karena radiusnya mencakup sebuah negara dan benua. Pada sebagian besar WAN, komponen yang dipakai dalam komunikasi biasanya terdiri dari dua komponen, yaitu: kabel transmsi dan elemen switching. Kabel transmisi berfungsi untuk memindahkan bit-bit dari satu komputer ke komputer lainnya, sedangkan elemen switching di sini adalah sebuah komputer khusus yang digunakan untuk menghubungkan dua buah kabel transmisi atau lebih. Saat data yang dikirimkan sampai ke kabel penerima, elemen switching harus memilih kabel pengirim untuk meneruskan pesan tersebut. Pada sebagian besar WAN, jaringan terdiri dari sejumlah banyak kabel atau saluran telepon yang menghubungkan sepasang router. Router adalah perangkat antara yang dapat digunakan untuk menghubungkan jaringan lokal yang sama pada lapisan jaringan OSI. [Wahana,2003].

2.2 Aplikasi Chatting

Aplikasi *chatting* merupakan suatu aplikasi yang dibuat untuk memungkinkan komunikasi antara 2 atau lebih *user* dalam suatu jaringan yang diatur oleh sebuah *server*.

Sebagai salah satu media komunitas, layanan bernama chatting memang menjadi media paling cepat untuk penyebaran informasi, di samping email. Pengguna yang sedang sama-sama *online*, bisa langsung berinteraksi melalui teks-teks yang diketik atau bahkan berbicara langsung melalui *microphone*.

Dengan *chating* lewat internet anda bisa mendapatkan banyak teman dari penjuru dunia. Hal keamanan merupakan faktor penting bagi kita dalam berbagi informasi. Karena itu dibutuhkan suatu aplikasi *chatting* yang dapat menjamin keamanan tinggi sehingga menciptakan suasana yang akrab dan nyaman.

2.3 Server

Server adalah terminal induk dimana semua kontrol terhadap jaringan terpusat. *Server* berfungsi untuk melayani dan mengatur semua komputer yang terhubung ke dalam jaringan, termasuk hubungan dengan perangkat tambahan. Beberapa bentuk pelayanan yang diberikan oleh *server*, antara lain :

- *Resource Sharing*, berupa penggunaan perangkat keras bersama-sama, seperti *printer, scanner,* dan lain-lain
- *Data Sharing,* berupa pengolahan sebuah data atau informasi secara bersama-sama
- Mengatur keamanan dalam jaringan
- Mengatur hak akses bagi pengguna

Idealnya komputer yang digunakan sebagai *server* spesifikasinya lebih tinggi dari komputer *client*, mengingat tugas dan fungsinya yang lebih kompleks dari komputer *client*.

2.4 Client / Multiple Client

Client atau *workstation* merupakan komputer dimana pengguna jaringan bekerja. *Client* bisa mengakses data dan informasi dalam jaringan dengan batasan tertentu yang disebut dengan hak akses.

Multiple client mempunyai pengertian bahwa dalam sebuah jaringan komputer terdapat banyak komputer *client* yang dapat mengakses jaringan tersebut.

2.5 Bahasa Pemrograman Java

Java dibuat dan diperkenalkan pertama kali oleh sebuah tim *Sun Mycrosystems* yang dipimpin oleh Patrick Naughton dan James Gosling pada tahun 1991 dengan *code name* ***Oak***. Tahun 1995 Sun mengubah nama *Oak* tersebut menjadi *Java*. Ide pertama kali kenapa Java dibuat adalah karena adanya motivasi untuk membuat sebuah bahasa pemrograman yang bersifat *portable* dan *platform Independent* (tidak tergantung mesin atau sistem operasi) yang dapat ditanamkan ada berbagai macam peralatan elektronik

Java merupakan suatu teknologi yang unik dan revolusioner dan merupakan teknologi pertama di dunia software yang memiliki semboyan "*write once, run anywhere*". Semboyan tersebut telah terbukti karena banyak program Java dapat dijalankan di berbagai *platform* Sistem Operasi, seperti Linux, Windows maupun Unix.[Susanto,2003]

Java telah mengatasi masalah portabilitas yang sering menjadi kendala dan hambatan dalam pembuatan suatu aplikasi *software* karena program java dapat dijalankan secara langsung tanpa banyak perubahan berarti di sistem operasi atau *platform* lain.

Java juga didesain untuk menghasilkan program dengan seminimal mungkin *bug* karena kelebihannya sebagai berikut :

1 Java didesain untuk menghilangkan alokasi memori dan dealokasi memori secara manual. Java memiliki *garbage collector* otomatis yang mencegah adanya *memory leak. Memory leak* adalah masalah yang sering dihadapi programmer C

dan C++ di mana memori yang digunakan untuk objek atau variabel yang sudah tidak digunakan tidak didealokasikan sehingga menyebabkan kehabisan memori karena proses alokasi maupun dealokasi yang tidak diatur dengan baik.

2 Java memiliki array yang tidak memerlukan pointer sehingga memudahkan para programer.

3 Java menghilangkan pewarisan berganda yang terdapat pada C++. Walaupun kelihatannya lebih sebagai suatu kekurangan, namun banyak para ahli yang mengakui bahasa konsep pewarisan berganda lebih banyak mengakibatkan kerugian daripada keuntungan. Java telah didesain sedemikian rupa sehingga tidak akan diperlukan teknik ini dalam pembuatan program apa pun.

4 Java mengurangi pointer aritmetik. Pengaksesan lokasi memori secara langsung dengan menggunakan pointer memungkinkan program untuk melakukan suatu tindakan yang tidak seharusnya atau tidak boleh dilakukan. Untuk mengurangi dan menghilangkan kemungkinan kesalahan seperti ini, penggunaan pointer pada Java telah dibatasi dengan menggunakan reference.

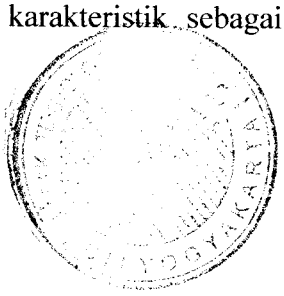
5 Java menghilangkan banyak kebingungan apabila terjadi proses *assignment* (pemberian nilai) pada statemen kondisional seperti berikut :

- If (varnya = 5)

Kode di atas menyebabkan program Java tidak dapat dikompilasi karena Java membedakan tanda = yang digunakan untuk pemberian nilai dan untuk pengecekan kondisi *True* atau *False* yang ahrus menggunakan tanda = ganda (= =).

6 Java menghilangkan *multiple inheritance* pada C++ dan menggunakan *interface* yang memiliki kemampuan yang sama tetapi lebih sederhana [Indrajani,2004].

Sebagai bahasa pemrograman, java termasuk dalam kategori bahasa pemrograman tingkat tinggi (*high-level*) yang memiliki karakteristik sebagai berikut :



a. *Simple* (sederhana)

Java menggunakan bahasa yang sederhana dan mudah dipelajari.

b. *Object-oriented* (Berorientasi objek)

Java telah menerapkan konsep pemrograman berorientasi objek yang modern dalam implementasinya.

c. *Distributed* (terdistribusi)

Java didisain untuk berjalan pada lingkungan yang terdistribusi seperti halnya internet.

d. *Interpreted* (tafsir)

Aplikasi Java dapat dieksekusi pada *platform* yang berbeda-beda dengan melakukan interpretasi pada *bytecode*.

e. *Robust* (kuat)

Java mendukung pemrograman yang bebas kesalahan yang bersifat *strongly typed* (seperti bahasa pascal) memiliki *run time checking.*

f. *Secure* (aman)

Aplikasi Java dapat dipastikan keamanannya, terutama untuk aplikasi *internet.*

g. *Achitecture-Neutral* (Netral secara arsitektur)

Java tidak terikat pada suatu mesin atau sistem operasi tertentu.

h. *Portable*

Program Java dapat dieksekusi di-*platform* manapun selama tersedia Java Virtual Machine untuk platform tersebut.

i. *High-Performance* (berkinerja tinggi)

Bytecode Java telah sangat teroptimasi sehingga proses eksekusi program dapat dilakukan dengan cepat sekalipun dilakukan dengan cara interpretasi terhadap bytecode.

j. *Multithreaded* (Multithreading)

Java mendukung *Multithreading* secara langsung, sehingga banyak *thread* yang dapat dieksekusi. [Indrajani,2004].

k. *Dynamic* (Dinamis)

Program java dapat melakukan suatu tindakan pada saat eksekusi program dan bukan pada saat kompilasi program [Indrajani,2004].

2.6 Protokol TCP/IP

Dalam jaringan komputer, kita mengenal adanya protokol. Protokol adalah suatu aturan atau mekanisme dimana dua atau lebih komputer dapat saling berinterkoneksi. Sebuah protokol tentu saja mendefinisikan suatu format paket data yang akan dipertukarkan untuk menunjang mekanisme tersebut.

Salah satu protokol yang paling banyak digunakan saat ini adalah protokol TCP/IP. Protokol ini juga yang digunakan dalam jaringan terbesar, yaitu internet. Salah satu fungsi utama protokol TCP/IP adalah menyediakan sebuah mekanisme komunikasi point-to-point. Satu proses pada satu komputer berkomunikasi dengan proses lain di komputer yang lain, atau pada komputer yang sama. Komunikasi dibentuk dengan dua buah aliran data (*stream*). Satu *stream* membaca data dari satu proses ke proses lain, sedangkan satunya lagi mengirim data dengan arah yang lain. Artinya ada *stream* input dan *stream* output dari tiap proses yang saling berkomunikasi. [Susanto,2003].

Setiap proses dapat membaca data yang sudah ditulis oleh proses lain, dan tentu saja dalam keadaan normal, data yang terbaca tersebut sama seperti dengan yang dituliskan sebelumnya.

Dalam hal berhubungan dengan suatu mesin dari mesin yang lain dan meyakinkan bahwa telah terkoneksi pada mesin yang diinginkan, harus ada cara yang unik mengidentifikasikan mesin dalam satu jaringan agar satu mesin dengan mesin yang lain memiliki identitas yang berlainan. Identitas inilah yang disebut sebagai alamat.

Pada protokol TCP/IP, untuk melakukan identitas tiap mesin, tiap mesin yang terhubung dalam jaringan harus memiliki alamat unik dan alamat ini diatur dengan protokol IP (*Internet Protocol*) yang memiliki panjang 32 bit.

Alamat IP memiliki dua bentuk, yaitu :

1. *DNS (Domain Name Service)*

   Misal ada sebuah *domain name* uii.ac.id, dan jika terdapat sebuah komputer yang disebut www dalam domain tersebut, nama untuk komputer tersebut secara lengkap adalah www.uii.ac.id.
2. Alamat IP yang dibentuk dengan menuliskan suatu deretan 4 kelompok angka yang masing-masing dipisahkan dengan titik (dot) dan tiap kelompok angka memiliki nilai antara 0-255. sebagai contoh adalah 192.168.3.10 [Susanto,2003]

2.7 Socket

Pada awal tahun 1980-an, ARPA (*The Advanced Research Projects Agency*) mendirikan kelompok di Universitas California, Berkeley untuk memindahkan perangkat lunak TCP/IP ke dalam sistem operasi UNIX. Para perancang membuat sebuah *interface*/antarmuka yang digunakan oleh aplikasi untuk berkomunikasi. Oleh karena itu digunakan *system calls* yang sudah ada dan menambahkan *system calls* yang baru yang menyediakan fungsi-fungsi TCP/IP. Hasil dari proyek tersebut menghasilkan *socket* atau *socket interface*. *Socket* API (*Application Programming Interface*) memberikan fungsi-fungsi umum yang mendukung komunikasi jaringan dengan bermacam-macam protokol jaringan. Protokol TCP/IP memberikan mekanisme untuk melakukan transfer data. Pada umumnya TCP/IP membolehkan *programmer* untuk membangun komunikasi antara dua aplikasi untuk melewatkan data. Dan kita bisa mengatakan proses tersebut merupakan proses komunikasi *peer-to-peer* atau *end-to-end*. Walaupun TCP/IP digunakan untuk mengirimkan data antara komputer yang berbeda akan tetapi TCP/IP tidak memberi tahu kapan interaksi terjadi dan bagaimana cara mengorganisasi komunikasi yang terjadi. Untuk mengatasi

masalah tersebut, maka dibuatlah suatu metode yang digunakan untuk mengorganisasi TCP/IP dan dapat digunakan oleh seluruh aplikasi yang mengimplementasikan TCP/IP. Metode tersebut dikenal dengan *Client-Server*

Setiap aplikasi jaringan selalu menggunakan metode *client-server* untuk komunikasinya. Jika tanpa metode *client-server* maka komunikasi antara dua komputer tidak akan pernah terjadi dikarenakan kecepatan komputer yang sangat cepat. Sebuah komunikasi akan terjadi jika aplikasi yang akan berkomunikasi dalam keadaan hidup. Misal, andai kita membuat koneksi antara aplikasi A dengan aplikasi B. Pada saat aplikasi A dijalankan, dia langsung mengeksekusi dan mengirimkan pesan ke aplikasi B. Padahal kita belum sempat menghidupkan aplikasi B sehingga kondisi ini akan menyebabkan *error*. Masalah di atas dapat diatasi dengan metode *client-server*, karena setiap aplikasi yang akan berkomunikasi minimal mempunyai satu *server* dan satu *client*. Aplikasi yang bertugas sebagai *server* akan dieksekusi terlebih dahulu dan akan berada dalam mode *listening* atau mendengarkan *client*. Pada saat *server* mendengarkan *client*, kita dapat menghidupkan aplikasi yang bertugas sebagai *client* dan mulai membangun koneksi dengan *server*. Dengan metode *client-server*, komunikasi akan terorganisasi dengan baik dan tidak akan terjadi *error* koneksi seperti contoh di atas. Struktur aplikasi *server* berbeda dengan *client*. *Server* lebih sedikit rumit karena dia memberikan *service*/pelayanan kepada *client*, sedangkan *client* hanya memberikan *request*/permintaan kepada *server*.

Socket adalah sebuah abstraksi perangkat lunak yang digunakan sebagai suatu "terminal" dari suatu hubungan antara dua mesin atau proses yang saling berinterkoneksi. Di setiap mesin yang saling berinterkoneksi, harus terpasang *socket* [Susanto,2003].

Socket client-server merupakan media yang digunakan untuk membangun aplikasi yang berbasis jaringan. *Socket* akan menjembatani komunikasi antara *server* dan *client*. Di atas sudah dijelaskan bahwa *socket* merupakan *interface* komunikasi jaringan. Jadi *socket* bisa kita anggap sebagai pintu masuk dan jembatan komunikasi jaringan.

2.8 Port

Selain menangani identifikasi mesin dengan alamat IP, protokol TCP/IP juga memiliki mekanisme untuk mengidentifikasikan sebuah proses pada sebuah mesin. Identifikasi untuk proses ini dikenal dengan alamat *port* yang memiliki panjang 16 bit. *Port* merupakan lokasi komunikasi dari aplikasi yang akan dihubungi. Beberapa layanan (proses) sudah memiliki alamat *port* yang baku, oleh karena layanan (proses) tersebut sudah terdefinisikan secara standar, dan ini yang dikenal dengan *well known port*, yang memiliki nomor antara 0-1023. contohnya adalah layanan HTTP yang memiliki alamat *port* 80.

