BAB III

ANALISIS DAN PERANCANGAN SISTEM

3.1. Analisis Sistem

Analisis sistem merupakan gambaran umum mengenai analisis kebutuhan, yang berupa data dan informasi yang dibutuhkan serta perangkat keras maupun perangkat lunak yang digunakan untuk membangun sistem. Adapun kebutuhan-kebutuhan yang diperlukan untuk membangun dan mengembangkannya, yang meliputi :

3.1.1. Kebutuhan *Input Client*

- Masukan nomor *port* untuk menyambung ke *server*
- Masukan nomor *IP host* untuk menyambung ke *server*
- Masukan *nick* untuk pengenal didalam percakapan
- Masukan kunci yang digunakan enkripsi dan dekripsi yang di input melalui aplikasi server
- Masukan pesan yang ingin dikirim

3.1.2. Kebutuhan *Input Server*

- Masukan nomor *port* yang akan digunakan *server* untuk menerima client
- Masukan kunci enkripsi dekripsi yang nanti akan dikirim ke setiap *client* yang terhubung ke *server*

3.1.3. Kebutuhan Output Client

- Menampilkan daftar nick yang terhubung dengan ke *server*
- Menampilkan pesan masuk yang dikirim *client* lain
- Menampilkan *nick* yang digunakan
- Menampilkan pesan proses sistem
- Tampilan *interface*

3.1.4. Kebutuhan Output Server

- Menampilkan pesan proses sistem

3.1.5. Kebutuhan Perangkat Keras (*Hardware*)

- Notebook dengan processor AMD Athlon II dual core 2.3 GHz
- RAM (Random Access Memory) 2 GB DDR3
- Harddisk dengan kapasitas 160 GB

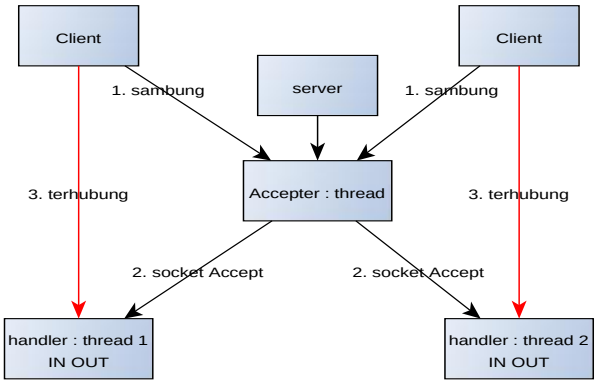
3.1.6. Kebutuhan Perangkat Lunak (*Software*)

- Sistem operasi windows 7 ultimate 32 bit
- Netbean 7.1
- JDK versi 7 update ke 3

3.2. Perancangan Sistem

Perancangan sistem ini berisi mengenai pemodelan konsep, algoritma, serta *flowchart* yang digunakan dalam pembuatan program *server* dan *client* yang akan dibangun, yaitu sebagai berikut :

3.2.1. Konsep Dasar *Server Multi Client*



Gambar 3.1 Konsep *Server Multi Client*

Konsep diatas dapat dijelaskan bahwa setiap *client* yang terhubung akan di *accept* kemudian dibuatkan sebuah *thread* untuk menangani permintaan *client*

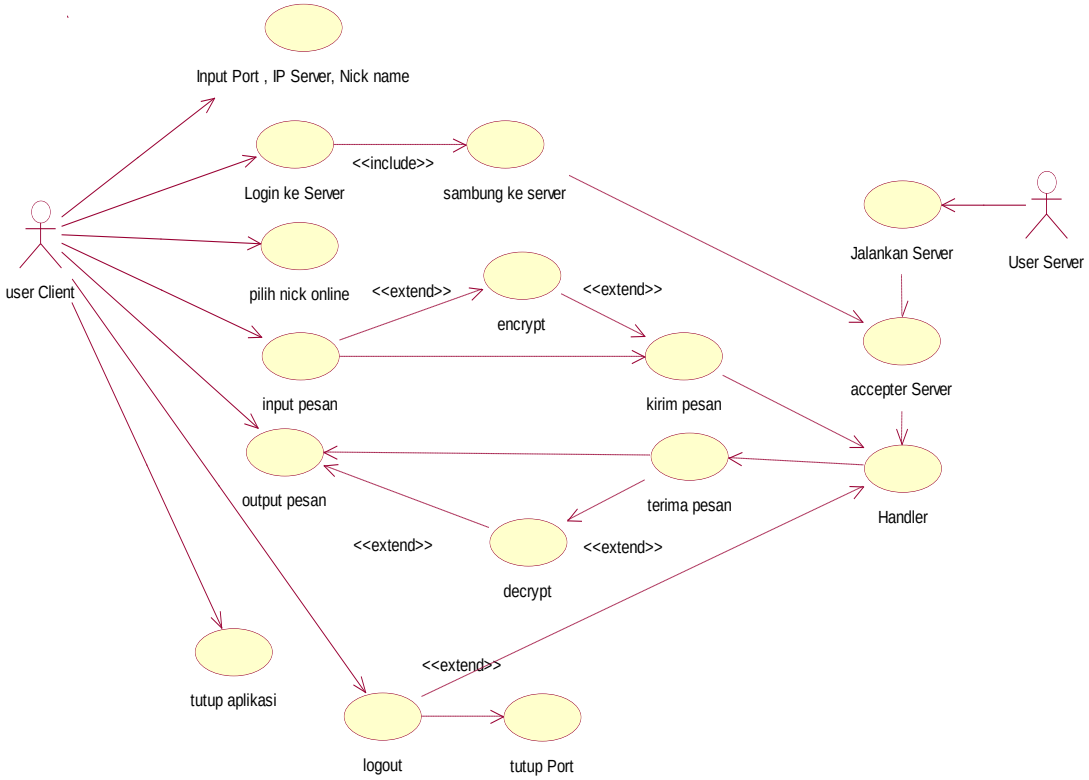
3.2.2. Pemodelan Dengan Diagram UML

Dalam membangun sistem *server* dan *client* untuk *chat* digunakan diagram-diagram UML (*Unified Modeling Language*) untuk memodelkannya, di antaranya :

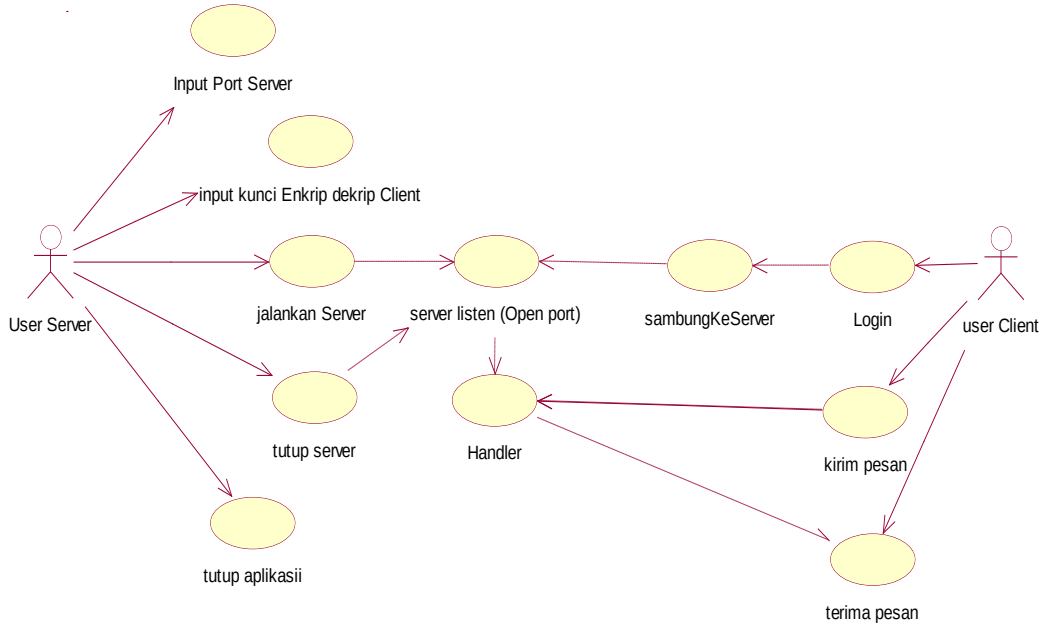
- Usecase
- Squence diagram
- Class diagram
- Activity diagram
- Flowchart

