BAB III

ANALISA DAN DESAIN SISTEM

III.1. Analisa Masalah

Salah satu fungsi dari sistem jaringan komputer yang banyak digunakan adalah penerapan *file* transfer, dimana dengan penerapan *file* transfer ini setiap pengguna dapat saling mengirim data. Fitur keunggulan perangkat komputer adalah dapat menjadi media komunikasi data transfer. Banyak keuntungan yang di dapat dalam proses ini, salah satunya mengirim data melalui media jaringan hanya dengan mengirim dokumen *file* dengan terhubung pada jaringan komputer. Di samping keuntungan juga banyak ditemukan beberapa kesulitan dalam proses transfer data *file,* salah satunya adalah pengiriman data dengan ukuran cukup besar yang memakan waktu yang cukup lama. Memandang hal ini dibutuhkan solusi untuk dapat mentransfer data dengan implementasi pemampatan data dengan algoritma kompresi *file* yang diterapkan pada media transfer yang ada pada jaringan komputer.

Salah satu algoritma pemampatan data *file* adalah *GZip. GZip* merupakan algoritma standar yang telah dipakai banyak pihak untuk melakukan proses kompresi. *GZip* atau *GNUZip* pertama kali diciptakan oleh *Jean-loup Gailly* dan *Mark Adler* untuk *Listing* dekompresi pada tanggal 31 Oktober 1992. Algoritma *GZip* sendiri telah menjadi metode kompresi *default* di sistem operasi *Linux* dan *BeOS*, sehingga kemampuan kompresi datanya cukup baik. Secara teoritis, algoritma *GZip* sendiri merupakan pengembangan dari algoritma *Deflate* yang

merupakan kombinasi dari algoritma *LZ77* dan *Huffman Coding*, tetapi dengan lisensi *open source* sehingga dapat diterapkan oleh banyak pihak dengan lebih bebas.

III.1.1. Strategi Pemecahan Masalah

Tujuan pengujian penelitian ini adalah memeriksa apakah kebutuhan fungsional sudah terpenuhi yakni dapat mengompresi dan mendekompresi *file* yang ada. Tujuan lain pengujian ini adalah untuk mengukur performa aplikasi yang mencakup rasio kompresi, entropi, dan waktu kompresi. Beberapa rancangan kasus uji yang dibuat dalam penelitian ini mencakup dua macam kasus uji. Pengujian pada penelitian ini mencakup beberapa kasus uji sebagai berikut :

1. Pengujian terhadap beberapa ukuran *Search Buffer*. Pengujian dilakukan terhadap ukuran *search buffer* yang berbeda-beda.
2. Pengujian terhadap berbagai macam tipe *file.* Pengujian dilakukan terhadap berbagai jenis *file* teks(txt).
3. Pengujian terhadap beberapa *file* sekaligus. Pengujian dilakukan terhadap banyak *file* sekaligus dalam *storage.* Pada kasus ini dua buah sekaligus.
4. Pertama-tama program membaca *file* sebesar y, lalu dimasukkan ke dalam suatu *array of byte* dengan ukuran x. Apabila ukuran y lebih besar dari x maka isi *file* yang diambil hanya sebesar x terlebih dahulu. Sebelumnya *file output* diberi *header* yang menunjukkan bahwa *file* ini benar telah terkompresi oleh algoritma *Gzip*. Pengompresian dilakukan terhadap *buffer* tersebut. Bagian yang disimpan adalah isi dari *byte* tersebut dan posisi *byte*.

5. Setelah seluruh isi *buffer* tersebut terkompresi, isi *buffer* dimasukkan ke *file output*. *Buffer* lalu di isi kembali sisa dari *file* dengan ukuran y-x (karena sudah diambil sebanyak x pada kompresi sebelumnya) sebanyak x lalu dilakukan kompresi terhadap *buffer* tersebut. Hal ini dilakukan terus-menerus sampai *file* dengan ukuran y sudah selesai terkompresi seluruhnya. Dan dicari perbandingan sebelum dan setelah dilakukan proses. Dari perbandingan ini akan diperoleh kelebihan dan kekurangan dari algoritma tersebut.

III.1.2. Penerapan Algoritma Gzip

Menurut Soetam Rizky Wicaksono, *GZip* merupakan algoritma standar yang telah dipakai banyak pihak untuk melakukan proses kompresi. *GZip* atau *GNU Zip* pertama kali diciptakan oleh *Jean-loup Gailly* dan *Mark Adler* untuk *Listing* dekompresi pada tanggal 31 Oktober 1992. Secara teoritis, algoritma *GZip* sendiri merupakan pengembangan dari algoritma *Deflate* yang merupakan kombinasi dari algoritma *LZ77* dan *Huffman Coding*, tetapi dengan lisensi *open source* sehingga dapat diterapkan oleh banyak pihak dengan lebih bebas. Membuang sebagian data yang tidak berguna, tidak dirasakan, tidak dilihat oleh manusia sehingga manusia masih beranggapan bahwa data tersebut masih bisa digunakan walaupun sudah dikompresi. Kelemahan *GZIP* ini beberapa kali saya mencoba banyak konfigurasi *.htaccess* untuk kompresi *GZip* di aplikasi, dari berbagai web lain. Namun, sepertinya kompresi *Apache GZip (mod_gzip)* tidak didukung oleh *server Apache* bagaimanapun juga sepertinya gagal. *Content-encoding* dari *file-file plain-text* tidak seperti yang diinginkan ("*gzip*" atau