Protokol telah memutuskan untuk menetapkan penggunaan nomor *port* yang digunakan untuk server yang spesifik, nomor *port* tersebut adalah *well known port numbers*. Dan membagi nomor port dalam 3 kelompok, yaitu :

1. *Well known ports* : nomor *port* ini bermula dari 0 1023
2. *Registered ports* : nomor *port* ini bermula dari 1024 - 49.151
3. *Dynamic ports* : nomor *port* ini bermula dari 49.152 65.535 [Susanto,2003].

BAB III
METODOLOGI

3.1 Analisis Kebutuhan

3.1.1 Metode Analisis

Metode analisis merupakan suatu cara dalam menguraikan sebuah sistem melalui identifikasi, perancangan, dan implementasi suatu perangkat lunak. Tahap analisis juga mengevaluasi permasalahan dan hambatan yang terjadi di dalam membangun suatu sistem sehingga dapat dilakukan perbaikan. Metode analisis adalah langkah penting dalam perancangan perangkat lunak. Langkah ini sangat mempengaruhi perancangan yang dibuat beserta implementasinya. Metode analisis yang digunakan adalah metode analisis dengan metode analisis berorientasi obyek dengan menggunakan standar UML (*Unified Modelling Language*) sebagai alat bantunya.

Pada tahap ini dilakukan dengan pencarian dan penentuan permasalahan yang dihadapi serta semua kebutuhan seperti kebutuhan masukan sistem, kebutuhan keluaran sistem, antarmuka sistem, dan fungsi-fungsi yang dibutuhkan.

3.1.2 Hasil Analisis

Hasil analisis yang diperoleh dari sistem ini adalah proses-proses kebutuhan masukan, kebutuhan keluaran, maupun teknologi yang akan diimplementasikan. Merupakan pemilihan kebutuhan sistem yang harus diwujudkan dalam perangkat lunak, yang meliputi fungsi-fungsi yang dibutuhkan dan antarmuka yang *user friendly*.

3.1.2.1 Kebutuhan Masukan

a. Kebutuhan masukan untuk aplikasi *server* :
   1) *Server* hanya menunggu koneksi dari *client*, yang dibutuhkan untuk menampilkan daftar *client* yang *online* pada saat itu.

2) Server akan menentukan nomor port yang akan digunakan pada jaringan tersebut, dan akan disimpan sebagai defaultnya.

b. Kebutuhan masukan untuk aplikasi *client* :

1) Untuk koneksi ke *server*, *client* diharuskan untuk Login dan mengisi nama yang akan ditampilkan pada daftar *client*.

3.1.2.2 Kebutuhan Keluaran

a. Kebutuhan keluaran dari aplikasi *server* :

1) Berupa tampilan data koneksi *client* yang masuk atau keluar
2) Tampilan daftar pengguna yang terkoneksi pada saat itu.

b. Kebutuhan keluaran dari aplikasi *client* :

1) Teks area untuk menampilkan ruang *chatting*, baik ruang umum maupun pribadi.
2) Tampilan daftar pengguna yang sedang terkoneksi.

3.1.2.3 Kebutuhan Antarmuka

Kebutuhan terhadap antar muka yang dibuat mempertimbangkan kondisi supaya aplikasi menjadi menarik, sederhana dan mudah digunakan oleh pengguna. Antar muka yang diinginkan sebaik mungkin bersifat *user friendly*, sehingga pengguna dapat menggunakan sistem dengan tidak memberi kesan sulit kepada pengguna. Sistem dibangun dengan meminimalkan kesalahan sehingga dapat memberikan informasi sesuai yang dibutuhkan.

3.1.2.4 Fungsi – fungsi yang dibutuhkan sistem

Aplikasi chatting ini membutuhkan beberapa fungsi pemrograman, antara lain:

1. Fungsi untuk koneksi *client* ke *server*.
2. Fungsi untuk menampilkan informasi yang dibutuhkan baik pada *client* maupun *server*.

3. Fungsi untuk penyampaian pesan dari *client* ke *server* atau sebaliknya, baik untuk ruang umum maupun pribadi.

3.1.2.5 Kebutuhan perangkat lunak dan perangkat keras

Perangkat lunak dan perangkat keras yang dibutuhkan dalam membangun sistem ini adalah :

1. Kebutuhan perangkat lunak

   Kebutuhan *software* ini berisi mengenai kebutuhan perangkat lunak apa saja yang akan digunakan dalam perancangan sistem dan rekayasa pemrograman. *Software* yang digunakan yaitu sistem operasi Windows XP. *Compiler* yang digunakan adalah NetBeans 5.5.

2. Kebutuhan Perangkat Keras

   Kebutuhan perangkat keras ini meliputi spesifikasi perangkat keras digunakan dalam pengoperasian sistem yang telah dibuat. Spesifikasi tersebut adalah :

   processor : AMD Athlon XP 2400+
   Hard disk : 40 GB
   Memory : 512 MB
   Monitor : VGA

3.2 Perancangan

3.2.1 Metode Perancangan

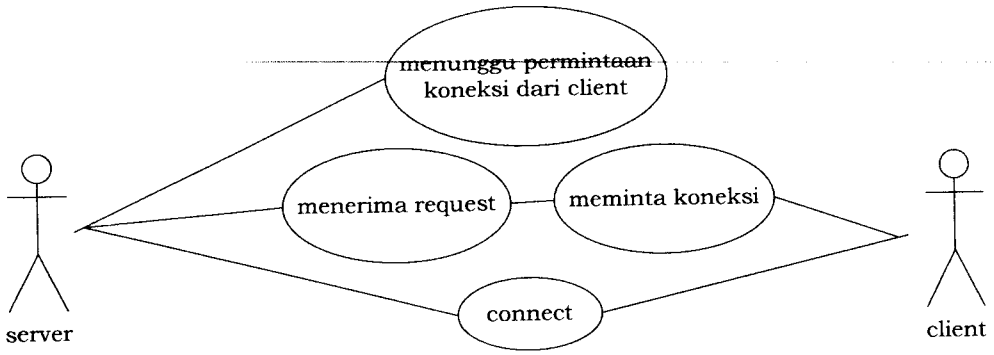
Metode yang digunakan adalah metode perancangan sistem berorientasi obyek. Dalam hal ini penggunaan UML (*Unified Modelling Language*) sangat sesuai karena aplikasi yang dibangun di atas teknologi Java yang mengimplementasikan paradigma pengembangan sistem berorientasi obyek.

3.2.2 Hasil Perancangan

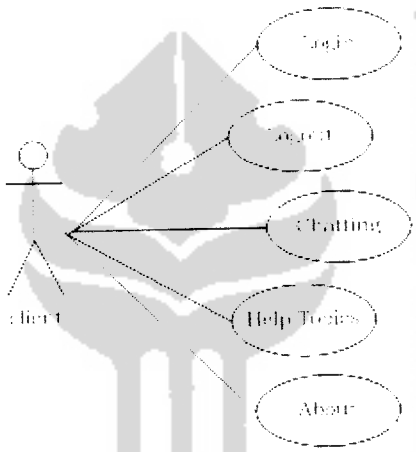
UML memiliki beberapa konsep dasar yang diabstraksikan dalam bentuk *structural classification*, *dynamic behavior*, dan *model management*. Hal terpenting dalam penggunaan UML adalah pembuatan diagram yang sesuai dengan analisis dan pengembangan sistem. Notasi-notasi UML mampu merepresentasikan rancangan sistem yang berorientasi obyek sehingga menjadi lebih mudah ketika rancangan nantinya diimplementasikan pada bahasa pemrograman berorientasi obyek seperti Java. Pada perancangan aplikasi ini terbagi menjadi tiga diagram yaitu *Use Case Diagram*, *Class Diagram*, dan *Sequence Diagram* yang merupakan bagian dari *Interaction Diagram*.

3.2.2.1 *Use Case Diagram*

Use case diagram berisi gambaran fungsionalitas yang diharapkan dari sebuah sistem dengan fokus penekanan pada apa yang dilakukan oleh sistem, bukan bagaimana sistem melakukan sesuatu. Dalam *use case diagram* terdapat dua pihak yang saling berhubungan yaitu *actor* dan *use case* yang berkaitan dengan *actor* . Dengan diagram ini dapat diketahui cakupan dari sistem, siapa saja aktor yang berperan dalam sistem dan interaksi antara aktor dengan elemen-elemen *use case* dalam sistem. Diagram *Use case* yang memperlihatkan aktor-aktor yang berperan dalam sistem, diperlihatkan pada Gambar 3.1 dan gambar 3.2.



Gambar 3.1 Diagram *Use Case* Koneksi *Client* ke *Server*



Gambar 3.2 Diagram *Use Case* aplikasi *client*

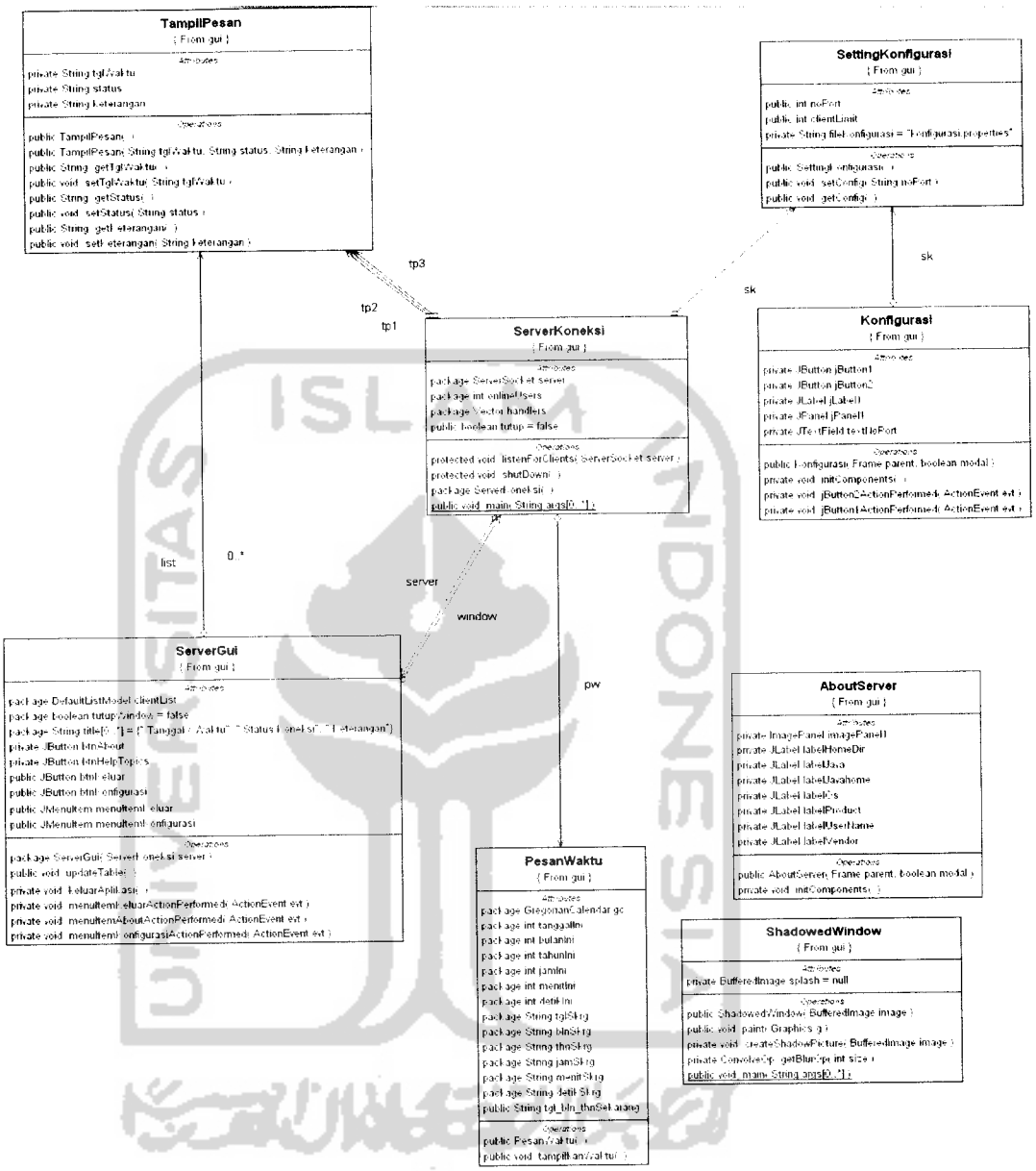
3.2.2.2 *Class Diagram*

Diagram class menggambarkan struktur *class* di dalam sistem. *Class* merepresentasikan sesuatu yang ditangani oleh sistem. *Class* dapat berhubungan dengan yang lain melalui berbagai cara yaitu *associated* (terhubung satu sama lain), *dependent* (satu *class* tergantung/menggunakan *class* yang lain), *specialed* (satu *class* merupakan spesialisasi dari *class* lainnya), atau *package* (grup bersama sebagai satu unit).

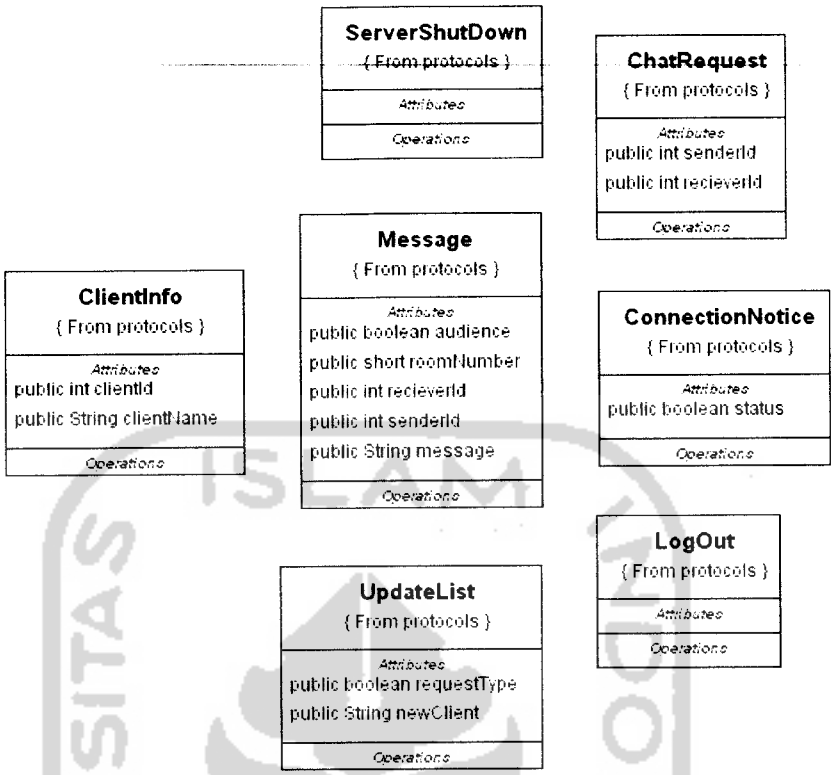
Class adalah deskripsi sekelompok obyek dari *properti* (atribut), sifat (operasi), relasi antar obyek dan semantik yang umum. *Class* merupakan *template* untuk membentuk obyek. Setiap obyek merupakan contoh dari beberapa *class* dan obyek tidak dapat menjadi contoh lebih dari satu *class*. Penamaan *class* menggunakan kata benda tunggal yang merupakan abstraksi yang terbaik. Pada UML, *class* digambarkan dengan segi empat yang dibagi. Bagian atas merupakan nama dari *class*. Bagian tengah merupakan struktur dari *class* (atribut) dan bagian bawah merupakan sifat dari *class* (operasi).