3.2.2.1. Usecase Diagram

Usecase diagram merupakan diagram UML yang digunakan untuk memodelkan proses bisnis berdasarkan perspektif pengguna sistem. Berikut ini pemodelan sistem dengan *usecase*



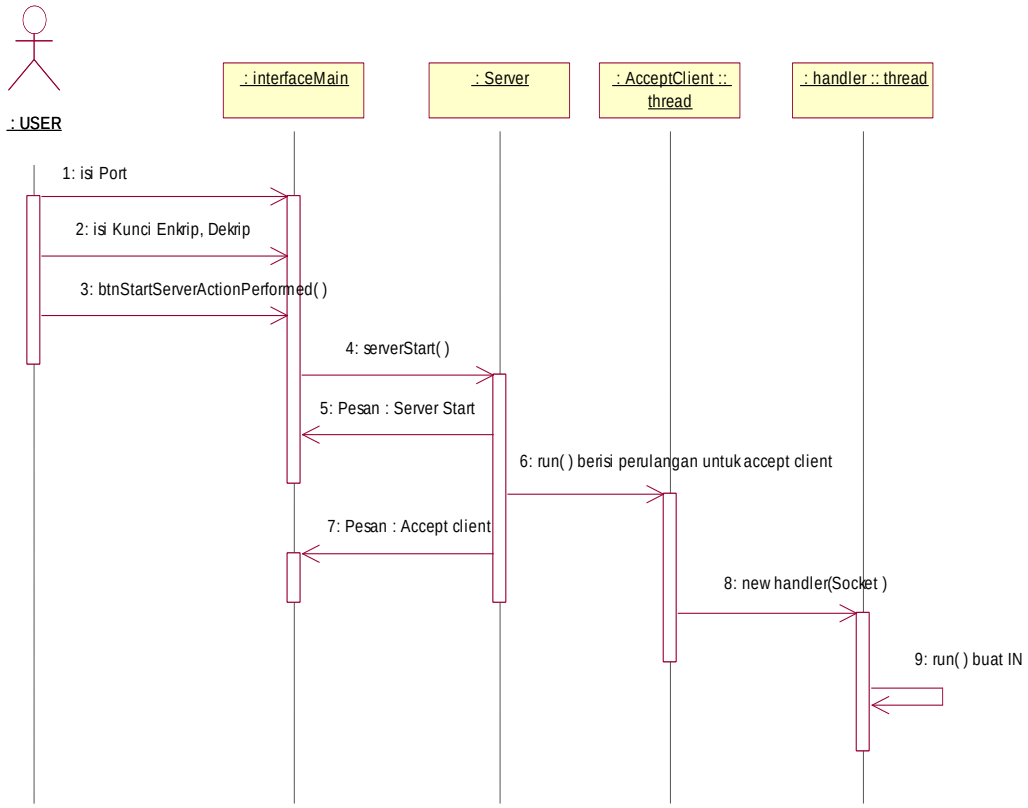
Gambar 3.2 Usecase Diagram *Client*



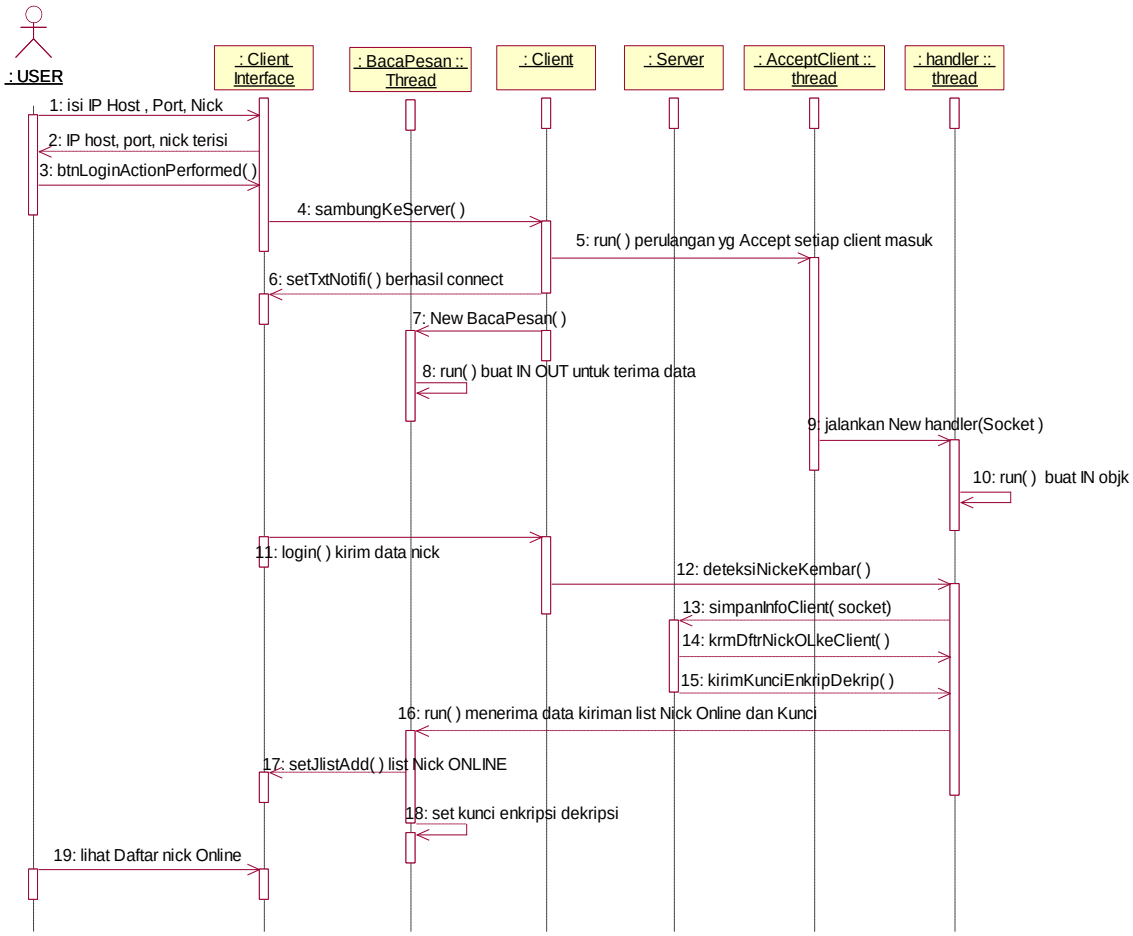
Gambar 3.3 Usecase Diagram *Server*

3.2.2.2. Sequence Diagram

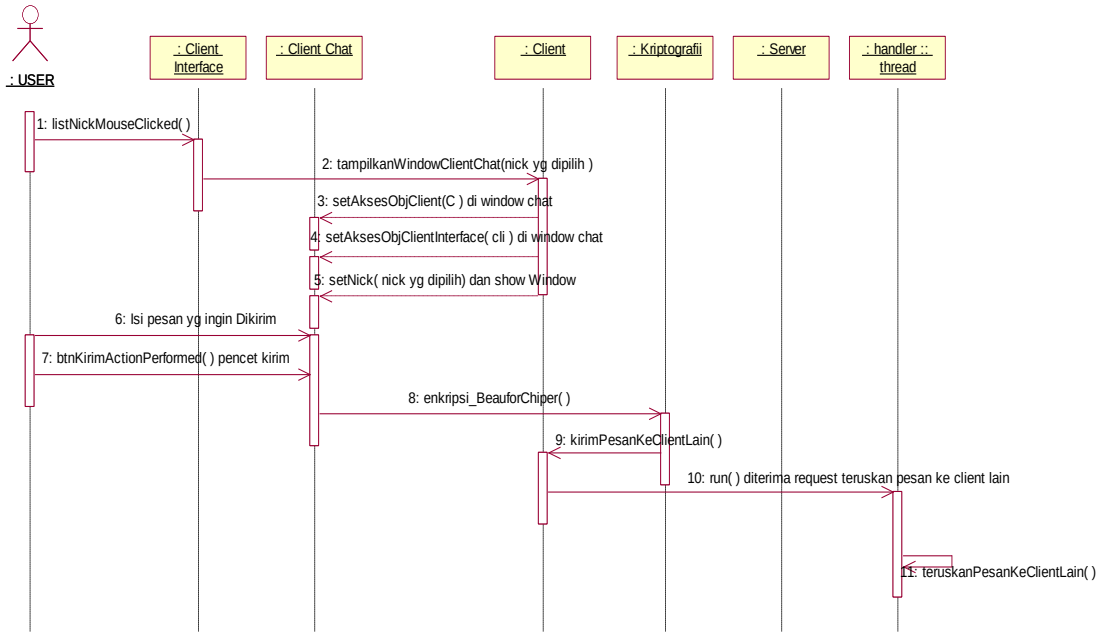