"*deflate*"). Data hasil kompresi dapat didekompres lagi dan hasilnya tepat sama seperti data sebelum proses kompresi. Contoh aplikasi: *ZIP, RAR, GZIP, 7-Zip*. Teknik ini digunakan jika dibutuhkan data setelah dikompresi harus dapat diekstrak / dikompres lagi tepat sama. Contoh pada data teks, data program/biner, beberapa *image* seperti *GIF* dan *PNG.*

Menurut Charles Januari, *GZip* yang telah dibuat ini kemudian dilakukan uji coba, sehingga kita bisa melihat hasil kompresi dan simulasi yang telah dibuat. Data dari hasil penelitian dengan algoritma *GZip* ini beserta pengimplementasiannya akan dibandingkan dengan *file* aslinya. Dari perbandingan ini akan diperoleh kelebihan dan kekurangan dari algoritma ini, dan selanjutnya dari perbandingan tersebut akan diperoleh kesimpulan. Analisis dilakukan dengan menggunakan *file* teks dengan isi 'AAAAAAAAAAABCABCAAAAA'. Proses kompresi mula-mula dilakukan dengan metode *LZ77* dimana pada metode ini isi *file* dibaca sebagai sebuah *string* yang ditampung dalam variabel baru yaitu *Siliding Windows* (SW) dan *Read Ahead* (RA). SW merupakan *array* yang dapat menampung 10 *byte* dan RA dapat menampung 11 *byte*. SW kemudian di isi dengan 10 karakter pertama dari *frase* dan RA diisi dengan 11 karakter dimulai dari *index* terakhir SW ditambah 1. Karena SW dan RA merupakan *array* dengan tipe data *byte*, maka dibutuhkan *function* '*ord*' untuk mengkonversikan nilai *char* ke bentuk *byte*. Terdapat sebuah variabel 'hasil' yang akan menyimpan hasil dari kompresi yang telah dilakukan. Pertama-tama, variabel hasil di isi dengan SW.

Frase 'AAAAAAAAAAABCABCAAAAA'

Hasil ‘AAAAAAAAAA’

Setelah literasi awal dilakukan, kemudian dibandingkan apakah isi dari RA sama atau merupakan bagian dari SW, jika tidak maka *index* akhir dari RA akan dikurangi 1. Proses ini terus dilakukan jika isi dari RA sama dengan 2. Jika RA sama dengan 2 maka variabel hasil akan ditambahkan dengan isi dari RA dan *index* awal SW akan bergeser sebanyak 2 ke kanan.

Frase ‘AAAAAAAAAAABCABCAAAAA’

Hasil ‘AAAAAAAAAAAB’

Jika isi dari RA sama atau terdapat pada SW maka pada variabel hasil akan dicatat 2 buah kode yang terdiri dari *offset* dan *length*. Setelah itu *index* awal SW akan bergeser sebanyak isi dari RA.

Frase ‘AAAAAAAAAAABCABCAAAAA’

Hasil ‘AAAAAAAAAAABCA22’

Pada proses kompresi akan berhenti apabila *index* akhir dari SW sudah sama dengna panjang dari *frase*. Proses kemudian dilanjutkan dengan menambah *flag* pada setiap 8 *byte* pada variabel hasil. Hasil akhir dari proses kompresi dengan metode *LZ77* ini ditampung dalam variabel ‘hasil2’.

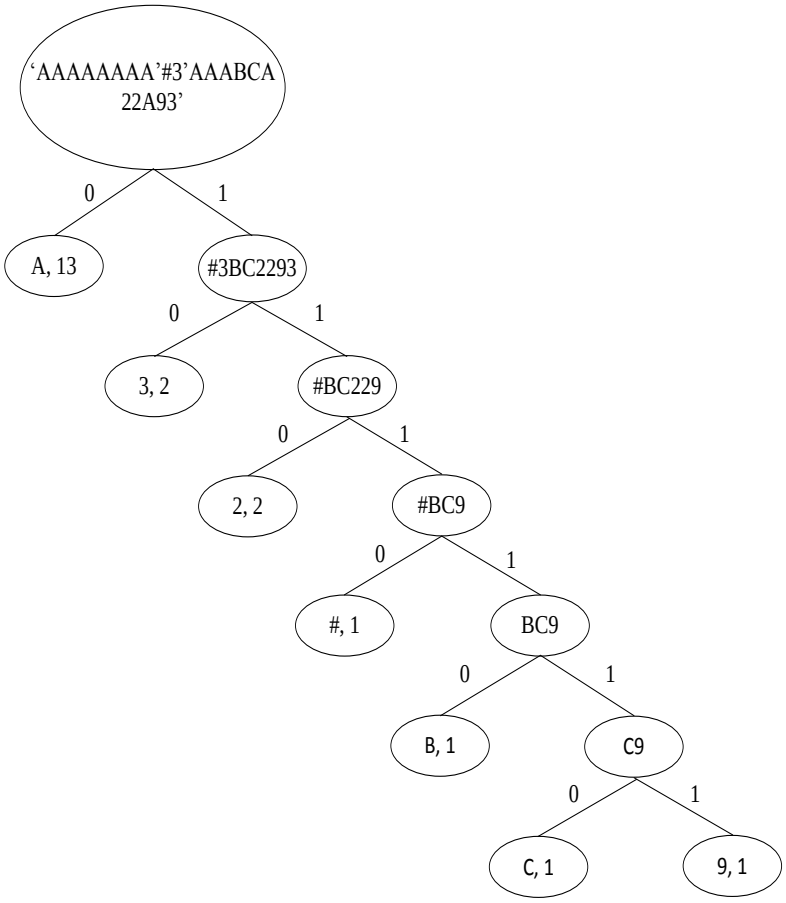
Frase ‘AAAAAAAAAAABCABCAAAAA’