Diagram class digunakan untuk menjembatani proses analisis dan proses desain yang akan dilakukan. Setelah menentukan elemen-elemen *use case*, tahap selanjutnya adalah menganalisis *use case* untuk mengidentifikasikan *class* yang terlibat dan menentukan atribut dari tiap-tiap *class* tersebut. *Class* dan atribut tersebut akan digunakan dalam proses desain.

Untuk *server* sendiri memiliki 2 paket *class*, yaitu paket *class gui* dan *protocols*. Untuk class *protocols* ini merupakan *class* pelengkap yang diperlukan dalam *class server*. Lebih detailnya lihat diagram *class gui* pada Gambar 3.3 dan diagram *class protocols* pada Gambar 3.4.

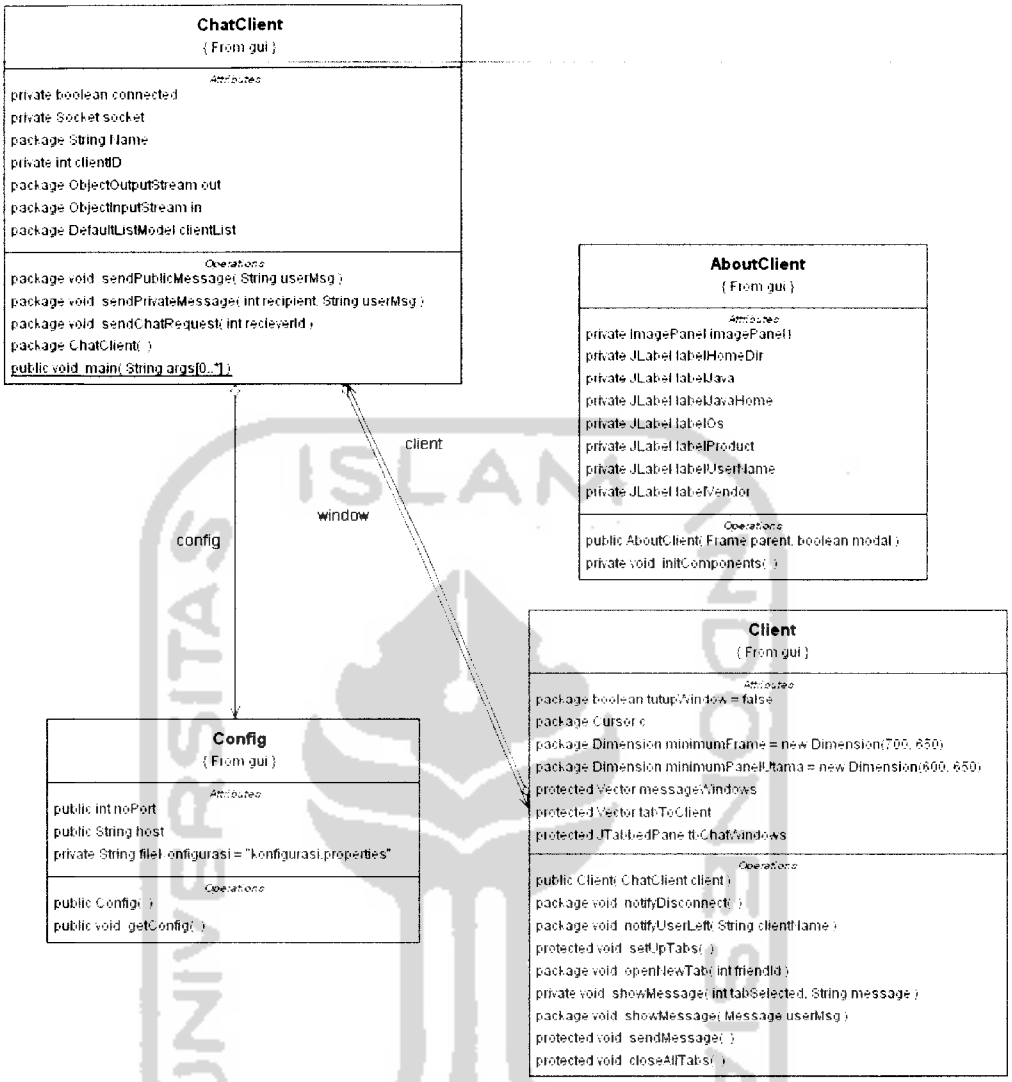


Gambar 3.3 Diagram *class gui Server*

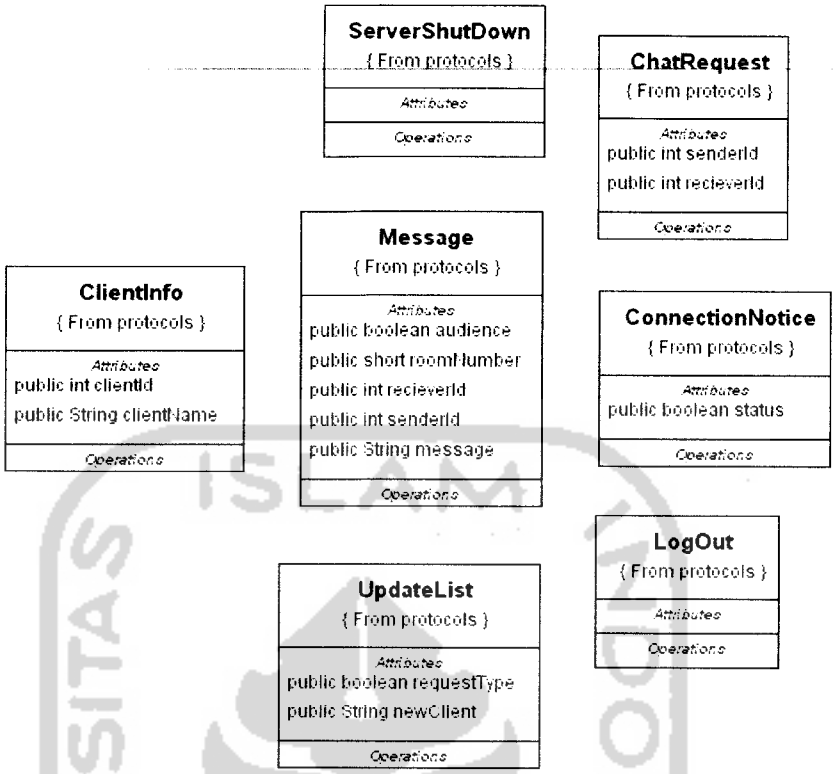


Gambar 3.4 Diagram *class protocols*

Untuk *client* sendiri juga memiliki 2 paket *class*, yaitu paket *gui* dan *protocols*. Untuk class *protocols* ini merupakan *class* pelengkap yang juga diperlukan dalam *class gui*. Lebih detailnya lihat pada Gambar 3.5 dan diagram *class protocols* pada Gambar 3.6.



Gambar 3.5 Diagram *class gui client*



Gambar 3.6 Diagram *class* protocols

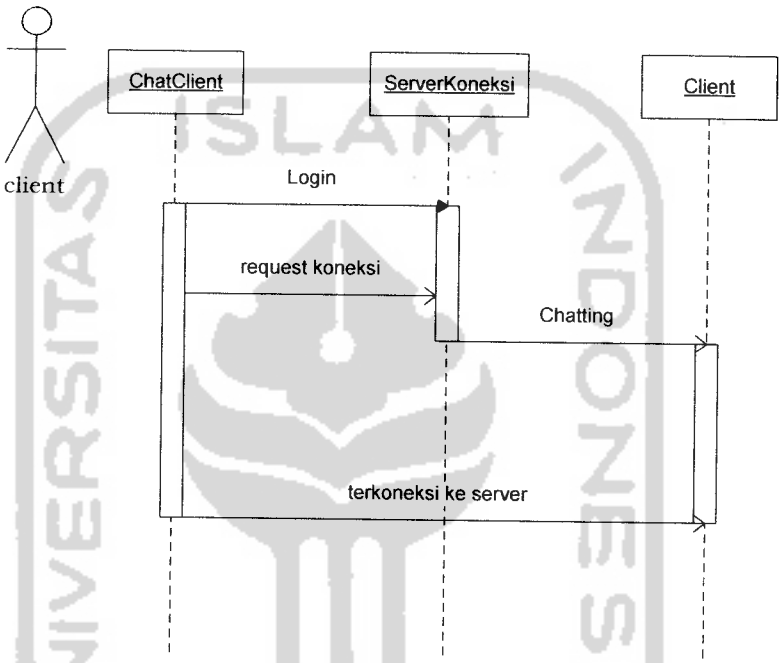
3.2.2.3 *Sequence Diagram*

Diagram *sequence* adalah *interaction* diagram yang memperlihatkan event-event yang berurutan sepanjang berjalannya waktu. Diagram *sequence* menunjukkan interaksi yang terjadi antar obyek didalam dan sekitar (termasuk pengguna, *display*, dan sebagainya) berupa *message* yang digambarkan terhadap waktu. Diagram *sequence* terdiri atas dimensi vertikal (waktu) dan dimensi horisontal (obyek-obyek yang terkait).

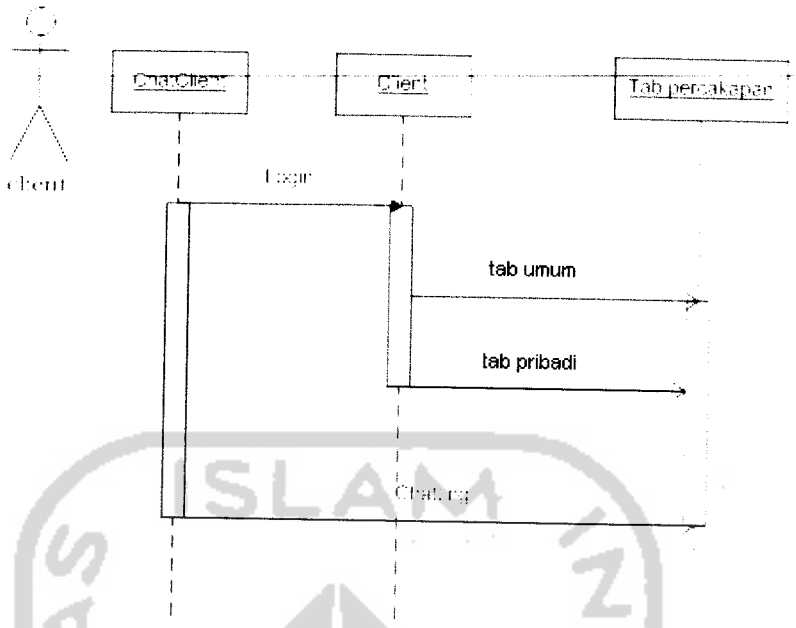
Diagram *sequence* biasa digunakan untuk menggambarkan skenario atau rangkaian langkah-langkah yang dilakukan sebagai respon dari sebuah *event* untuk menghasilkan *output* tertentu. Diawali dari apa yang men-*tigger* aktivitas tersebut, proses atau perubahan apa saja yang terjadi secara *internal* dan *output* apa yang akan dihasilkan. Masing-masing obyek, termasuk aktor memiliki *lifeline* vertikal.

Message digambarkan sebagai garis berpanah dari satu obyek ke obyek lainya. Pada fase desain berikutnya, *message* akan dipetakan menjadi operasi/metode dari *class*. Activation bar menunjukkan lamanya eksekusi sebuah proses, biasanya diawali dengan diterimanya sebuah *message*.

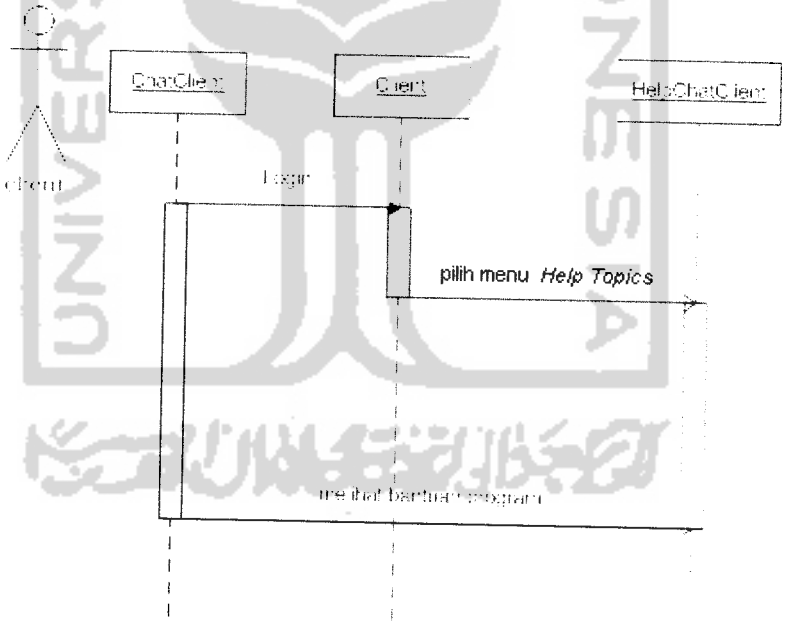
Dari tahapan analisis kebutuhan yang dilakukan sebelumnya maka dibentuk beberapa diagram sequence untuk menunjukkan urutan proses dari masing-masing *use case*.