Sequence diagram ini adalah diagram UML yang digunakan untuk memodelkan pengiriman pesan antar objek yang menunjukan interaksi antar objek tersebut di mana disusun secara sekuensial (berurutan). Pemodelannya, yaitu sebagai berikut :



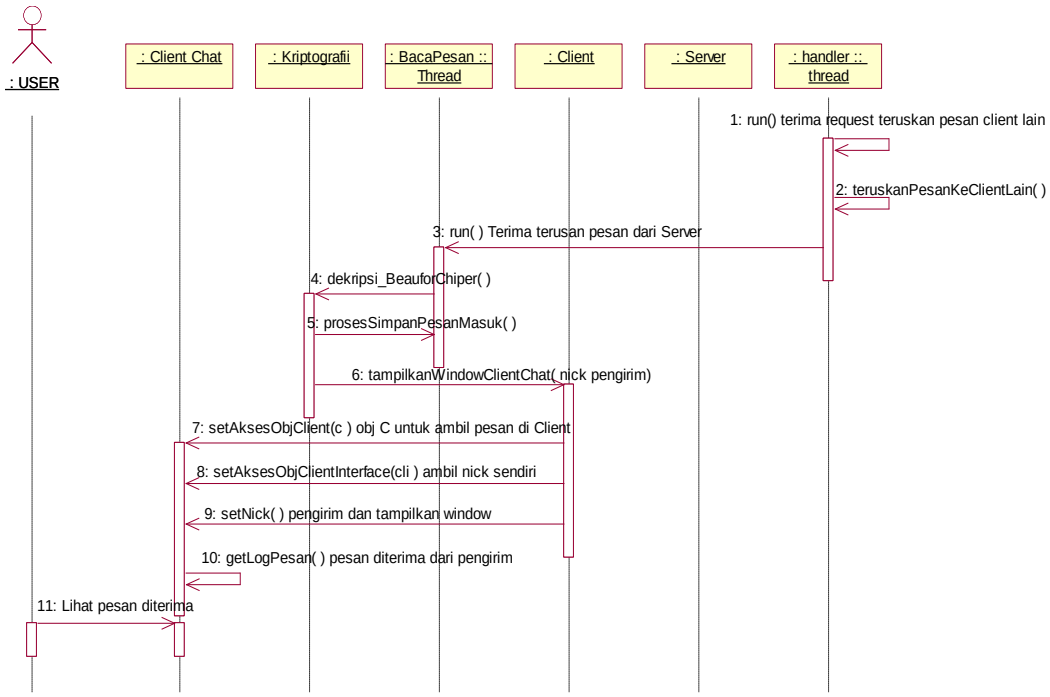
Gambar 3.4 Sequence Diagram *server listen*



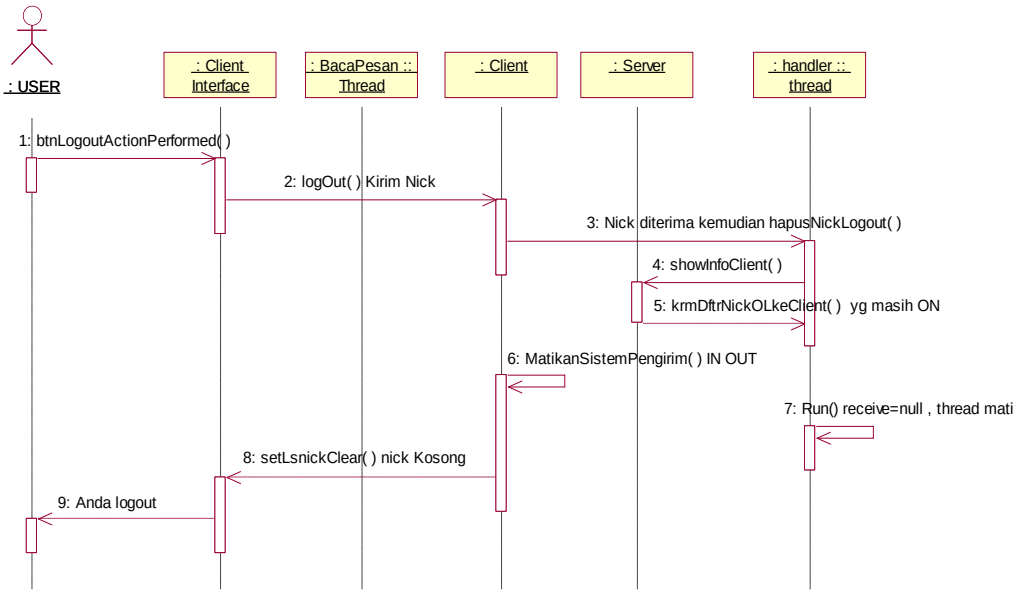
Gambar 3.5 Sequence Diagram *Client login*