Hasil ‘AAAAAAAAAAABCA2293’

Hasil2 ‘AAAAAAAA’#3’AAABCA22A93’

Isi dari variabel hasil 2 tersebut kemudian dikompresi lagi dengan metode *Huffman*. Langkah pertama dari proses kompresi dengan metode *Huffman* adalah dengan membentuk sebuah *tree* yang berasal dari kumpulan *node-node*. Setiap

node memiliki variabel data dan *value* dan juga memiliki 1 anak yaitu *left* dan *right*. Nilai dari variabel hasil 2 akan dikonversikan menjadi bilangan *ASCII*(0-225). Nilai dari variabel *value* diperoleh dari jumlah kemunculan karakter tersebut.



Gambar III.1. Pohon *Huffman*

Adapun pada *Huffman tree* setiap *leaf* adalah karakter yang terdapat pada sebuah *file* yang akan dikompresi. Setiap mencapai suatu *leaf*, proses akan mencatat alur yang terjadi ke dalam sebuah variabel '*path*' dan kemudian akan di simpan ke dalam variabel *dictionary*, dapat dilihat pada tabel III.1.

Tabel III.1. Kode *dictionary Huffman*

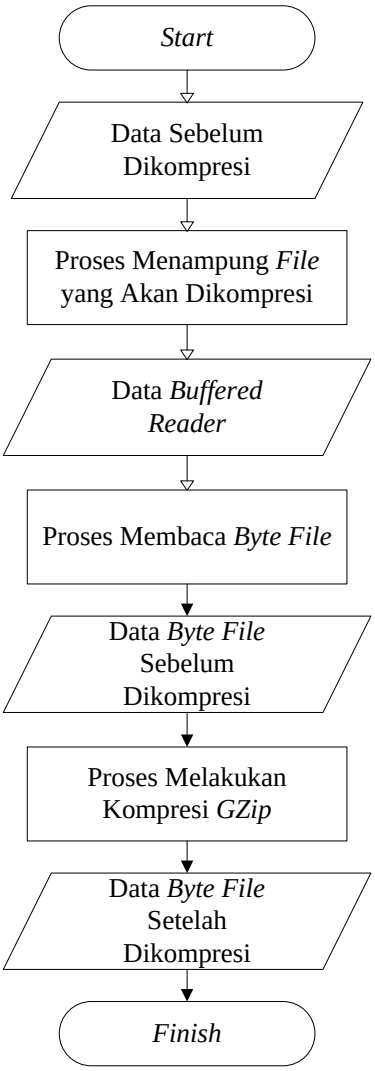
Karakter	Frekuensi	Peluang	Kode *Huffman*
A	13	13/21	0
3	2	2/21	10
2	2	2/21	110
#	1	1/21	1110
B	1	1/21	11110
C	1	1/21	111110
9	1	1/21	1111110

Setelah dikompres dengan menggunakan Kode *Huffman*, *string* tersebut dapat direpresentasikan menjadi rangkaian *bit*:

010110111011110111101111110

Setelah *dictionary* telah diperoleh, maka langkah terakhir yaitu menyalin *dictionary* ke *file output* kemudian mengkonversi setiap karakter yang terdapat pada *file input* sesuai dengan *dictionary* karakter tersebut. (Charles Januari Napitupulu; 2010: 3)

Untuk *flowchart* algoritma *GZip* dapat dilihat pada keterangan berikut ini, yang dimana terdapat prosedur yang ada pada algoritma *GZip* yaitu:



Gambar III.2. *Flowchart* Algoritma *GZip*

III.1.3. Spesifikasi Perangkat Yang Digunakan

Dalam proses perancangan aplikasi ini akan menggunakan perangkat lunak sebagai berikut :

1. *SDK Java*, sebagai mesin pemrograman *Java.*
2. Sistem Operasi *Windows 7* (*Seven*) sebagai OS yang berjalan pada perangkat komputer.