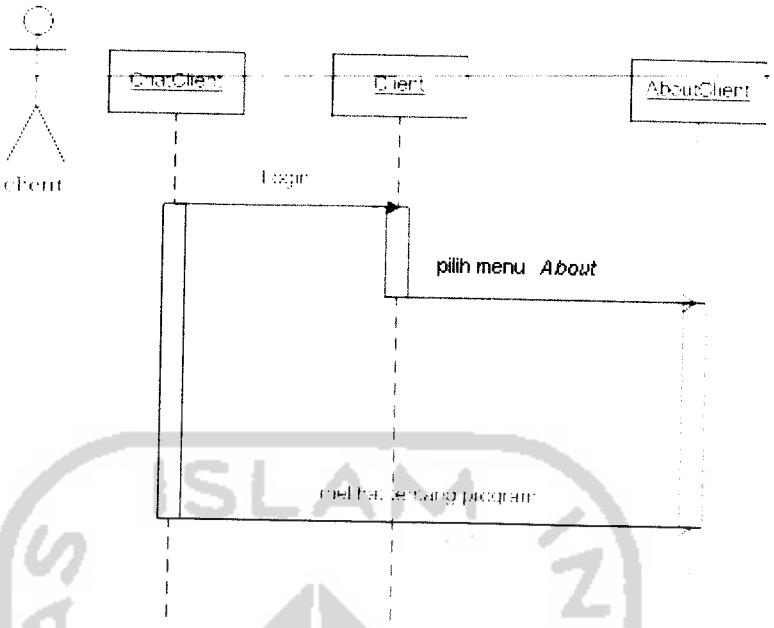
Gambar 3.7 Diagram Sequence koneksi *client* ke *server*



Gambar 3.8 Diagram Sequence *client* Chatting



Gambar 3.9 Diagram Sequence *client* untuk melihat bantuan program



Gambar 3.10 Diagram Sequence *client* untuk melihat tentang program

3.2.2.4 Perancangan Antarmuka

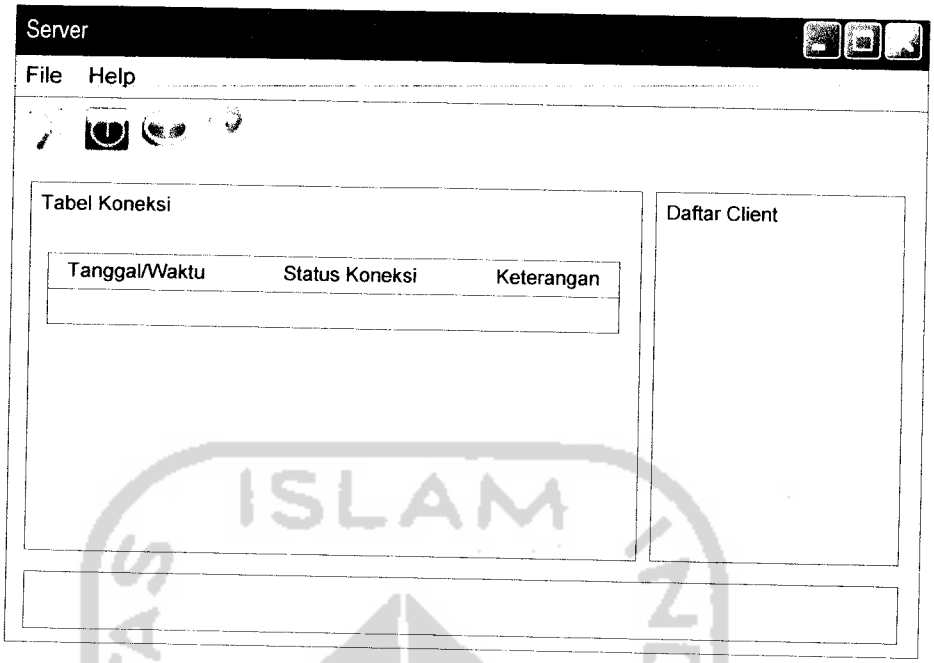
Rancangan antarmuka untuk aplikasi pengelola game adalah sebagai berikut :

1. Rancangan tampilan untuk menu utama *Server*

   Pada tampilan halaman utama terdapat dua menu yang masing-masing memiliki submenu yaitu :

   a. *File* memiliki dua submenu yaitu Konfigurasi port dan Keluar yang digunakan untuk keluar dari aplikasi.
   b. *Help* memiliki dua submenu yaitu *Help Topics* dan *About*.
   c. Di sebelah kiri terdapat table untuk menampilkan informasi koneksi client ke server.
   d. Dan kotak yang di sebelah kanan untuk menampilkan daftar *client* yang terkoneksi saat itu.

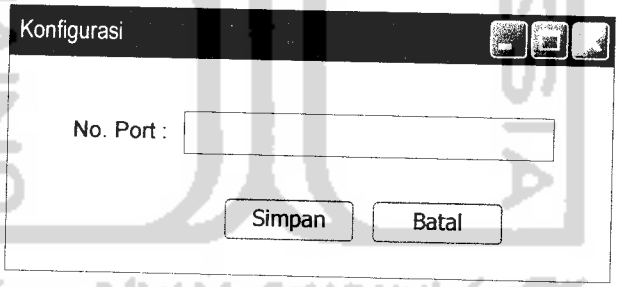
   Rancangan tampilan halaman utama aplikasi server dapat dilihat pada gambar 3.11.



Gambar 3.11 Rancangan tampilan halaman utama *Server*

2. Rancangan tampilan menu konfigurasi *server*

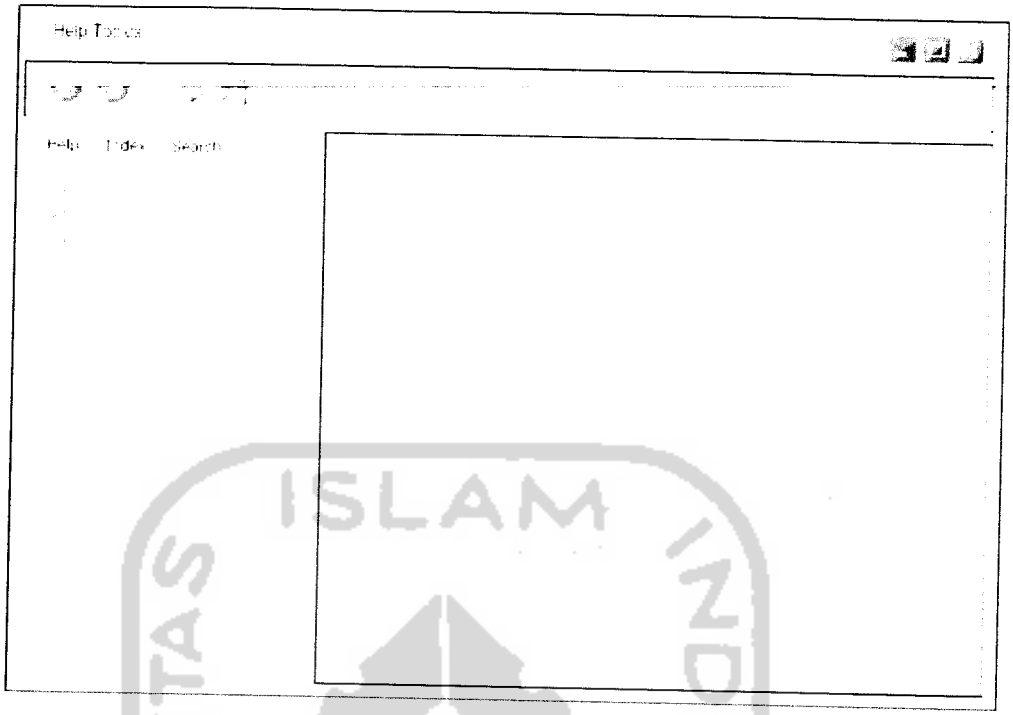
Pada menu konfigurasi *server* diperlukan untuk menentukan nomor port yang akan digunakan dalam koneksi ke jaringan. Seperti pada gambar 3.12.



Gambar 3.12 Rancangan menu konfigurasi *Server*

3. Rancangan tampilan *form Help Topics Server*

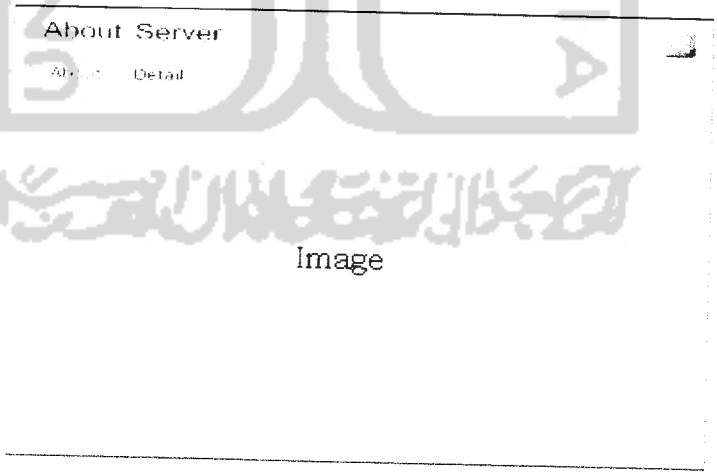
Form Help Server ini berisi tentang bantuan cara menggunakan program *Server*. Tampilan ini dapat dilihat pada gambar 3.13.



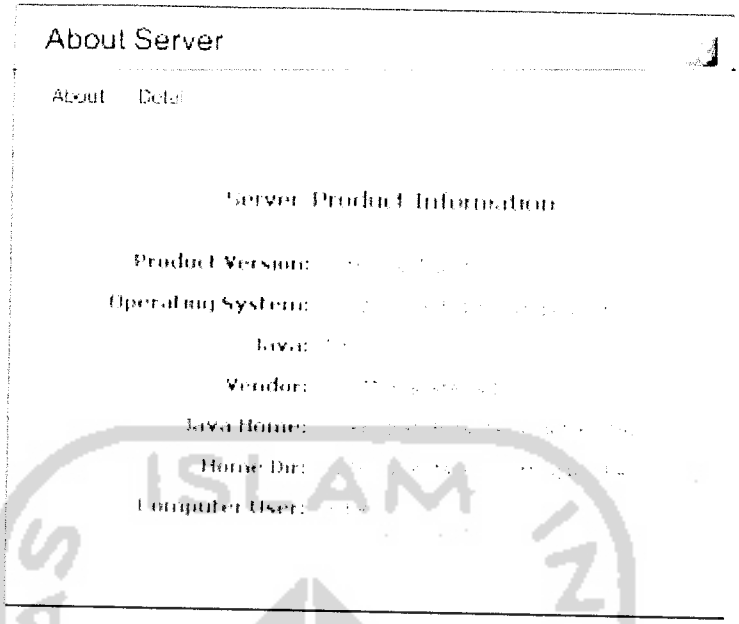
Gambar 3.13 Rancangan *Form Help Server*

4. Rancangan tampilan *form About Server*

Form About Server berisi tentang identitas pembuat program. Dan pada form *Detail* ini menampilkan informasi tentang sistem yang digunakan pada komputer yang digunakan. Lebih jelas dapat di lihat pada Gambar 3.14 dan Gambar 3.15.



Gambar 3.14 Rancangan *Form About Server*



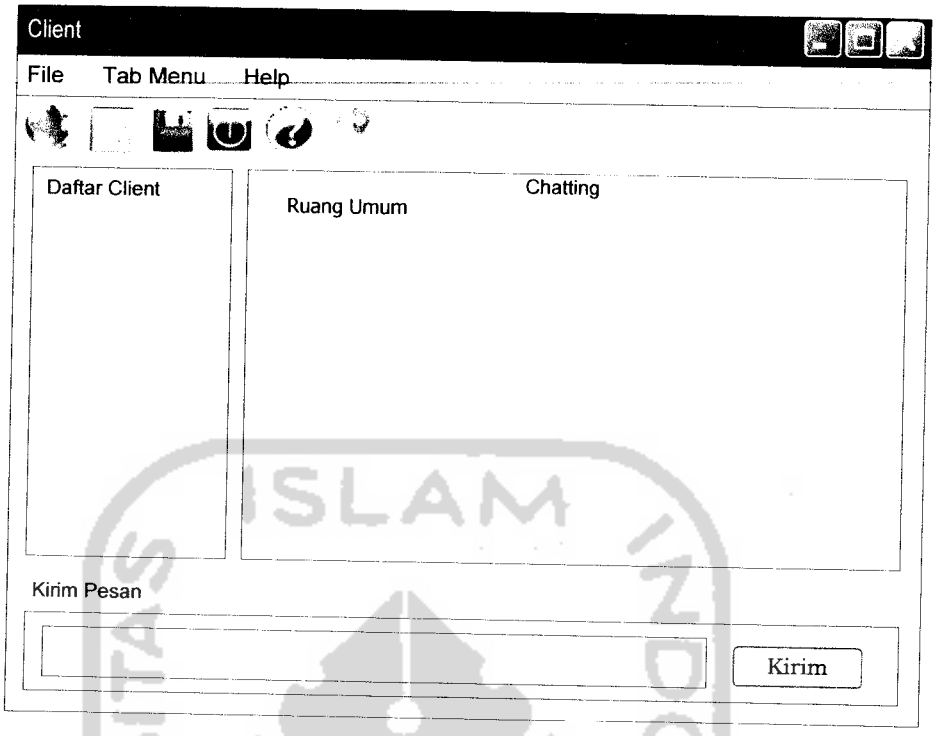
Gambar 3.15 Rancangan *form* informasi sistem komputer

5. Rancangan tampilan Halaman utama *Client*

Pada tampilan menu utama terdapat 2 submenu yang masing-masing memiliki submenu yaitu :

a. *File* memiliki 4 submenu. Login, Logout, Simpan percakapan dan Keluar.
b. Tab Menu untuk menutup satu atau semua tab pada ruang chatting
c. *Help* memiliki dua submenu yaitu *Help Topics* dan *About*.

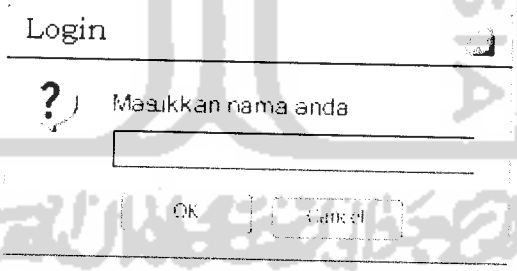
Rancangan halaman utama *client* dapat dilihat pada gambar 3.16.



Gambar 3.16 Rancangan *Form* Halaman utama *Client*

6. Rancangan tampilan input nama pengguna

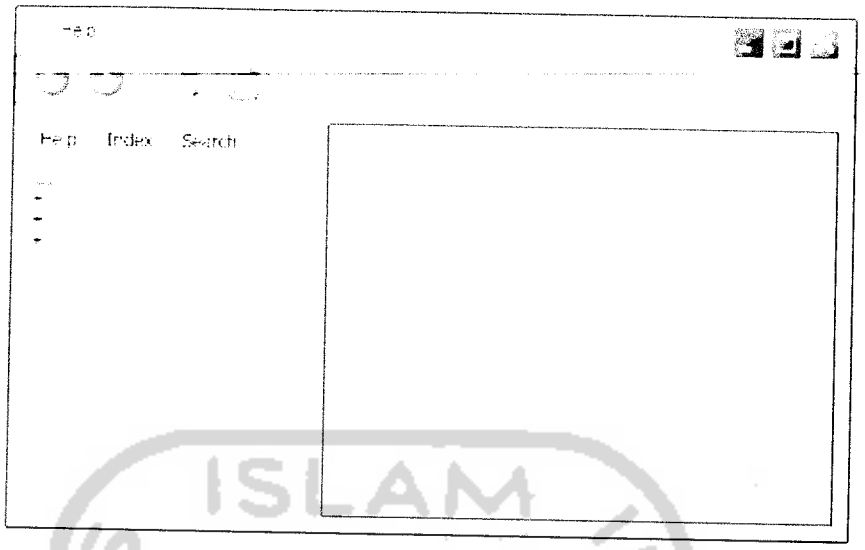
Sebelum pengguna terkoneksi pada server, pengguna harus Login terlebih dahulu dengan mengisikan nama. Seperti pada Gambar 3.17.