Gambar 3.6 Sequence Diagram *user* kirim pesan



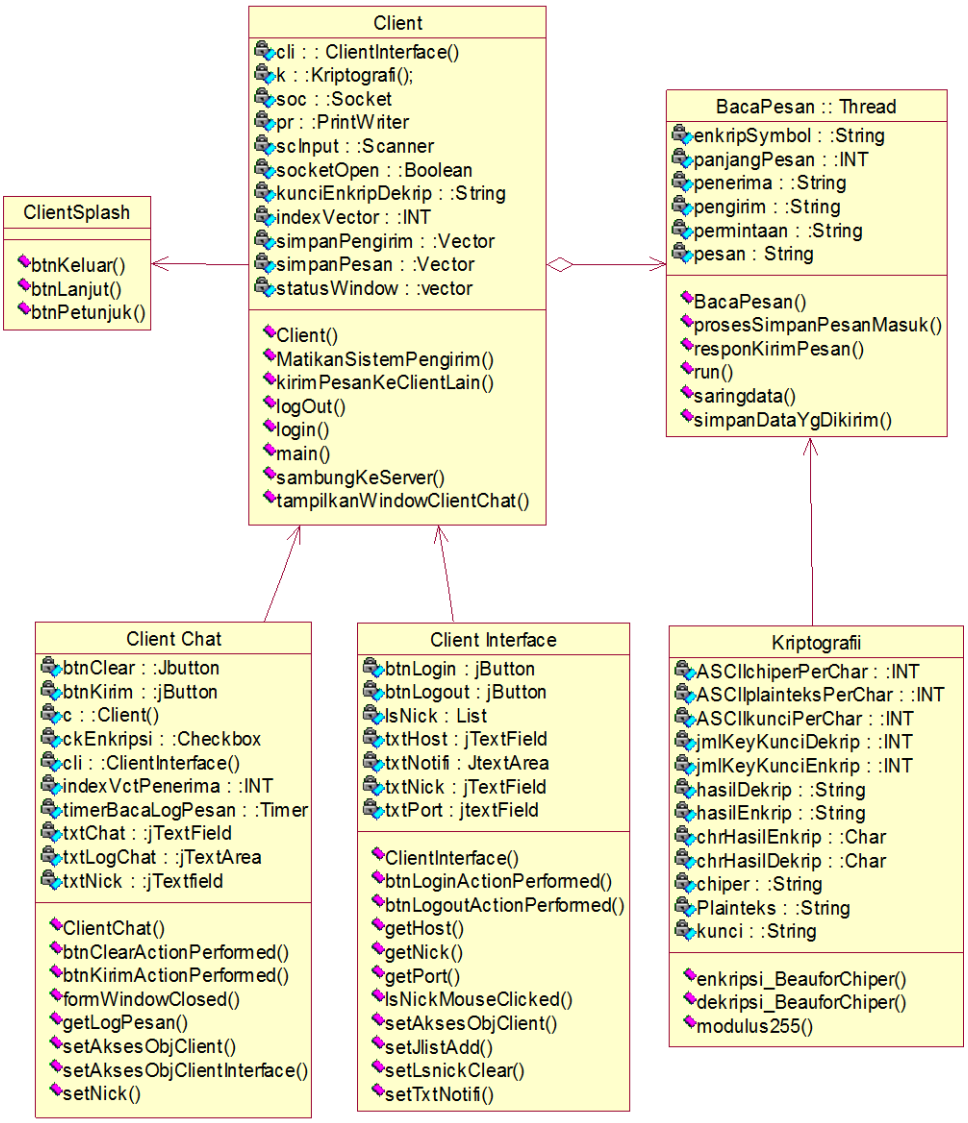
Gambar 3.7 Sequence Diagram *user* terima pesan



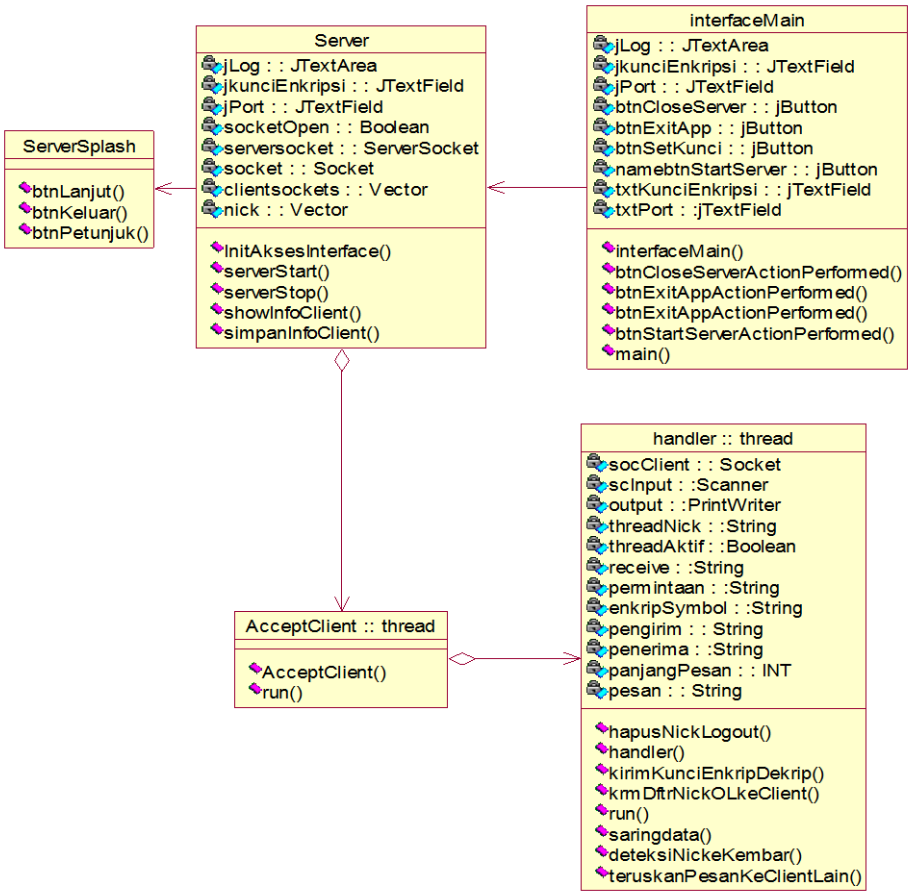
Gambar 3.8 Sequence Diagram *user logout*

3.2.2.3. Class Diagram

Class diagram merupakan diagram UML yang digunakan untuk memodelkan struktur kelas dan hubungan antar kelas yang ada. Di bawah ini merupakan pemodelan *class* diagram *client dan server* :