Sedangkan untuk perangkat keras yang akan digunakan dalam proses perancangan adalah :

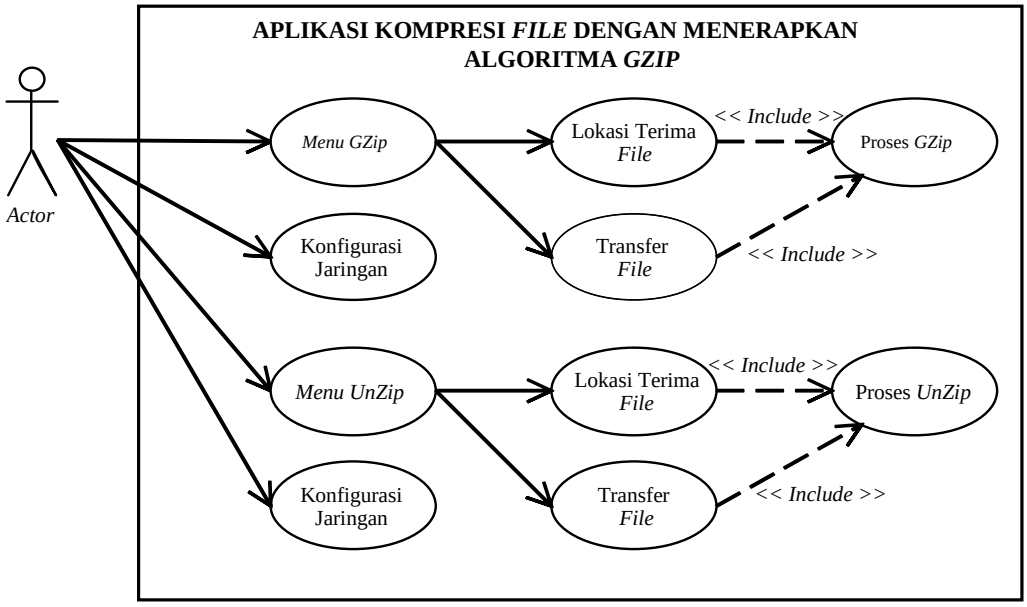
1. Personal Komputer dengan spesifikasi *ProsessorIntelAtom* 1,6 Ghz, *RAM* 2GB, *Hardisk* 320 *GByte*, *LAN Card*.
2. *Generic PnP Monitor*, dan *Mouse*.

III.2. Perancangan Sistem

Pada tahapan ini merupakan gambaran dari sistem yang akan dikembangkan. Penjelasan lebih *detail* mengenai perancangan tersebut digambarkan dengan konsep diagram UML. Berikut adalah diagram dari rancangan aplikasi yang akan penulis rancang.

III.2.1. *Use Case Diagram*

Use case diagram menggambarkan aktor yang menggunakan aplikasi dan perilaku pengguna, seperti pada gambar III.3.

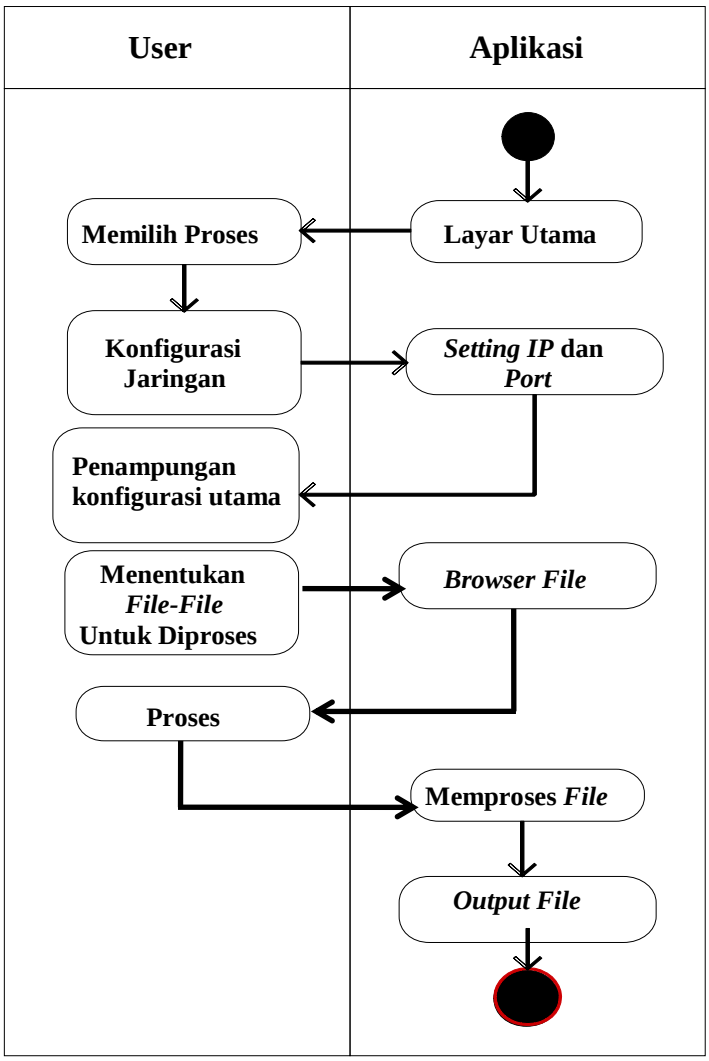


Gambar III.3. *Use Case Diagram*

Kegiatan aktor atau pengguna pada aplikasi kompresi *file* ini, pengguna dapat memilih melakukan kompresi dan dekompresi terhadap *file-file* yang di-*input*-kan. Pengguna dapat menentukan penyimpanan *output* dari *file-file* yang telah diproses.