Gambar 3.17 Rancangan *Input* menu Login

7. Rancangan tampilan *form Help Client*

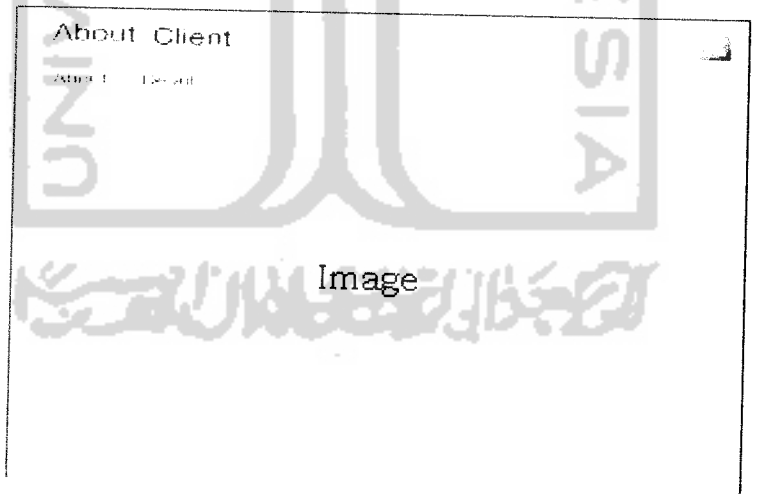
Form Help Client ini berisi tentang pengenalan aplikasi dan cara penggunaannya. Tampilan ini dapat dilihat pada gambar 3.18.



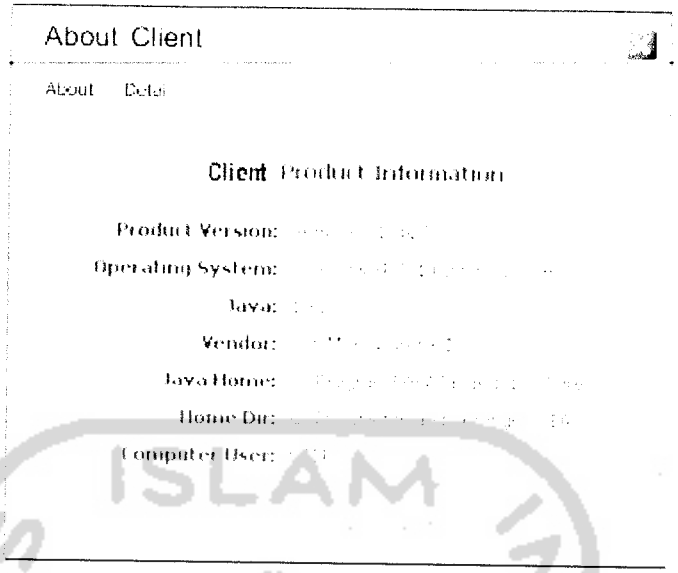
Gambar 3.18 Rancangan *form Help Client*

8. Rancangan tampilan *form About Client*

Form About Client berisi tentang identitas pembuat program. Dan pada form *Detail* ini menampilkan informasi tentang sistem yang digunakan pada komputer yang digunakan. Lebih jelas dapat di lihat pada Gambar 3.19 dan Gambar 3.20.



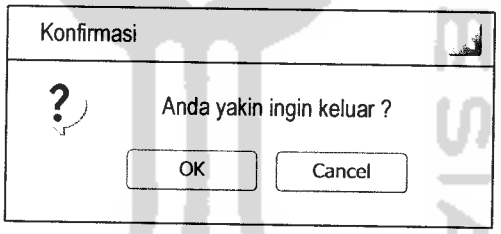
Gambar 3.19 Rancangan *Form About Client*



Gambar 3.20 Rancangan *form* informasi sistem komputer

9. Rancangan pesan konfirmasi

Rancangan form pesan konfirmasi keluar dari program dapat dilihat pada Gambar 3.21.



Gambar 3.21 Rancangan pesan konfirmasi

3.3 Implementasi

3.3.1 Batasan Implementasi

Untuk dapat mengimplementasikan perancangan sistem yang telah dilakukan diperlukan beberapa hal yaitu perangkat keras, perangkat lunak, dan antar muka.

3.3.1.1 Asumsi – Asumsi baru

Dalam proses pembuatan dan pengembangan aplikasi *chatting* ini banyak ditemui asumsi baru yang lebih beragam dari batasan yang telah direncanakan, antara lain :

1. Pengguna diharuskan memasukkan nama untuk melakukan koneksi ke server.
2. Pada program server dibutuhkan konfigurasi server yang digunakan untuk menentukan nomor port yang akan digunakan dalam koneksi.

3.3.1.2 Perangkat Keras yang Dibutuhkan

Perangkat keras yang dibutuhkan adalah :

a. Prosesor AMD Athlon XP 2400+ 2.00 GHz
b. RAM minimal 256 MB
c. *Harddisk* dengan kapasitas 40 GB atau lebih
d. VGA dan monitor dengan resolusi minimal 800 x 600 piksel.
e. *Mouse*
f. *Keyboard*

3.3.1.3 Perangkat Lunak yang Dibutuhkan

Perangkat lunak yang digunakan dalam pengembangan sistem adalah :

a. Sistem Operasi : Microsoft Windows Xp
b. *Development tool* : NetBeans 5.5

3.3.1.4 Bahasa dan *Compiler* yang dipakai

Bahasa yang digunakan adalah java untuk pemrograman, sedangkan editor yang dipilih adalah NetBeans 5.5, karena memiliki kelebihan dalam kemampuannya membantu mengimplementasikan sistem yaitu pendekatan pemrograman yang bersifat visual, dan kecepatan serta kehandalannya ketika dioperasikan.

3.3.2 Implementasi Sistem

Implementasi sistem merupakan tahap dimana sistem mampu diaplikasikan dalam keadaan yang sesungguhnya. Dari implementasi ini akan diketahui apakah sistem yang dibuat dapat berjalan dengan baik atau tidak dan menghasilkan output yang sesuai dengan perancangan yang ada.

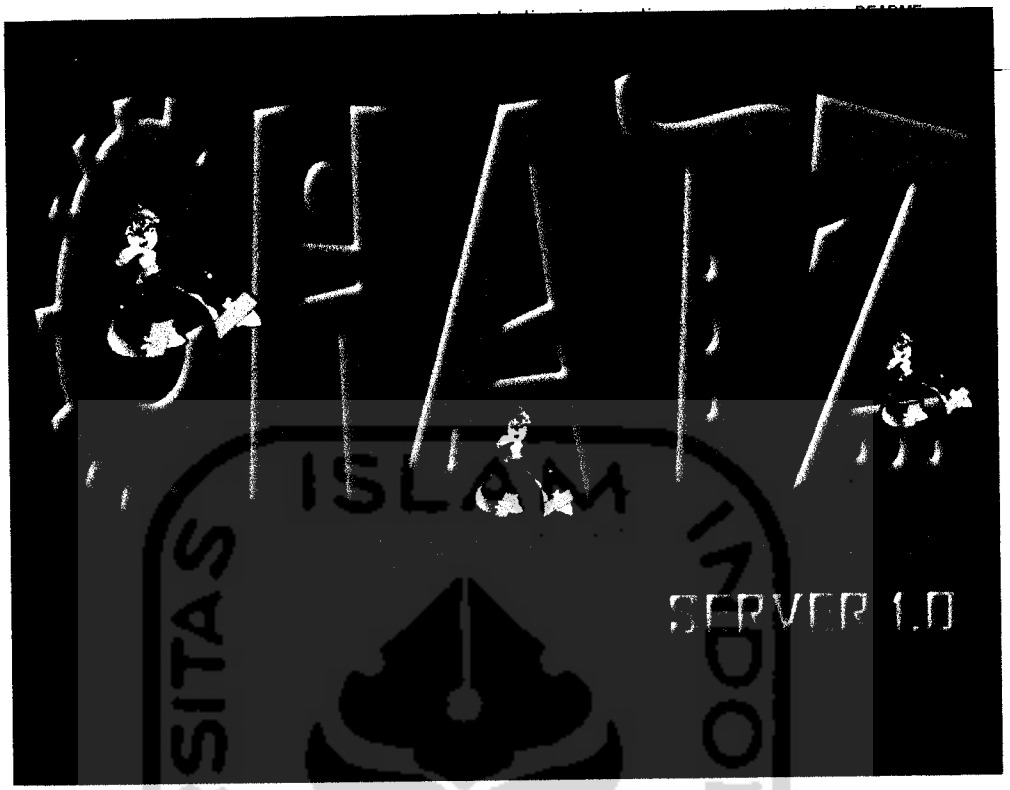
3.3.3 Implementasi Antar muka

3.3.3.1 Tampilan Halaman Utama *Server*

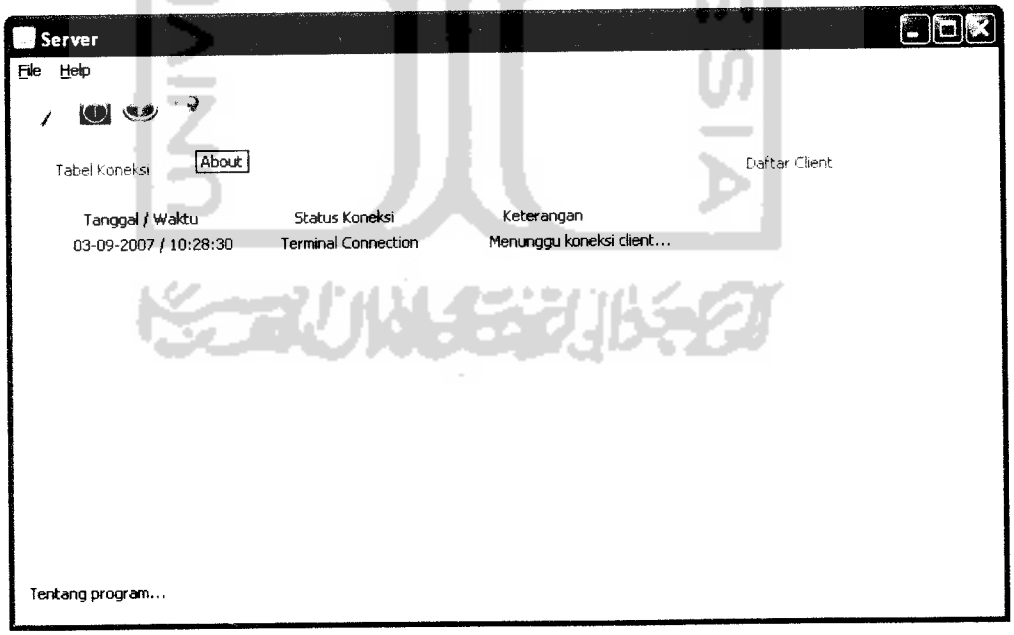
Pada tampilan menu utama terdapat dua menu yang masing-masing memiliki submenu yaitu :

a. *File* memiliki dua submenu yaitu Konfigurasi port dan Keluar yang digunakan untuk keluar dari aplikasi.
b. *Help* memiliki dua submenu yaitu *Help Topics* dan *About*.
c. Di sebelah kiri terdapat table untuk menampilkan informasi koneksi client ke server.
d. Dan kotak yang di sebelah kanan untuk menampilkan daftar *client* yang terkoneksi saat itu.

Dan tampilan halaman utama server dapat dilihat pada gambar 3.23.

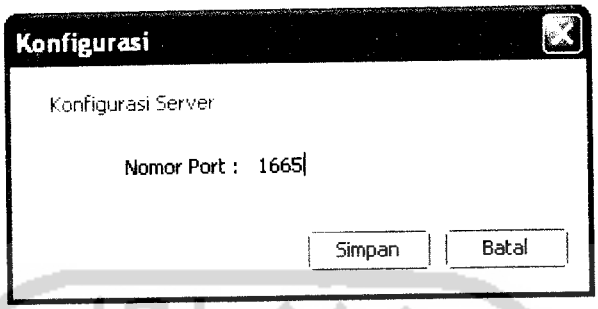


Gambar 3.22 Halaman progres *Server*



Gambar 3.23 Halaman utama *Server*

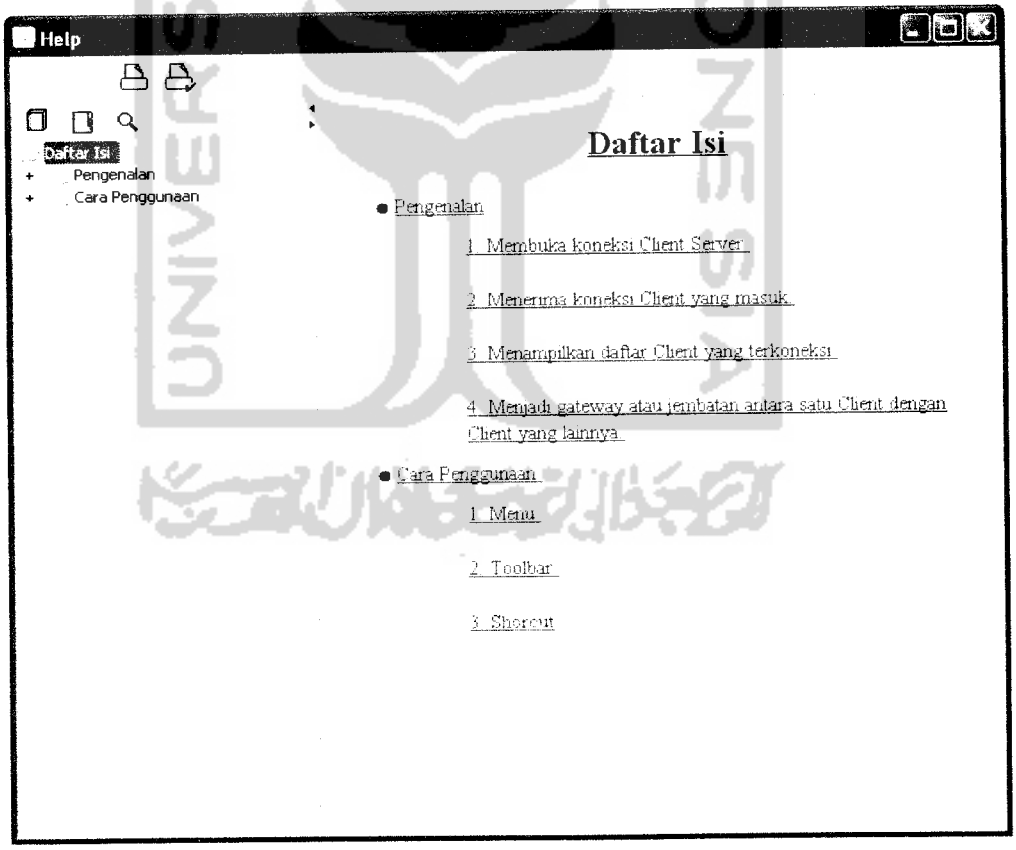
Pada menu konfigurasi *server* digunakan untuk menentukan nomor port yang akan digunakan dalam koneksi ke jaringan. Seperti pada gambar 3.24.