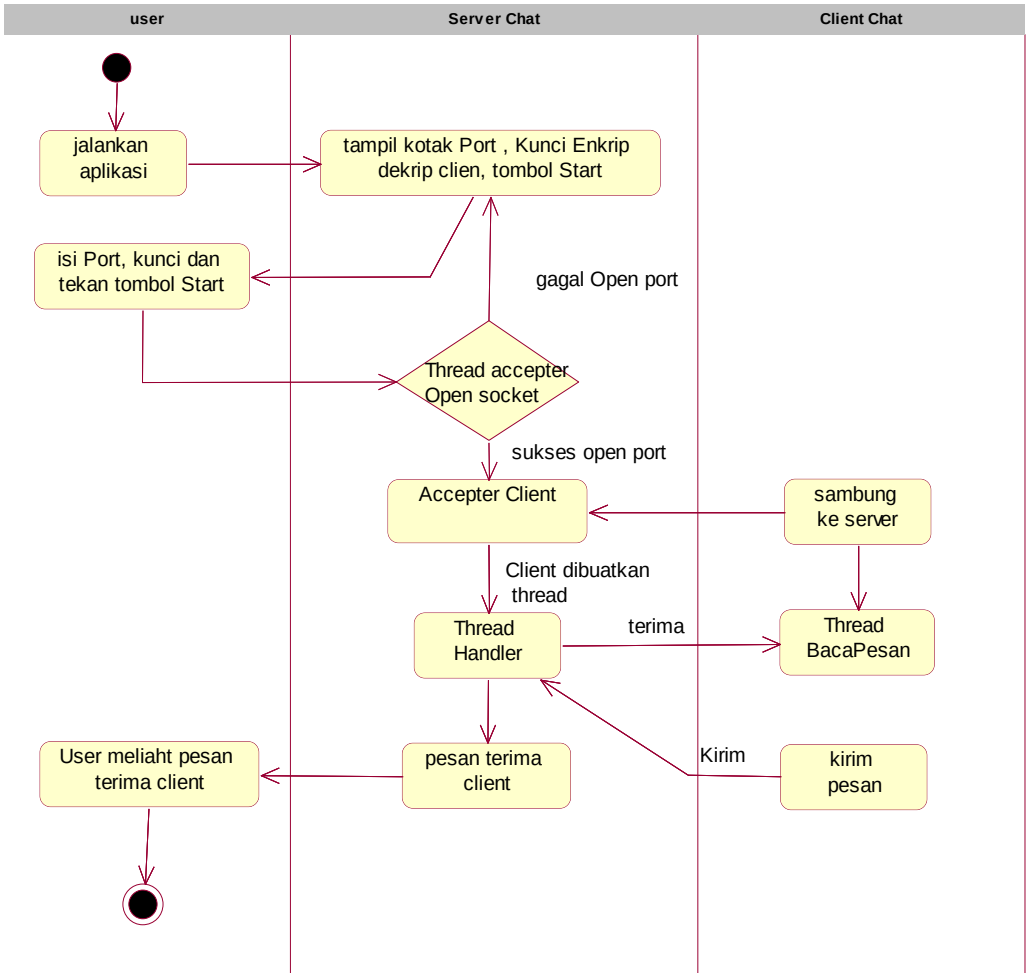
Gambar 3.9 Class Diagram *client*



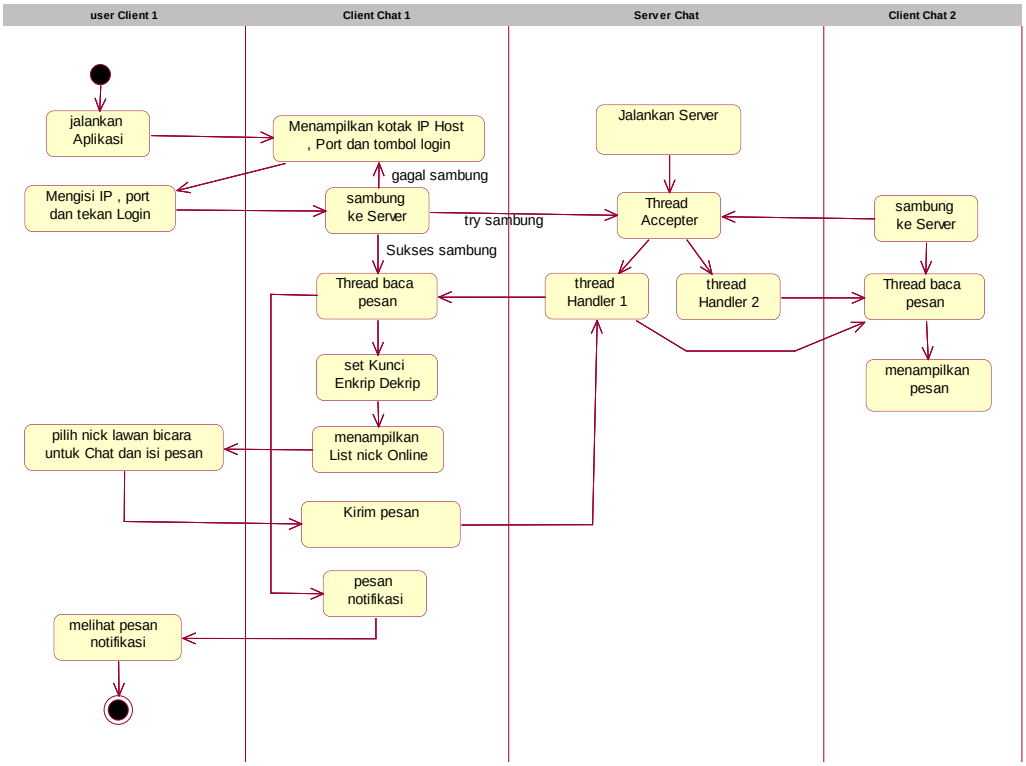
Gambar 3.10 Class Diagram *server*

3.2.2.4. Activity Diagram

Activity diagram adalah diagram UML yang digunakan untuk memodelkan perilaku *usecase* dan objek dalam sistem. Berikut adalah *activity* diagram dari *client dan server* :