III.2.2.*Activity diagram*

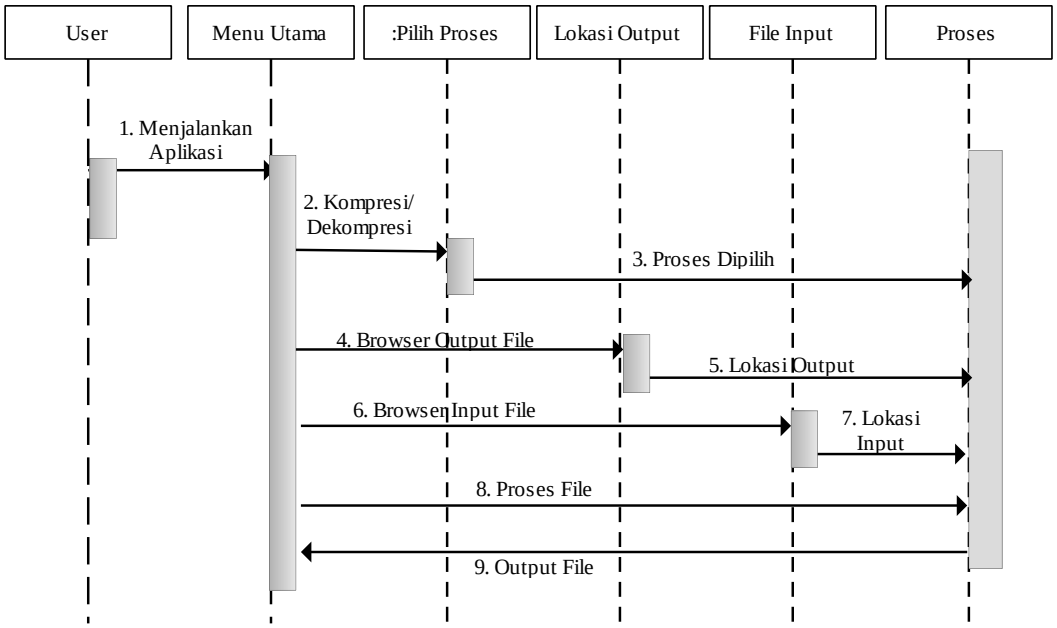
Activity diagram adalah teknik untuk mendiskusikan logika *prosedural*, proses bisnis dan aliran kerja. Dalam banyak kasus *Activity diagram* banyak mempunyai peran seperti halnya *flowchart*. *Activity diagram* bisa mendukung perilaku pararel sedangkan *flowchart* tidak bisa. Berikut ini adalah *activity diagram* aplikasi kompresi dan dekompresi yang dirancang, dapat dilihat pada gambar III.4.



Gambar III.4. *Activity Diagram* Aplikasi

III.2.3. *Sequence Diagram*

Sequence diagram menggambarkan kegiatan dari skenario penggunaan aplikasi, *sequence diagram* memilih proses yang dapat dilihat pada gambar III.5.

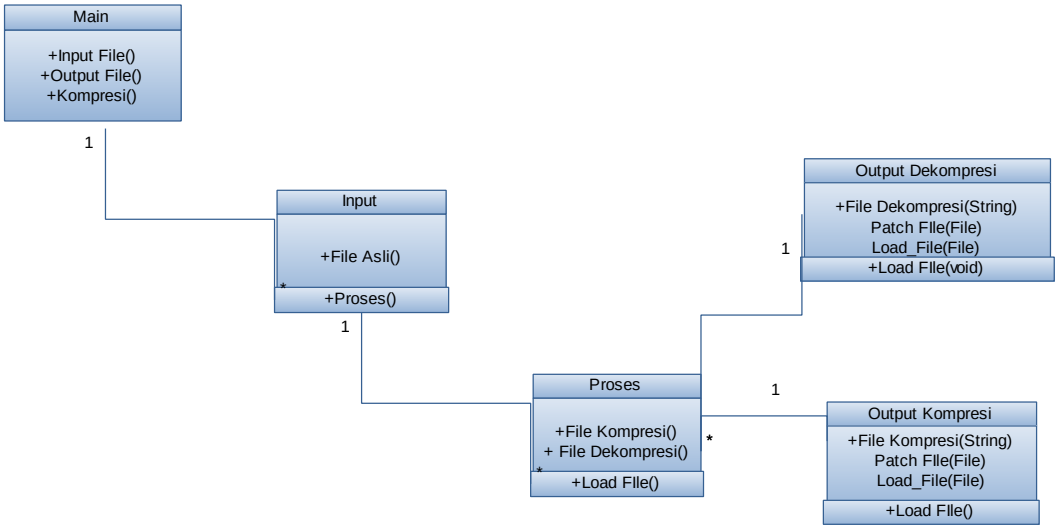


Gambar III.5. *Sequance Diagram* Pilihan Proses

Pengguna berinteraksi melalui pilihan proses yang ada pada *menu* utama, dapat dilihat pada *sequence diagram* di atas, pengguna memilih proses yang disediakan yaitu proses kompresi dan dekompresi. Setelah pilihan proses ditentukan oleh pengguna kembali pada *menu* utama.