Gambar 3.24 Menu konfigurasi *Server*

3.3.3.2 Tampilan *form Help Topics Server*

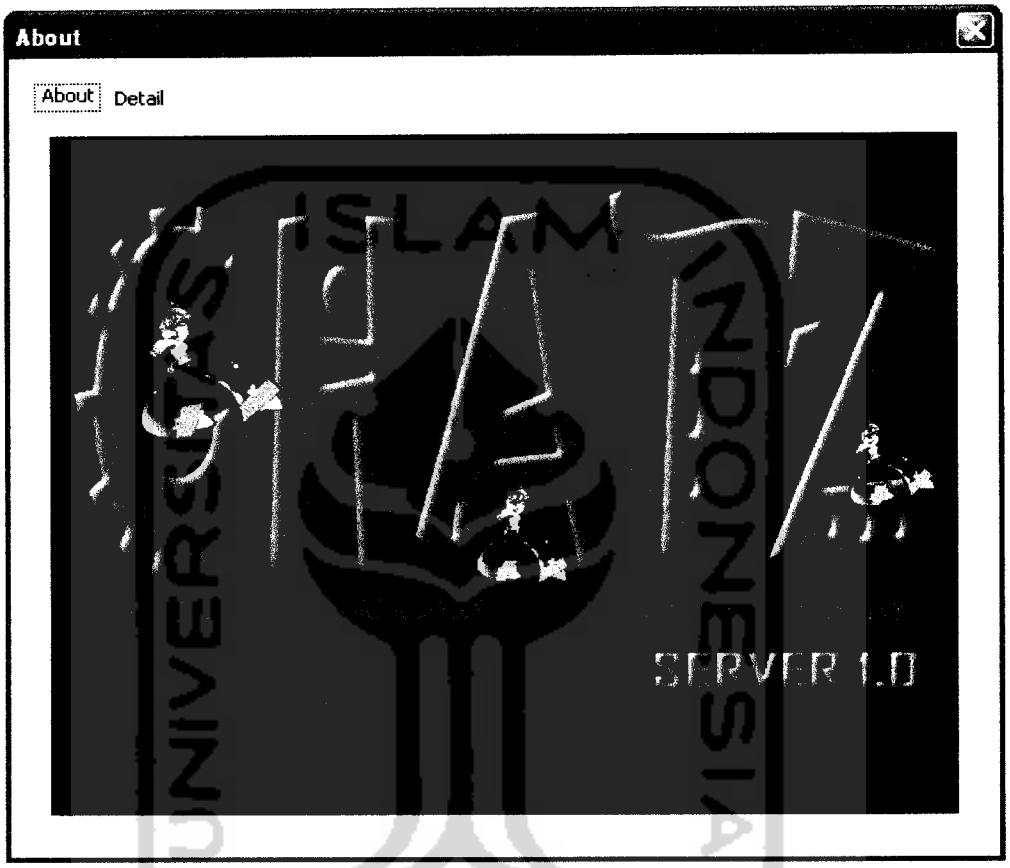
Form ini berisikan pengenalan program *Server*. Tampilannya dapat dilihat pada gambar 3.25.



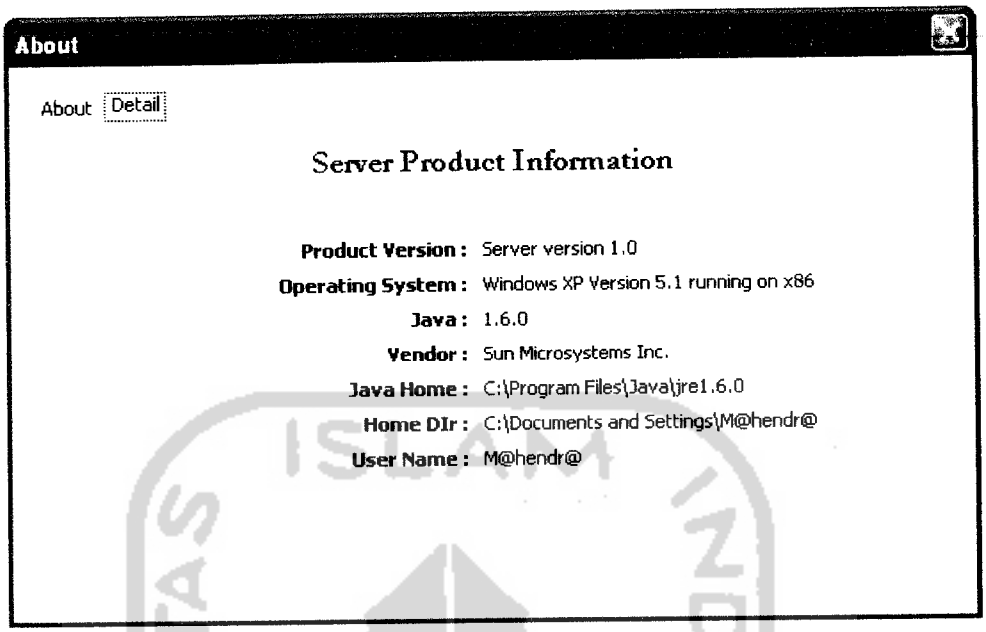
Gambar 3.25 Tampilan *form Help Topics Server*

3.3.3.3 Tampilan *form About Server*

Form About Server menampilkan informasi identitas pembuat program, dan pada halaman detail menampilkan informasi sistem pada komputer yang digunakan. Lebih jelas dapat dilihat pada Gambar 3.26, dan Gambar 3.27.



Gambar 3.26 Tampilan *form About Server*

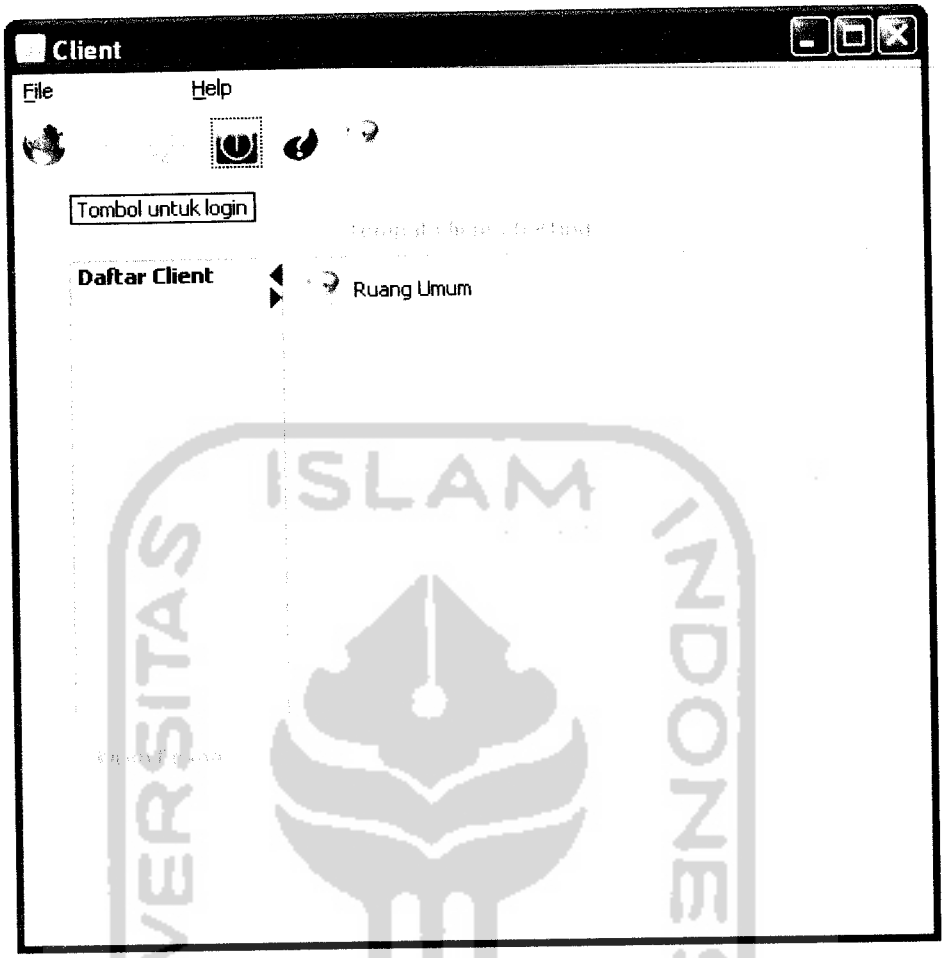


Gambar 3.27 Tampilan informasi sistem pada computer *server*

3.3.3.4 Tampilan Menu Client

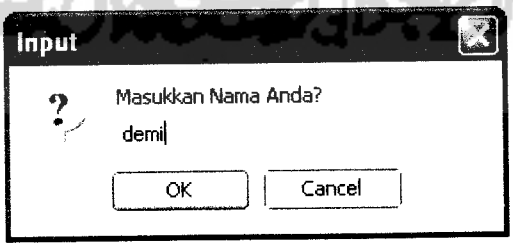
Pada tampilan menu utama terdapat 3 submenu yang masing-masing memiliki submenu yaitu :

a. *File* memiliki 4 submenu. Login, Logout, Simpan percakapan dan Keluar.
b. Tab Menu untuk menutup satu atau semua tab pada ruang chatting
c. *Help* memiliki dua submenu yaitu *Help Topics* dan *About*. Lihat gambar 3.28



Gambar 3.28 Tampilan halaman utama program *Client*

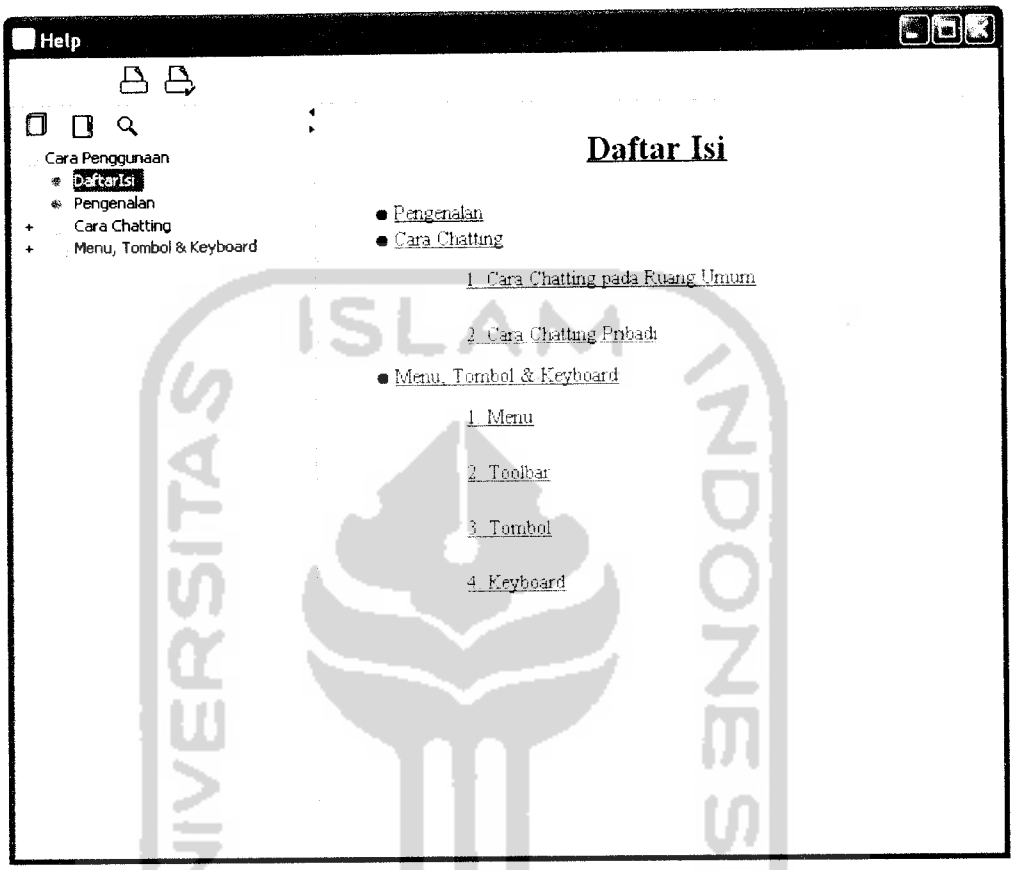
Pada menu *client* ini untuk terkoneksi dengan *server* pengguna diharuskan mengetikkan nama, untuk ditampilkan pada daftar *Client*. Dapat dilihat pada gambar 3.29.



Gambar 3.29 Tampilan Menu *Login*

3.3.3.5 Tampilan menu *Help Topics* pada *Client*

Menu ini berisi pengenalan tentang program, dan cara-cara bagaimana menggunakan program tersebut. Tampilan dapat dilihat pada Gambar 3.30.

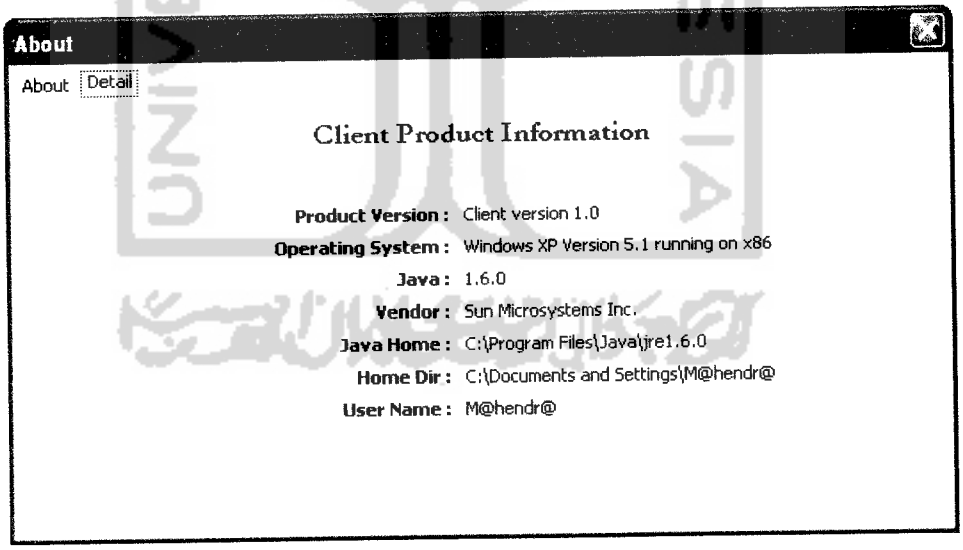


Gambar 3.30 Tampilan menu *Help Topics* pada *Client*

3.3.3.6 Tampilan Menu *About* pada *Client*

Selain ditampilkan tentang pembuat *software*, menu ini juga menampilkan informasi produk dimana dilengkapi dengan informasi *software* yang sedang digunakan. Tampilan dapat dilihat pada Gambar 3.31 dan Gambar 3.32.