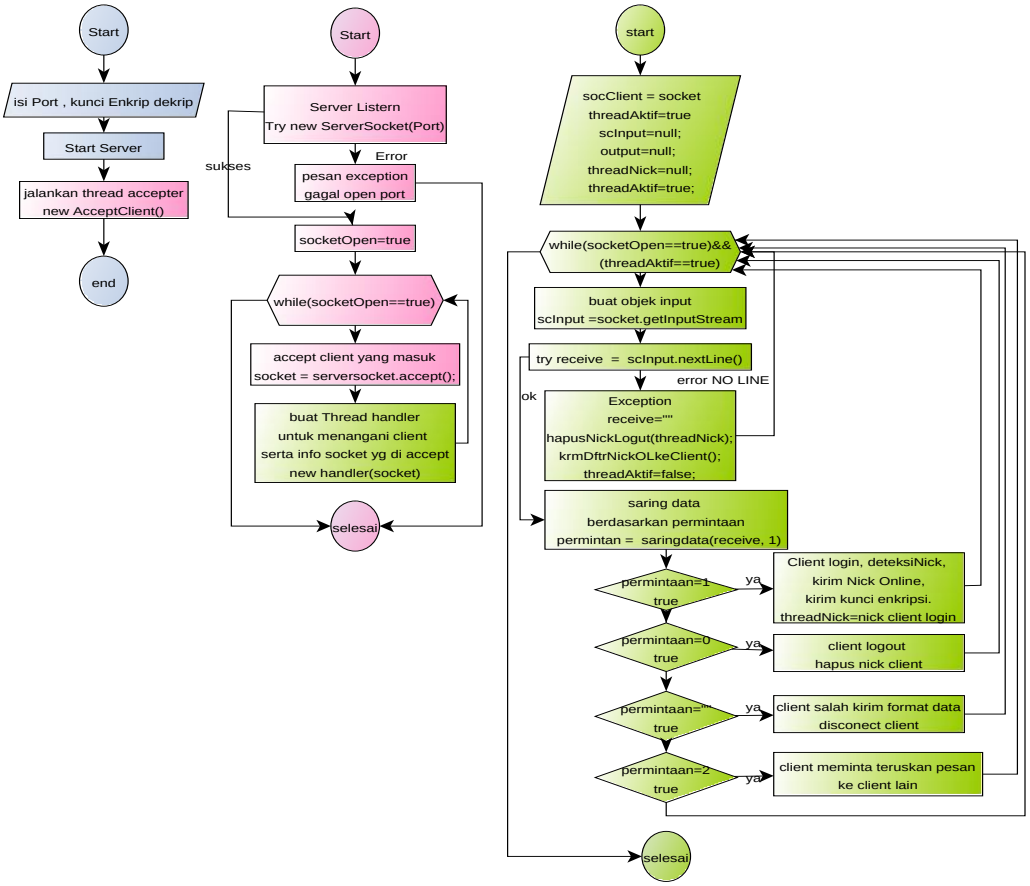
Gambar 3.11 Activity Diagram *Server*



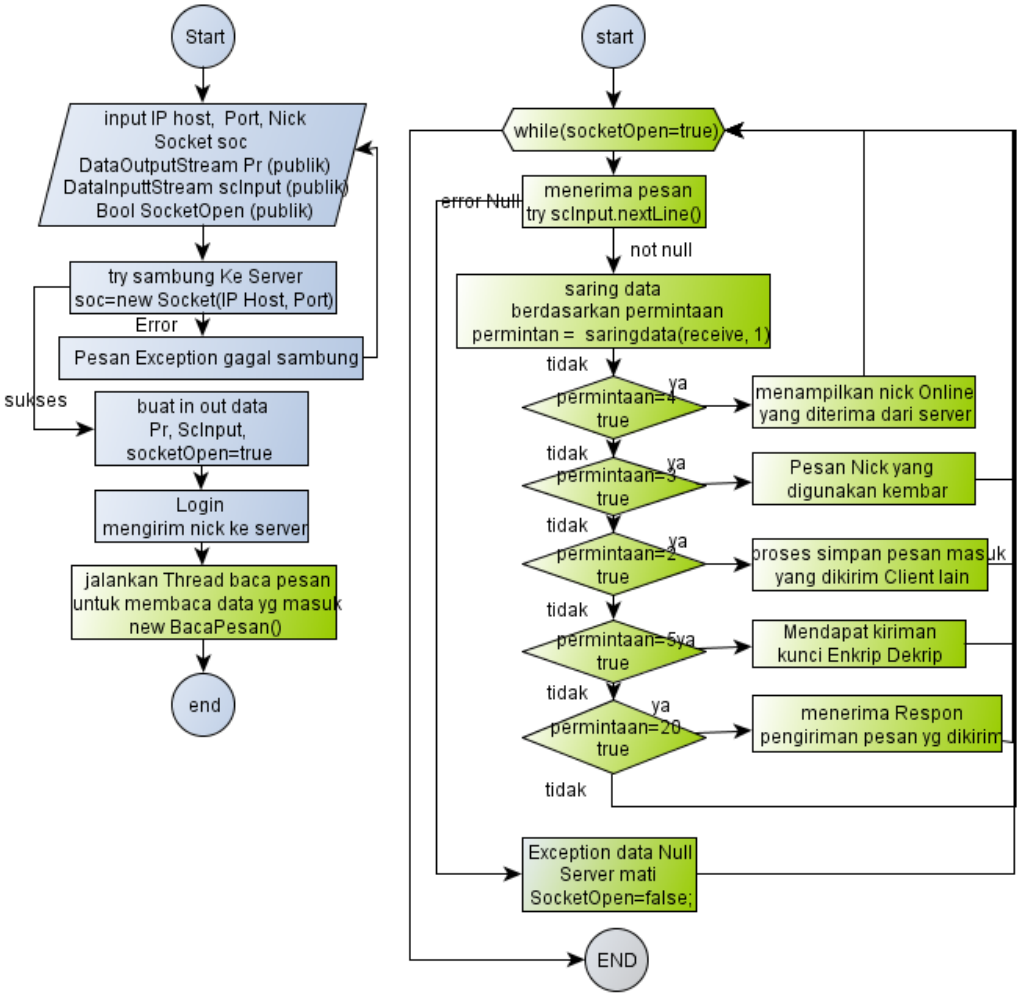
Gambar 3.12 Activity Diagram C*lient*

3.2.3. Flowchart Program

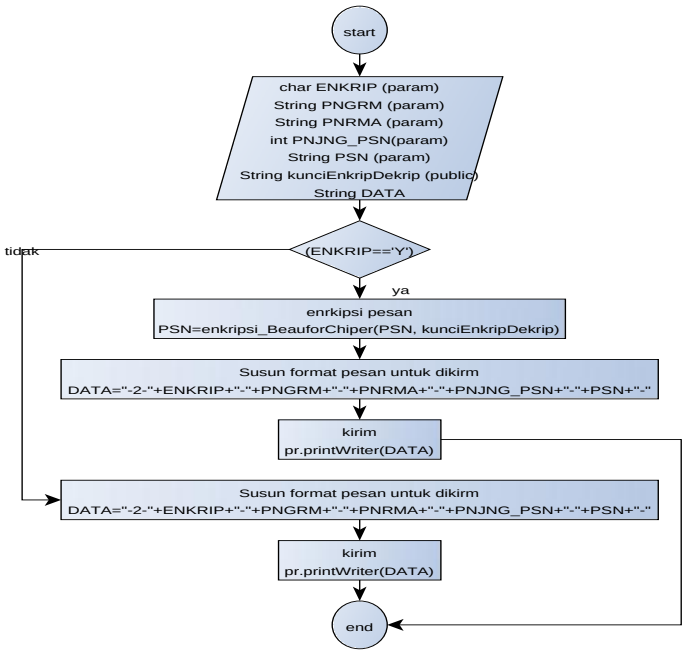
Flowchart program merupakan gambaran arus logika dari data yang akan diproses dalam suatu program bagaimana program dimulai, *input* diberikan dan proses yang dijalankan hingga program selesai dieksekusi. Berikut merupakan *flowchart* program dari aplikasi yang akan dibuat sebagai berikut :



Gambar 3.13 *Flowchart server listen*



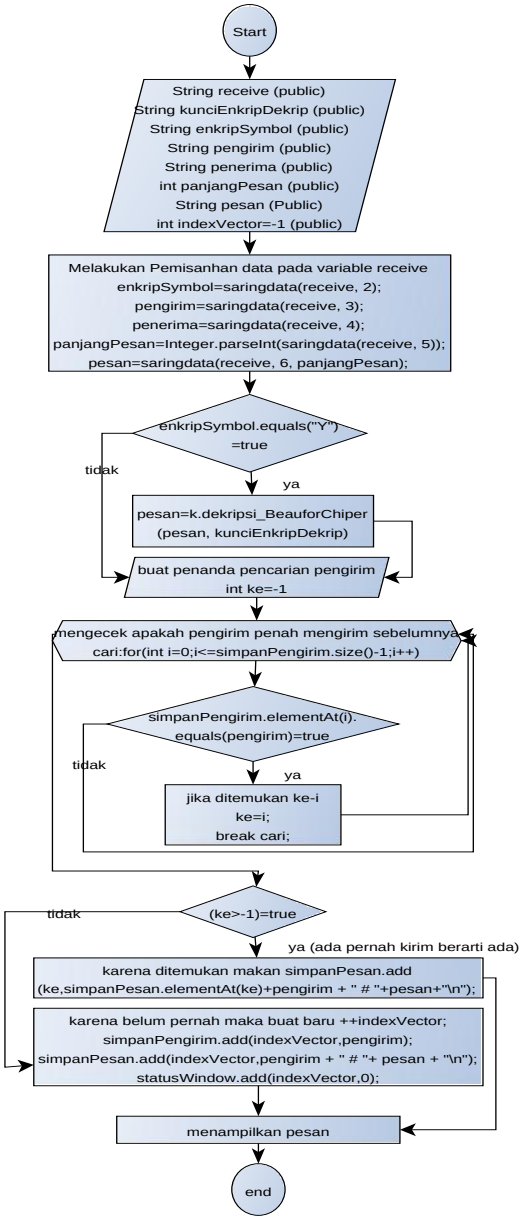
Gambar 3.14 *Flowchart client* sambung ke *server* dan *login*



Gambar 3.15 *Flowchart client* kirim pesan

Penjelasan : *Flowchart client* kirim pesan

1. Pada saat method ini dijalankan akan menerima parameter ENKRIP, PGRM, PRNM, PNJG_PSN, PSN dan menggunakan variable public kunciEnkripDekrip, juga membuat variable data
2. Jika ENRKIP berisi Y maka PSN akan di enkripsi kemudian gabungkan semua parameter kedalam variable data kemudian dikirm
3. Jika ENRKIP bukan berisi Y maka langsung digabungkan semua parameter kedalam variable data setelah itu baru dikirim, Selesai