III.2.4. *Class Diagram*

Pada *class diagram* berikut ini menggambarkan perilaku yang terjadi pada aplikasi yang diimplementasikan. Untuk lebih jelasnya dapat dilihat pada gambar III.6.



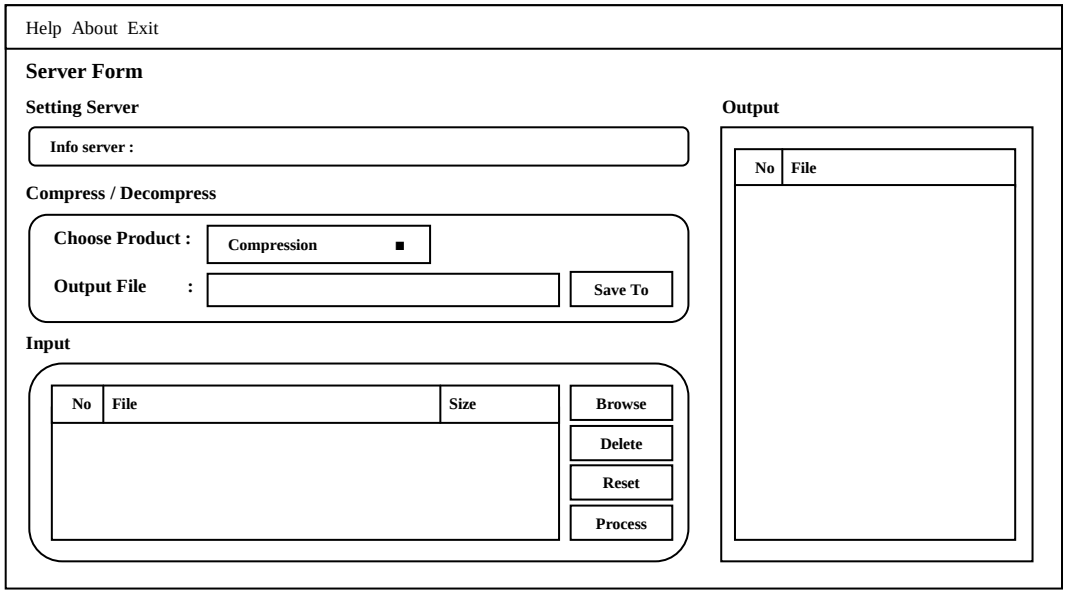
Gambar III.6. *Class Diagram* Pilihan Proses

III.3. Perancangan *Interface*

Perancangan *interface* adalah gambaran tampilan layar yang akan di desain, hal ini berguna agar proses perancangan dapat dilakukan sesuai desain yang telah dilakukan. Implementasi tampilan hasil program aplikasi yang telah dapat dijalankan harus sesuai dengan desain yang telah dibuat.

1. Rancangan Layar *Form* Utama *Server*

Pada halaman *form server* menampilkan *form* yang berfungsi sebagai media *server*, yang untuk dapat berkomunikasi dengan *client*, seperti gambar III.7.



Gambar III.7. *Form* Utama *Server*

Pada tampilan layar di atas adalah tampilan untuk pengguna menentukan lokasi *input* dan *output* serta melakukan proses, berikut ini cara-cara untuk menjalankan aplikasi dari tampilan layar utama *server* di atas :