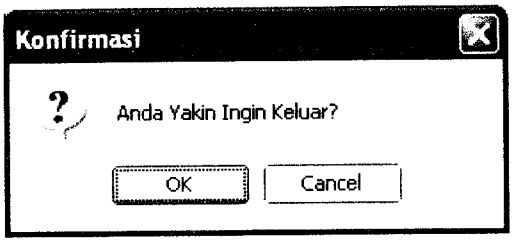
Gambar 3.31 Tampilan menu *About* pada *Client*



Gambar 3.32 Tampilan informasi sistem pada komputer *client*

3.3.3.7 Tampilan Pesan Konfirmasi

Tampilan pesan konfirmasi untuk keluar dari program dapat dilihat pada Gambar 3.33.



Gambar 3.33 Tampilan pesan konfirmasi

3.3.4 Implementasi Prosedural

Pada aplikasi Chatting ini terdapat beberapa prosedur yang digunakan dalam pemrograman, berikut adalah penggalan dari prosedur tersebut :

1. Prosedur koneksi *client* ke *server*

   Prosedur ini digunakan untuk mengatur koneksi client ke server, server bertugas untuk menunggu permintaan koneksi dan menerima koneksi dari client.

```
public class ServerKoneksi {
  ServerSocket server;
  int onlineUsers;
  Vector handlers;
  ServerGui window;
  SettingKonfigurasi sk;
  PesanWaktu pw;
  TampilPesan tp1, tp2, tp3;
  public boolean tutup = false;
  public class ChatHandler implements Runnable {
    protected ClientInfo clientInfo = new ClientInfo();
    protected Socket socket;
    protected ObjectInputStream in;
    protected ObjectOutputStream out;
    //menciptakan objek Thread
    protected Thread listener;
```

```
public ChatHandler( int clientID, Socket socket ) {
    clientInfo.clientId = clientID;
    this.socket = socket;
}
protected void listenForClients( ServerSocket server ) {
        pw.tampilkanWaktu();
        tp1.setTglWaktu(pw.tgl_bln_thnSekarang);
        tp1.setStatus("Terminal Connection");
        tp1.setKeterangan("Menunggu koneksi client...");
        window.list.add(tp1);
        window.updateTable();
        int clientID = 0;
```

2. Prosedur mengatur konfigurasi *server*

Prosedur ini digunakan untuk menentukan nomor port yang akan digunakan dalam koneksi, dan disimpan dalam file konfigurasi.

```
public class SettingKonfigurasi {

    public int noPort;
    private String fileKonfigurasi=konfigurasi.properties";

    public SettingKonfigurasi() {
    }

        public void setConfig(String noPort){

            Properties p = new Properties();
            p.setProperty("datasource.noPort",noPort);
```

3. Prosedur untuk menyampaikan pesan ke semua *client*

Prosedur ini digunakan untuk memberitahukan ke semua client apabila ada client baru yang terkoneksi, kemudian ditambahkan ke daftar client.

```
window.clientList.addElement( clientInfo.clientName);
out.writeObject( window.clientList);
```

```
UpdateList newClient = new UpdateList();
newClient.requestType = true;
newClient.newClient = clientInfo.clientName;
broadcast( newClient );

onlineUsers++;
```

4. Prosedur *client* untuk memutuskan koneksi dari *server*

Prosedur ini digunakan untuk mengatur client yang memutuskan koneksi dari server kemudian memberitahukan ke semua client.

```
public synchronized void stop() {
    if ( listener != null ) {
        try {
            listener.interrupt();
            listener = null;

            handlers.removeElement( this );
            window.clientList.removeElement(
            clientInfo.clientName );
            UpdateList newClient = new UpdateList();
            newClient.requestType = false;
            newClient.newClient = clientInfo.clientName;
            broadcast( newClient );

            out.close();
            socket.close();

            tp3 = new TampilPesan();
            pw.tampilkanWaktu();
            tp3.setTglWaktu(pw.tgl_bln_thnSekarang);
            tp3.setStatus("Terminal Connection");
            tp3.setKeterangan("Koneksi Client dengan ID " +
            clientInfo.clientId +
            " dan user name : " + clientInfo.clientName + "
            ditutup..." );
            window.list.add(tp3);
            window.updateTable();
```

BAB IV
HASIL DAN PEMBAHASAN

Hasil dan pembahasan merupakan proses pengujian terhadap sistem. Dari hasil pengujian akan diketahui apakah fungsi-fungsi yang ada dalam sistem ini dapat berjalan dengan baik dan memenuhi kebutuhan. Pengujian dilakukan dengan menjalankan proses-proses yang ada dalam sistem dengan memasukkan data sesuai kebutuhan.

Hasil dari pengujian ini kemudian dianalisis untuk mengetahui sejauh mana program dapat berjalan, apakah sesuai dengan yang diharapkan. Kekurangan-kekurangan yang ada akan menjadi masukan untuk kemudian diterapkan pada implementasi program selanjutnya.

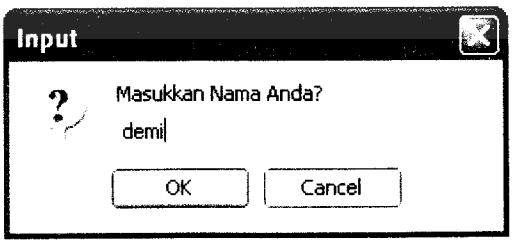
4.1 Pengujian Sistem

Pengujian sistem dilakukan dengan menjalankan sistem sesuai dengan konfigurasi dan data yang dibutuhkan sehingga dapat dilihat apakah sistem berjalan dengan baik sesuai dengan tujuan yang diinginkan.

4.1.1 Koneksi *Client* ke *Server*

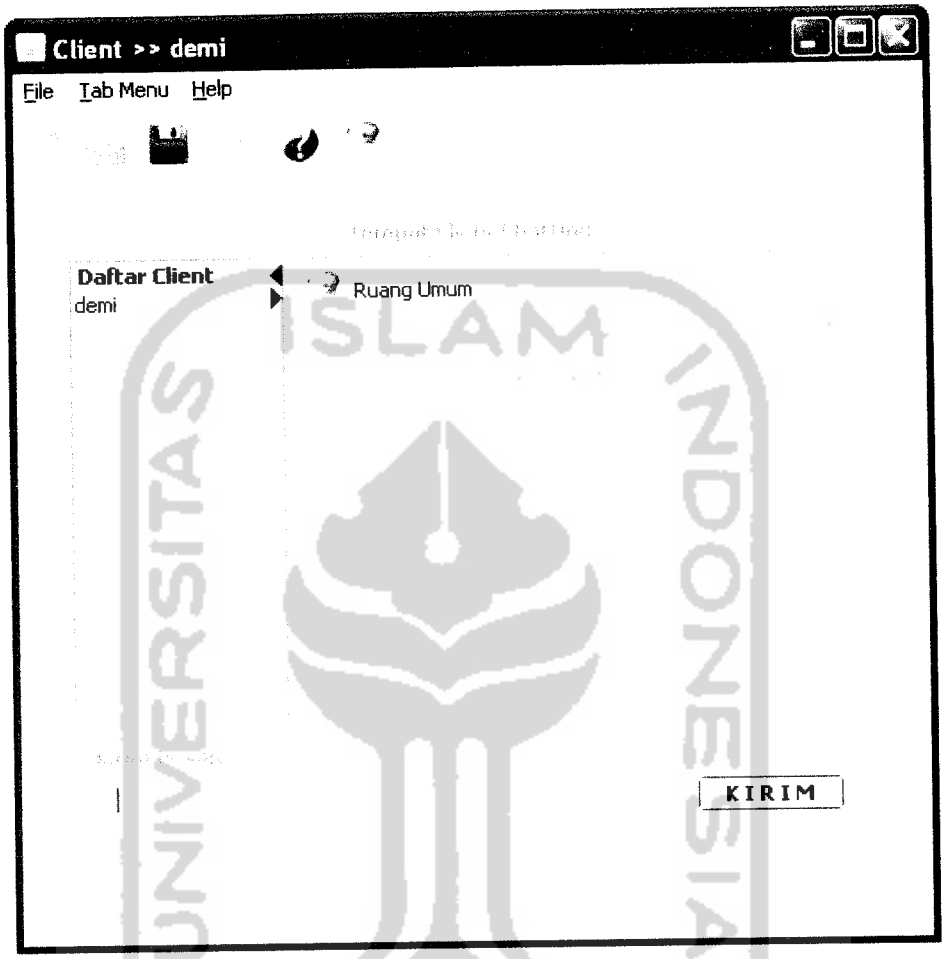
a. Pengujian normal

Pengujian normal pada proses koneksi *client* ke *server* ini untuk membuktikan apakah sistem dapat mengkoneksikan *client* ke *server*. Sebelum proses koneksi, terlebih dahulu pengguna harus Login. Seperti pada Gambar 4.1.



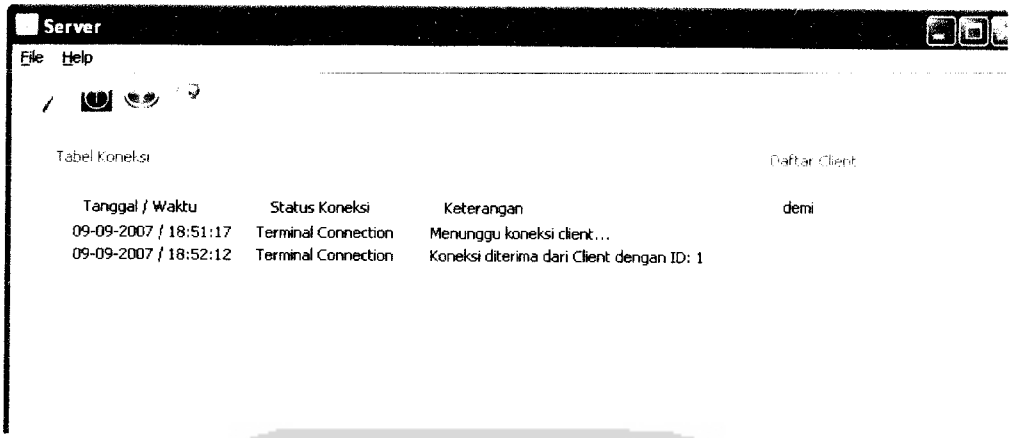
Gambar 4.1 Menu *Login*

Apabila koneksi sukses nama pengguna akan tertera pada menu bar diatas serta pada daftar *Client*, seperti pada Gambar 4.2.



Gambar 4.2 Halaman utama *client*

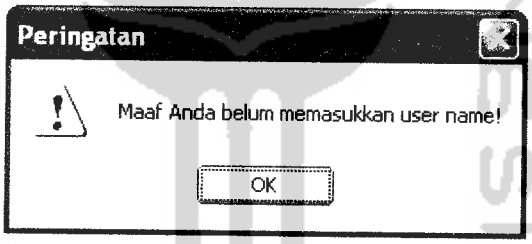
Dan pada halaman *server* akan muncul informasi bahwa *client* baru telah terkoneksi pada jaringan, dan nama pengguna akan tampil pada daftar *client*. Tampilan dapat dilihat pada Gambar 4.3.



Gambar 4.3 Tampilan koneksi *client* ke *server*

b. Pengujian tidak normal

Pada pengujian tidak normal ini untuk membuktikan apakah pesan kesalahan pada sistem berjalan, keadaan ini dijumpai ketika pengguna tidak memasukkan nama pada *form input* menu Login. Pesan kesalahan pada Gambar 4.6.



Gambar 4.4 Pesan kesalahan pada *menu Login*

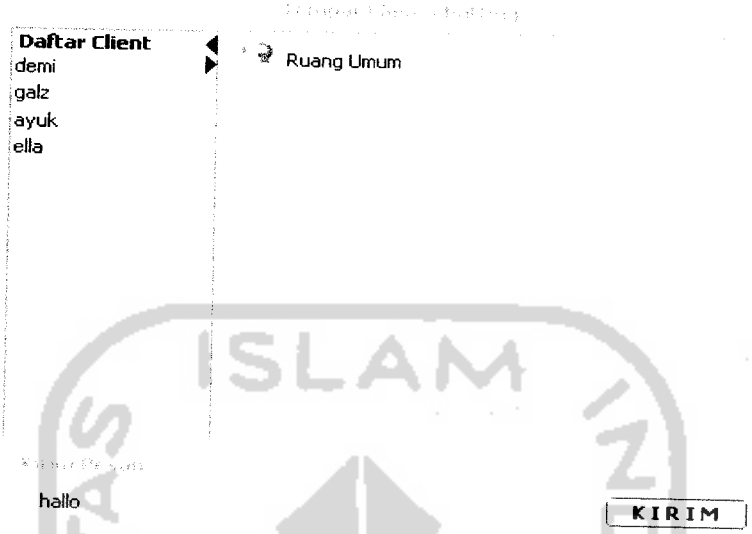
4.1.2 Pengujian proses chatting

Pengujian ini dilakukan untuk membuktikan apakah proses chatting berjalan dengan baik, chatting pada program ini memiliki 2 ruang, yaitu umum dan pribadi.

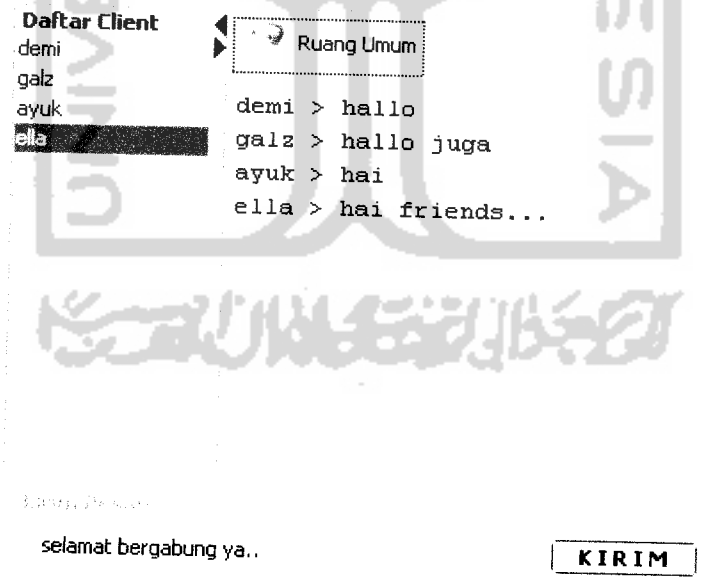
a. Pengujian pada ruang umum

Setelah *client* terkoneksi pada *server*, *client* dapat langsung melakukan *chatting* dengan *client* lain yang saat itu terkoneksi. Dan jika proses pengiriman pesan berhasil, pesan akan tampil pada tab ruang umum dan begitu juga pada tab

ruang umum client yang lain. Seperti yang dapat terlihat pada gambar 4.7, dan pada gambar 4.8.