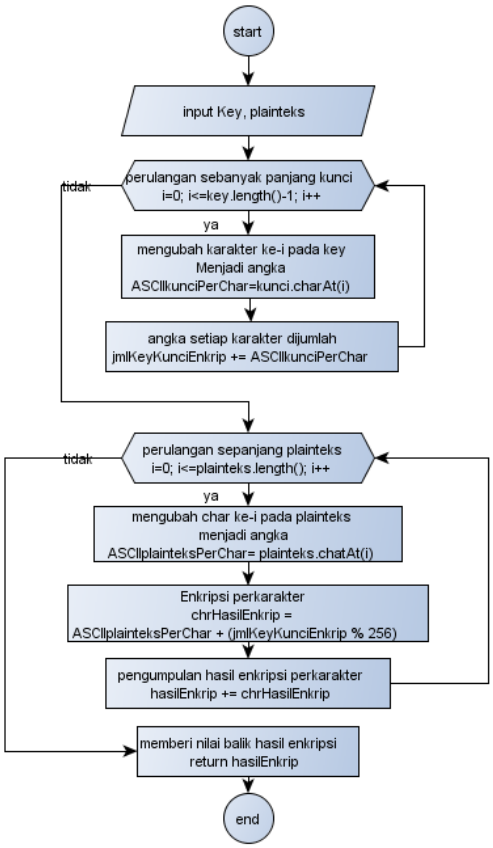


Gambar 3.16 *Flowchart client* simpan pesan diterima

Penjelasan : *flowchart client* simpan pesan diterima

1. Method ini menggunakan variable yaitu receive, kunciEnkripDekrip, enkripSymbol, pengirim, penerima, panjangPesan, pesan dan indexVector

2. Kemudian pisahkan isi receive yang menerima data dari *server* kedalam bagian masing-masing
3. Jika enkripSymbol Y maka pesan didekripsi dahulu menggunakan kunci enkripDekrip yang di terima saat *login*
4. Buat penanda pencarian *nick* pengirim dalam vector simpanPengirim, kemudian melakukan perulangan sepanjang vektor simpanPengirim.*size*()
5. Jika menemukan nick pengirim didalam vektor simpanPengirim maka simpan index tersebut ke=i setelah itu *break* cari maka paksa perulangan selesai
6. Selanjutnya jika ke > -1 lebih besar dari -1 artinya ditemukan berarti pernah mengirim maka simpan pesan tersebut pada vector simpanPesan pada index ke=I kemudian ditampilkan.
7. Jika ke tidak > -1 pengirim baru pertama kali mengirim pesan, maka dibuatkan vektor penyimpanan pesan baru yaitu simpanPengirim, simpanPesan, StatusWindow setelah itu disimpan dan ditampilkan
8. selesai



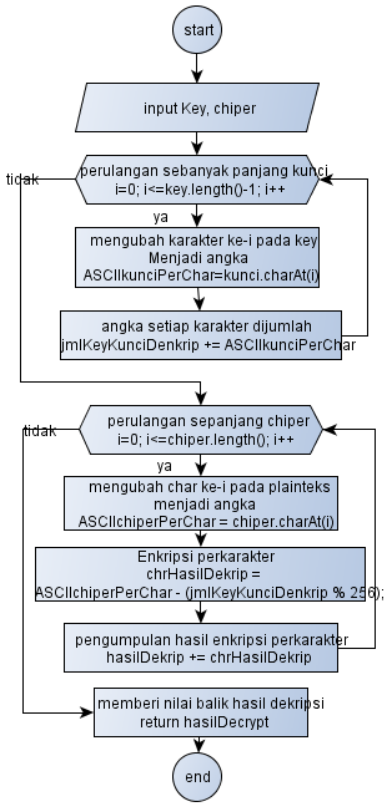
Gambar 3.17 *Flowchart* enkripsi

Penjelasan : *flowchart* enkripsi

1. Pada enkripsi ini memerlukan input plainteks dan kunci
2. Kemudian perulangan sepanjang kunci ubah perkarakter kunci kedalam angka ASCII (*char to integer*) dan dijumlahkan semuanya kedalam jmlKeyKunciEnkrip
3. Kemudian perulangan lagi sepanjang plainteks ubah perkarater plainteks menjadi angka ASCII kemudian dihitung menggunakan algoritma beaufort, setelah itu

angka ASCII di kebalikan ke huruf lagi (*int to char*) dimasukan kedalam variabel ASCIIplainteksPerChar

4. Kemudian karakter yg sudah dienkripsi ASCIIplainteksPerChar dikumpulkan kedalam variable hasilEnkrip, setelah perulangan selesai akan di return hasilEnkrip

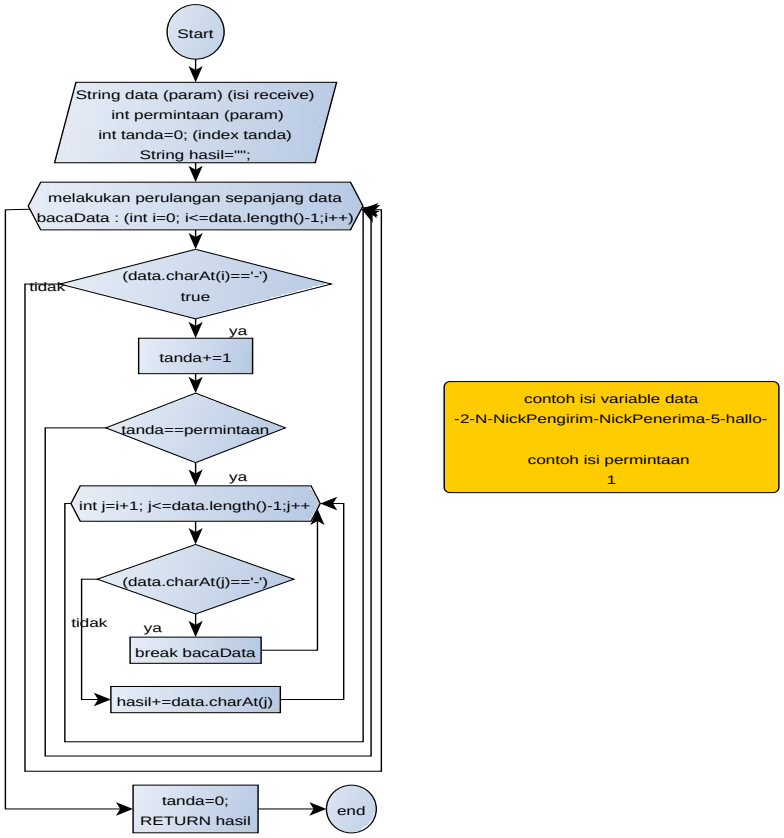


Gambar 3.18 *Flowchart* dekripsi

Penjelasan : *flowchart* dekripsi

1. Pada dekripsi ini memerlukan input chipper dan kunci

2. Kemudian perulangan sepanjang kunci ubah perkarakter kunci kedalam angka ASCII (*char to integer*) dan dijumlahkan semuanya kedalam jmlKeyKunciEnkrip
3. Kemudian perulangan lagi sepanjang chiper ubah perkarater chiper menjadi angka ASCII kemudian dihitung menggunakan algoritma beaufort, setelah itu angka ASCII di kebalikan ke huruf lagi (*int to char*) dimasukan kedalam variabel ASCIIchiperPerChar
4. Kemudian karakter yg sudah didekripsi ASCIIchiperPerChar dikumpulkan kedalam variable hasilDekrip, setelah perulangan selesai akan di return hasilDekrip



Gambar 3.19 *Flowchart* saringdata

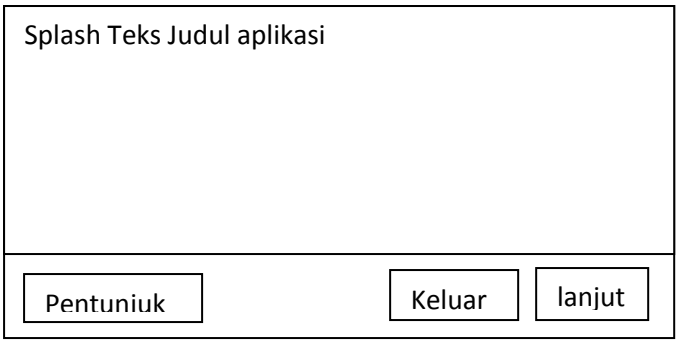
Penjelasan : *flowchart* saringdata

1. Pada saat method ini dijalankan akan menerima 2 parameter yaitu data, permintaan isi data dan permintaan dapat dilihat di kotak kuning juga membuat variable tanda dan hasil.
2. Melakukan perulangan sepanjang *data.length()* jika menemukan tanda '–' maka tanda+=1.