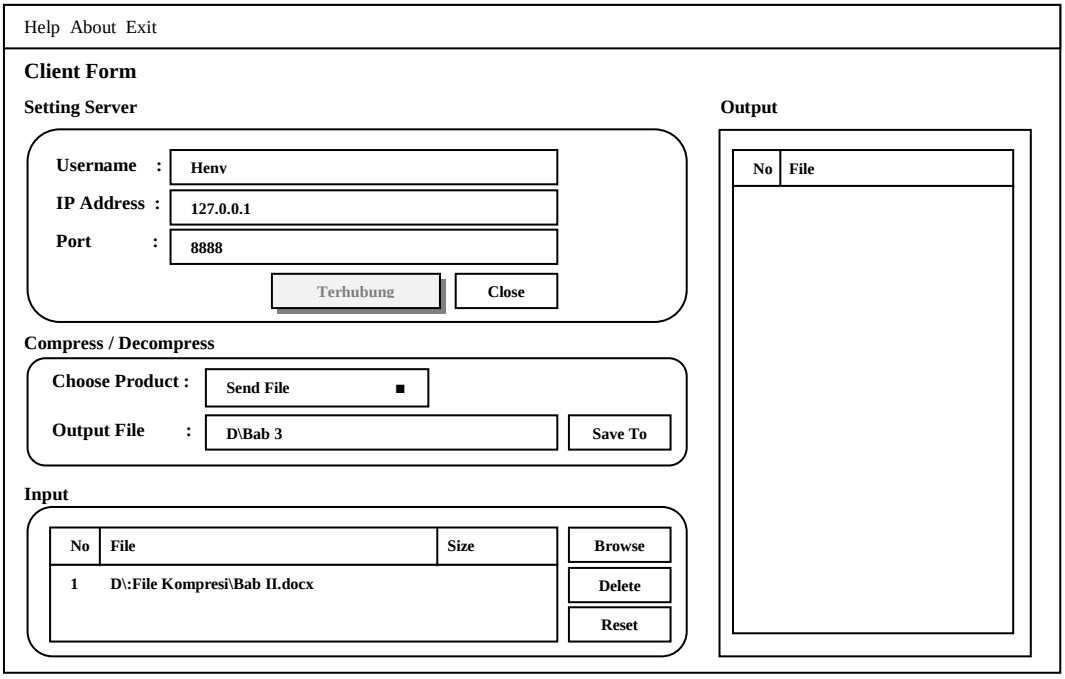
a. Tampilan layar ini tampil setelah jika pengguna menjalankan aplikasi, atau dengan nama yang sama yaitu sebagai *form server*.
b. Pengguna diharuskan menentukan proses apa yang ingin dilakukan dengan memilih pada pilihan ”*Chooses Procces*”, pilihan yang ada adalah ”Kompresi” untuk proses kompresi *file,* ”Dekompresi” untuk proses Dekompresi *file,* dan ”*Send File*” untuk proses pengiriman *file.*
c. Pengguna menentukan lokasi *output file* hasil proses dengan mengklik tombol ”*Save To*”.
d. Pengguna dapat memasukan *file-file* yang akan diproses ke dalam daftar *input file* dengan klik tombol ”*Browser*”, untuk menghapus daftar *input* pengguna dapat klik tombol ”*Delete*”, tombol *”Reset”* untuk menghapus

semua data pada *field.* Untuk melakukan proses pengguna dapat klik tombol ”*Process*”, prosedur ini terdapat pada panel ”*Input*”.

e. *Menu* yang ada diantaranya ”*Help*” untuk informasi bantuan penggunaan, ”*About*” untuk informasi aplikasi, dan ”*Exit*” untuk menutup aplikasi yang terdapat pada *form server* ini.

2. Rancangan Layar *Form* Utama *Client*

Pada halaman *form client* menampilkan *form* yang berfungsi digunakan oleh pengguna dari sisi *client*, yang untuk dapat terhubung dengan *server*, seperti gambar III.8.



Gambar III.8. *Form* Utama *Client*

Pada tampilan layar di atas adalah tampilan untuk pengguna menentukan lokasi *input* dan *output* serta melakukan proses, berikut ini cara-cara untuk menjalankan aplikasi dari tampilan layar utama *client* di atas :

a. Tampilan layar ini tampil setelah pengguna menjalankan aplikasi, atau dengan nama yang sama yaitu sebagai *form client*.

b. Pengguna diharuskan menentukan proses apa yang ingin dilakukan dengan memilih pada pilihan ”*Chooses Procces*”, pilihan yang ada adalah ”Kompresi” untuk proses kompresi *file,* ”Dekompresi” untuk proses Dekompresi *file,* dan ”*Send File*” untuk proses pengiriman *file.*

c. Pengguna menentukan lokasi *output file* hasil proses dengan mengklik tombol ”*Save To*”.

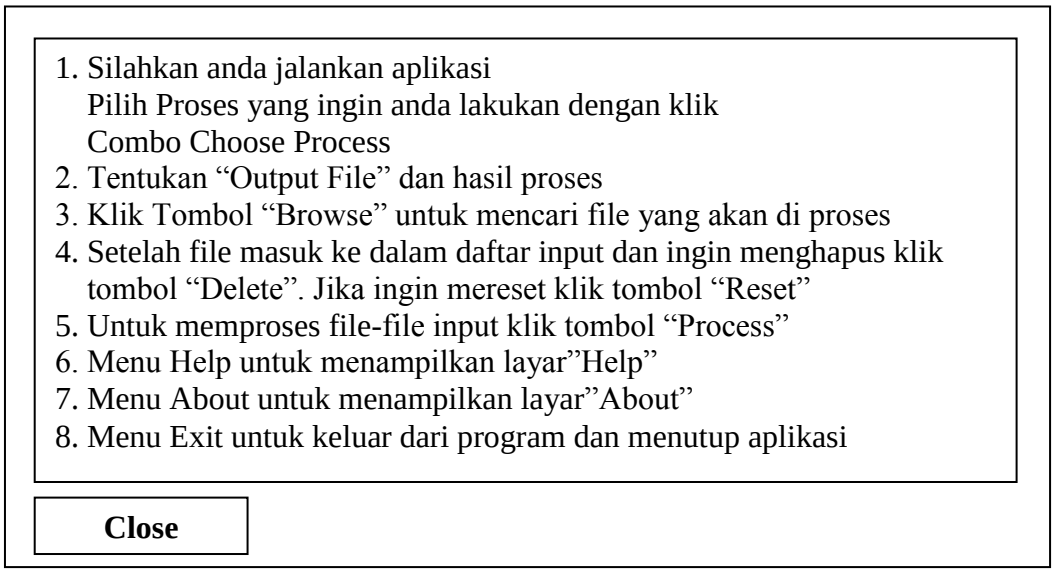
d. Pengguna dapat memasukan *file-file* yang akan diproses ke dalam daftar *input file* dengan klik tombol ”*Browser*”, untuk menghapus daftar *input* pengguna dapat klik tombol ”*Delete*”, tombol *”Reset”* untuk menghapus semua data pada *field.* Untuk melakukan proses pengguna dapat klik tombol ”*Process*”, prosedur ini terdapat pada panel ”*Input*”.