Gambar 4.5 Tampilan ruang umum pada chatting



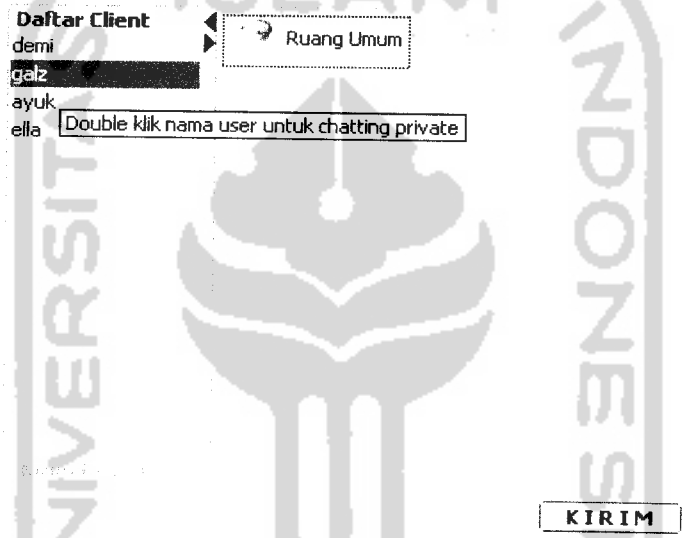
Gambar 4.6 Proses pengiriman pesan berhasil

b. Pengujian pada ruang pribadi

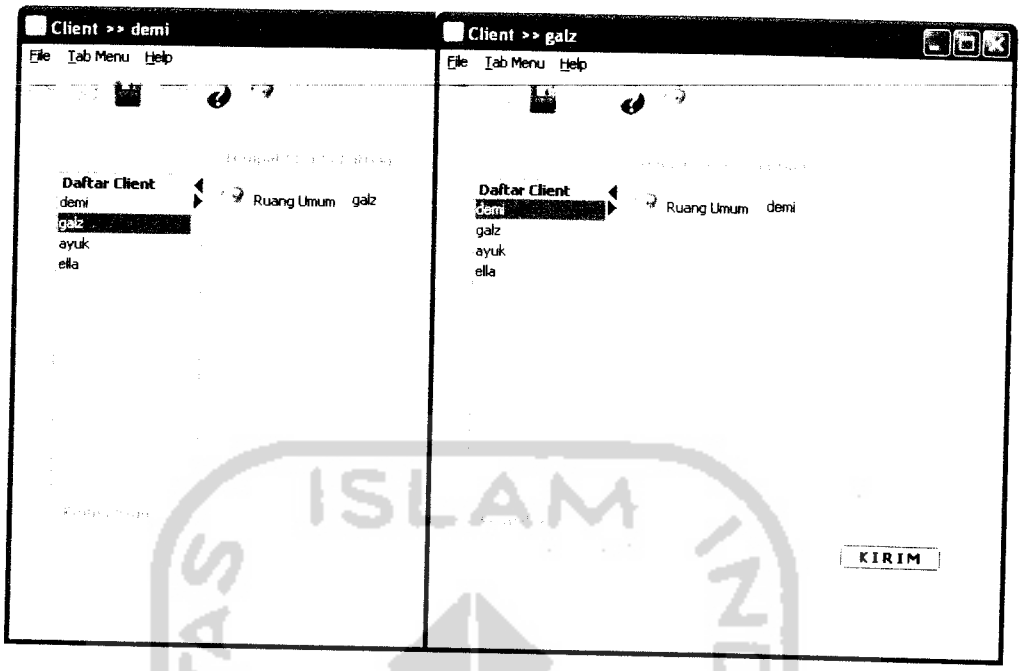
Untuk ruang pribadi pengguna tinggal mengklik dua kali untuk berbicara dengan client lain di ruang pribadi. Seperti yang terlihat pada Gambar 4.9.

Kemudian akan muncul tab baru dengan nama client yang diklik untuk chatting pribadi, begitu juga sebaliknya pada client lawannya. Dapat dilihat pada gambar 4.10.

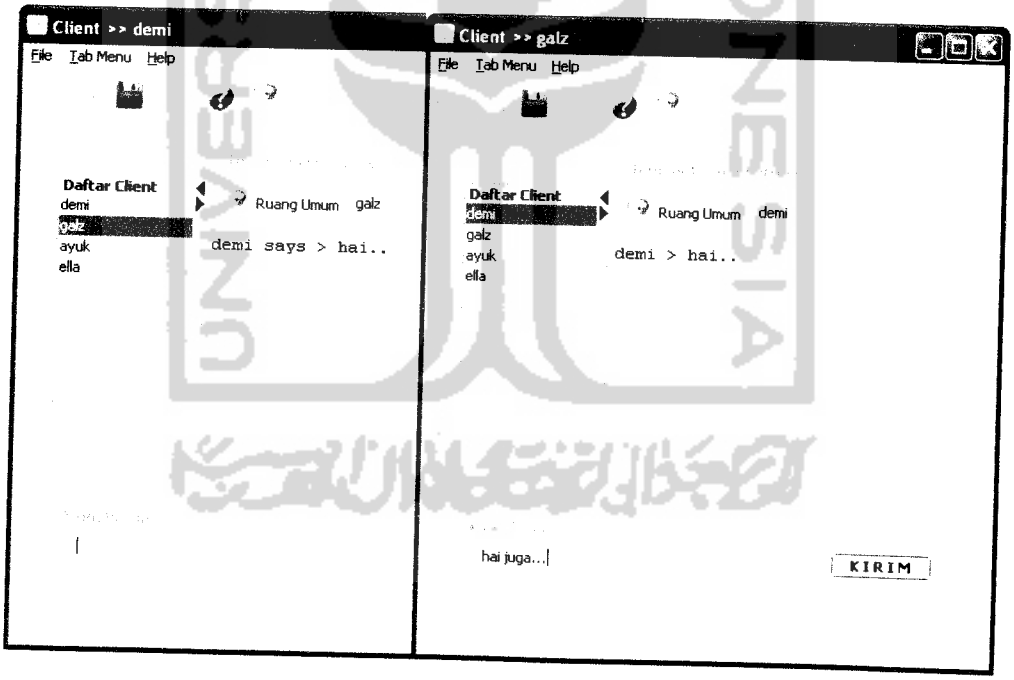
Dan apabila proses tersebut berhasil maka pesan yang dikirim akan tampil di tab ruang pribadi dengan nama client lawan. Seperti yang terlihat pada gambar 4.11.



Gambar 4.7 Informasi untuk *chatting* pribadi dengan *client* lain



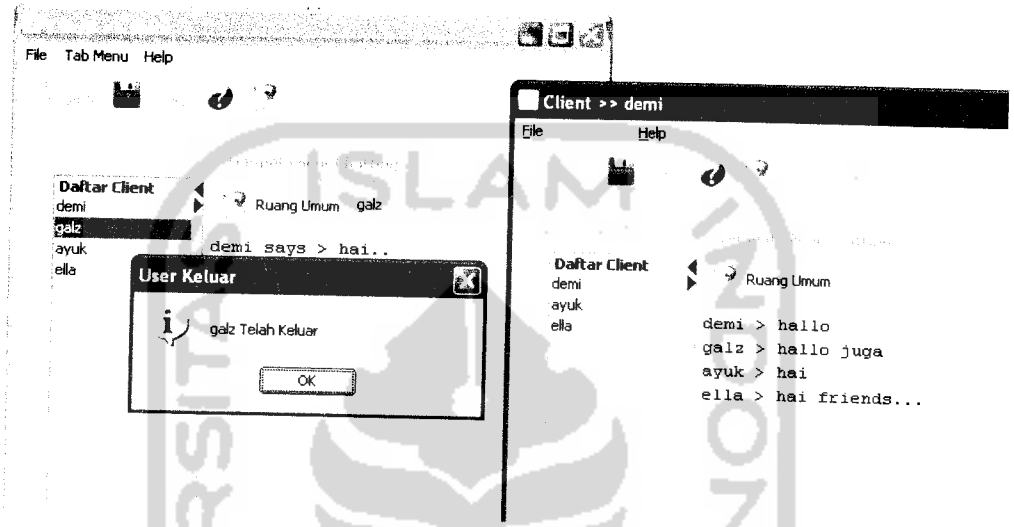
Gambar 4.8 Tampilan ruang pribadi dengan tab baru



Gambar 4.9 Pengiriman pesan pada ruang pribadi berhasil

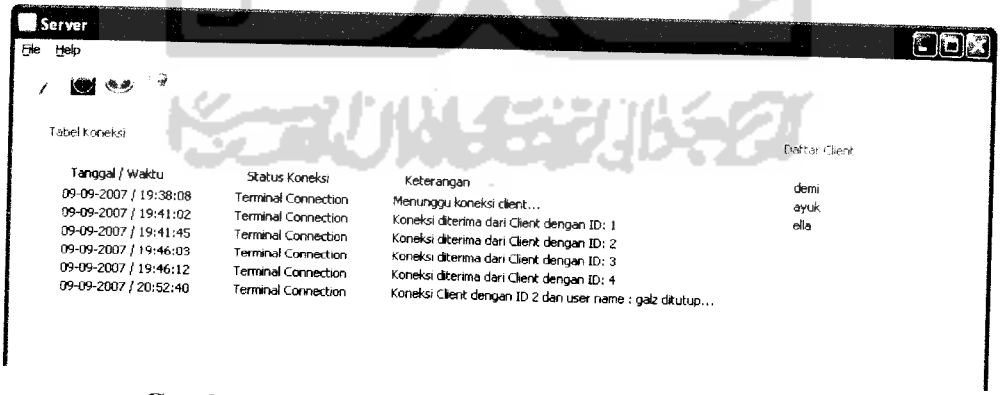
4.1.3 Pengujian penyampaian pesan pada *client*

Pengujian ini untuk membuktikan apabila pada saat itu kedua *client* sedang menggunakan ruang pribadi ketika *chatting* maka, *client* lainnya akan dikirimkan pesan bahwa *client* tersebut telah memutuskan koneksi dengan *server*. Dan secara otomatis *tab chatting* pada ruang pribadi akan tertutup. Seperti yang terlihat pada Gambar 4.12.



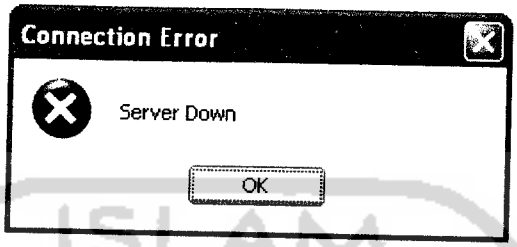
Gambar 4.10 Tampilan pesan *client* telah memutuskan koneksi

Pengujian ini juga dilakukan untuk membuktikan apabila salah satu *client* telah memutuskan koneksi, maka akan ditampilkan pesan ke *server* bahwa *client* tersebut telah memutuskan koneksi. Seperti yang terlihat pada Gambar 4.13.



Gambar 4.11 Tampilan pesan *client* memutuskan koneksi

Selain itu untuk membuktikan bahwa semua aktivitas *client* terhubung pada *server*, jadi apabila *server* memutuskan koneksi maka semua *client* yang berada dalam jaringan akan terputus secara otomatis. Dan *client* akan mendapatkan pesan bahwa koneksi ke *server* telah terputus, pesan tersebut dapat dilihat pada gambar 4.14.



Gambar 4.12 Pesan koneksi *error*

4.2 Analisis Hasil Pengujian

Analisis hasil pengujian merupakan kesimpulan yang dapat diambil dari hasil pengujian yang telah dilakukan. Berdasarkan pengujian tersebut dapat dilihat bahwa aplikasi *server* dapat menampilkan informasi koneksi *client* ke *server*.

Dalam pengujian ini, seluruh *client* terhubung pada satu *server* sehingga pada saat server memutuskan koneksi atau koneksi terputus secara tiba-tiba maka semua client terputus koneksinya dengan *server*.

Kelebihan dari aplikasi ini adalah :

1. Pada aplikasi *server* dapat menampilkan koneksi *client* ke *server*
2. Sistem ini dibuat untuk memudahkan pengguna, karena sistem diatur secara otomatis sehingga tidak terlalu menyulitkan pengguna saat berkomunikasi.

Kekurangan dari sistem yang dibangun ini adalah :

1. Tidak tersedianya fasilitas gambar ekspresi wajah pada client, seperti *smiley*.
2. Pada aplikasi *server* untuk saat ini tidak dapat menampilkan seluruh informasi yang dibutuhkan, dan server tidak dapat ikut *chatting*.

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Dari hasil penulisan tugas akhir ini, diperoleh kesimpulan sebagai berikut:

1. Aplikasi chatting ini terutama pada server dapat menampilkan informasi client yang terhubung pada server.
2. Sistem ini dibuat untuk memudahkan pengguna, karena sistem diatur secara otomatis sehingga tidak terlalu menyulitkan pengguna saat berkomunikasi.
3. Dengan menggunakan metode berorientasi objek untuk perancangan sistem, memudahkan dalam pengembangan sistem secara keseluruhan, sebagai contoh ketika ingin menambahkan suatu fungsi tertentu, cukup dengan menambahkan fungsi tersebut ke dalam *class*. Demikian pula ketika diimplementasikan dalam kode program, dengan menggunakan pemrograman berorientasi objek, maka fungsi tambahan tersebut cukup ditambahkan ke dalam *class* yang berhubungan.
4. Dengan menggunakan bahasa Java sebagai bahasa pemrograman maka aplikasi ini dapat dijalankan di berbagai *platform* sistem operasi seperti Linux, Windows maupun Unix.
5. Penggunaan NetBeans IDE 5.5 untuk mendesain dan menulis kode program membuat tampilan aplikasi ini lebih menarik dan lebih mudah dalam penulisan kode program.

5.2 Saran

Beberapa saran untuk pengembangan dan penelitian selanjutnya sebagai berikut :

1. Pada pengembangan lebih lanjut dari aplikasi yang sudah dibuat memiliki fasilitas untuk menampilkan gambar ekspresi wajah, seperti *smiley*.
2. Pada pengembangan lebih lanjut dari aplikasi yang sudah dibuat memiliki fasilitas yang lebih menarik seperti fasilitas chatting lain yang sudah ada.

DAFTAR PUSTAKA

1. Susanto, Budi, 2003, "Pemrograman Client/Server dengan Java 2", Jakarta : Elex Media Komputindo.

2. Wahana komputer, 2003, "Konsep Jaringan Komputer dan Pengembangannya", Jakarta : Salemba Infotek.

3. Hartati, Sri, G.. Suharto, Herry, B.. Wijono, Soesilo, M., 2007, "Pemrograman GUI Swing Java dengan Netbeans 5", Yogyakarta : ANDI.

4. Fikri, Rijalul. Fuadina Adam, Ipam. Prakoso, Imam., 2005, "Pemrograman JAVA", Yogyakarta : ANDI.

5. Indrajani, S.Kom, MM. Matin, S.Kom, 2004, "Pemrograman Berorientasi Objek dengan Java", Jakarta : Elex Media Komputindo.