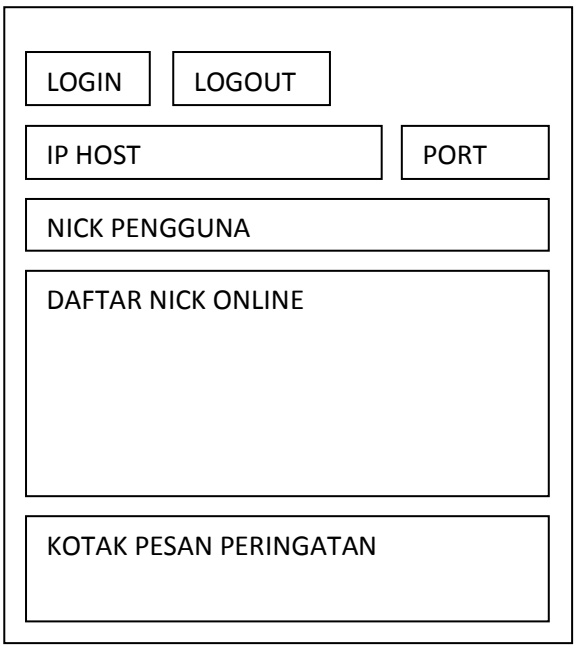
3. Jika tanda sesuai dengan permintaan maka lakukan perulangan untuk mengambil data, karena data didepan tanda ‘-’ maka j=i(tanda) +1.
4. Ketika dalam perulangan mengambil data menemui ‘-‘ maka paksa perulangan pertama ‘bacadata :‘ selesai kemudian return hasil, selesai.
5. Jika tidak menemui ‘–‘ maka karakter ke-j pada data tambahkan kedalam hasil, sampai ketemu tanda ‘–‘
6. (lihat No. 4) selesai.

3.2.4. Rancangan Tampilan

Tahap ini adalah tahap dimana rancangan tampilan sebelum di implementasikan kedalam program, sehingga diharapkan akan adanya gambaran tentang program yang akan dibuat.



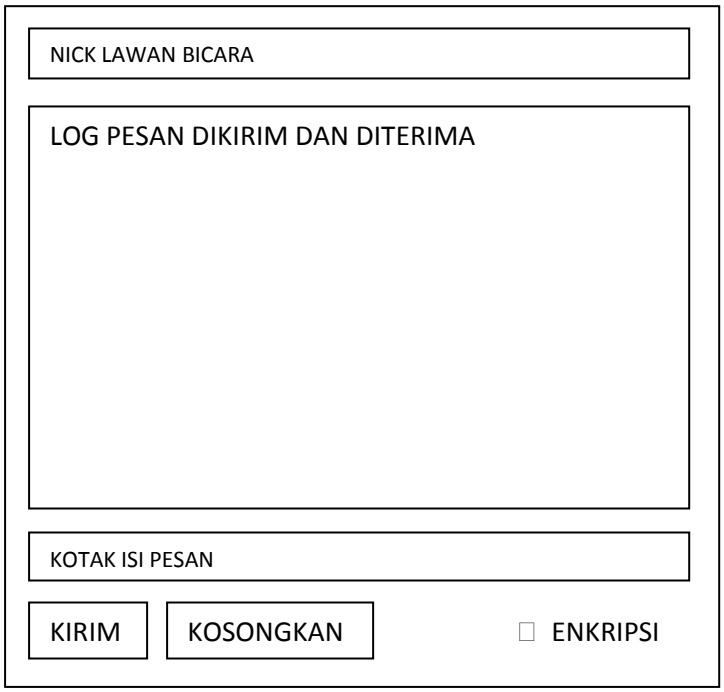
Gambar 3.20 tampilan rancangan *splash client*



Gambar 3.21 tampilan rancangan aplikasi ketika dibuka

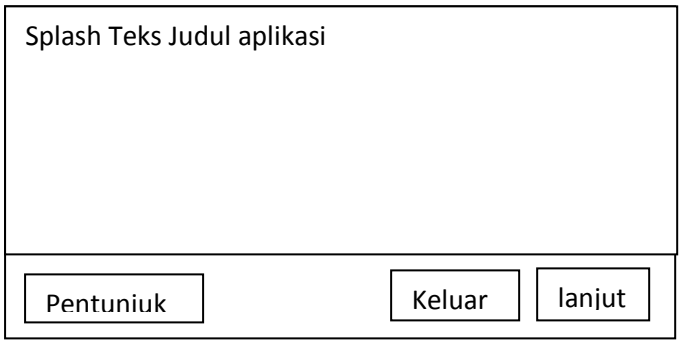
Gambar diatas merupakan tampilan awal ketika aplikasi client chat dibuka, pada saat dibuka user diminta memasukan *IP host server* juga *port* yang digunakan *server* dan *nick* yang akan digunakan sebagai pengenal dirinya ketika *login*, kotak pesan

akan menampilkan pesan ketika terjadi *error* atau menerima data dari *server* dan ketika *user* ingin berbicara dengan *client* lain dapat memilih daftar *nick* yang online dengan cara klik *nick*

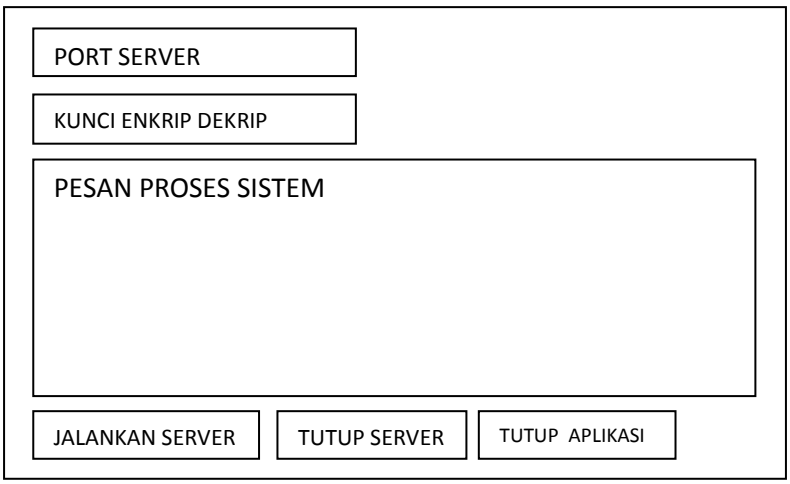


Gambar 3.22 Tampilan ketika memilih *nick* lawan bicara

Gambar diatas merupakan tampilan ketika *user* memilih nick lawan bicara pada daftar *nick* yang online untuk mengirim pesan user diminta memasukan pesan pada kotak isi pesan kemudian beri tanda centang jika pesan ingin diekripsi



Gambar 3.23 Tampilan rancangan *splash server*



Gambar 3.24 Tampilan rancangan aplikasi *server*

Gambar diatas merupakan tampilan ketika *user* membuka aplikasi server, user diminta memasukan *port* yang akan digunakan untuk menerima *client* dan kunci enkrip dekrip yang nanti akan di pasang kepada setiap *client* yang terhubung