e. *Menu* yang ada diantaranya ”*Help*” untuk informasi bantuan penggunaan, ”*About*” untuk informasi aplikasi, dan ”*Exit*” untuk menutup aplikasi yang terdapat pada *form client* ini.

f. Untuk proses pengiriman, pengguna diharuskan mengisi data ”*IP Address Server*” dan ”*Port*” agar dapat terhubung dengan *server*. Kemudian memilih proses ”*Send File*” pada pilihan ”*Choose Procces*”. Pengguna menentukan lokasi penyimpanan *file* dengan mengklik tombol ”*Save To*”. Klik tombol ”*Connect”* agar *client* dapat terhubung dengan *server*. Dan setelah terhubung akhirnya *server* dan *client* dapat melakukan pengiriman *file.*

3. Rancangan Layar *Form Help*

Pada halaman *form help* menampilkan *form* yang berfungsi sebagai alat bantu pengguna dan panduan penggunaan, seperti gambar III.9.



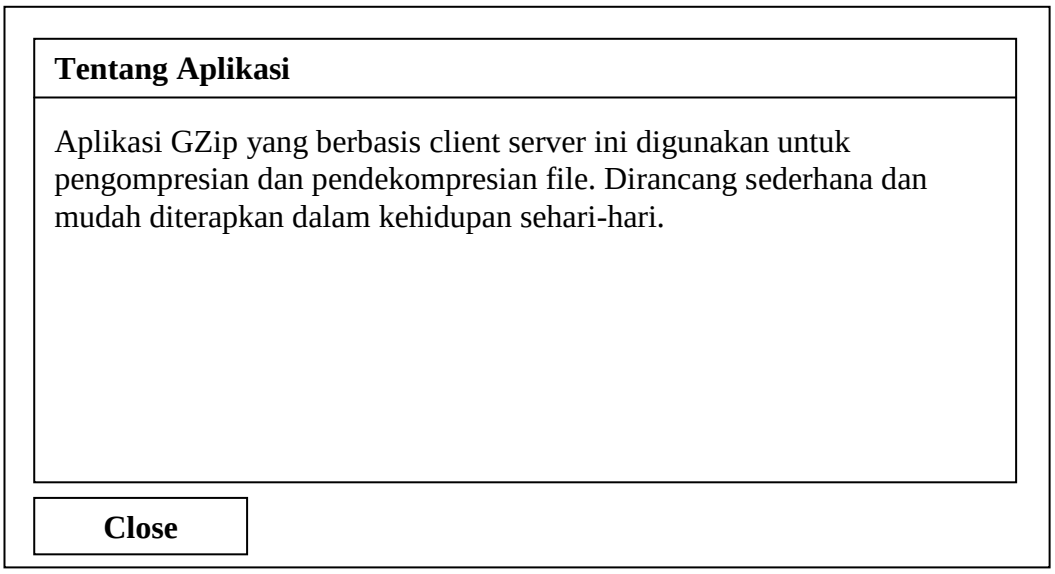
Gambar III.9. Rancangan Layar *Form Help*

Pada tampilan layar *help* pengguna dapat mengetahui informasi cara menggunakan aplikasi yang dirancang. Adapun cara agar dapat menampilkan layar *help* ini adalah sebagai berikut.

a. Pada layar *main* terdapat tombol *help,* klik tombol tersebut maka akan tampil *form "Help".*
b. Untuk menutup layar *help* pengguna dapat klik tombol "*Close*", pengguna akan kembali ke layar utama.

4. Rancangan Layar *Form About*

Pada halaman *form about* menampilkan *form* yang berfungsi sebagai informasi tentang aplikasi, seperti gambar III.10.



Gambar III.10. Rancangan Layar *Form About*

Pada tampilan layar *about* pengguna dapat mengetahui informasi dari tujuan perancangan aplikasi. Adapun cara agar dapat menampilkan layar *about* ini adalah sebagai berikut.

a. Pada layar utama terdapat tombol *about,* klik tombol tersebut maka akan tampil *form* "*About".*
b. Untuk menutup layar *about* pengguna dapat klik tombol "*Close*", pengguna akan kembali ke layar utama.