BAB III

ANALISIS DAN PERANCANGAN

3.1 Analisis Sistem

Analisis sistem merupakan penguraian dari suatu sistem yang utuh kedalam bagian – bagian komponennya dengan maksud untuk mengidentifikasikan dan mengevaluasi permasalahan sehingga permasalahan dapat dipahami dengan benar. Sehingga seorang analis dapat memahami secara keseluruhan tentang sistem yang akan dibangun.

3.1.1 Analisis Masalah

Permasalahan yang didapatkan dari hasil penelitian ini adalah membangun suatu aplikasi kompresi dengan menggunakan algoritma kompresi ZLIB yang dapat melakukan kompresi dan dekompresi pada saat data dikirimkan ke tujuan lain dalam lingkupan jaringan lokal (LAN).

Data dari hasil penelitian ini akan dibandingkan dengan dengan data aslinya pada proses kompresi dan dekompres serta membandingkan waktu yang dibutuhkan antara melakukan proses kompresi dan dekompresi dengan waktu yang dibutuhkan aplikasi untuk dapat melakukan proses kompresi dan dekompresi tersebut pada saat data dikirimkan dengan menggunakan aplikasi kompresi data.

3.1.2 Analisis dan kebutuhan fungsional

3.1.2.1 Actor Identification

Tahap pertama yang dilakukan untuk melakukan analisis berorientasi objek adalah menggunakan UML untuk menentukan *actor* yang berperan sebagai pengguna sistem. *Actor* dalam UML adalah untuk menampilkan peran sebagai pengguna (sesuatu diluar sistem yang dikembangkan yang dapat berupa perangkat keras, *end user*, sistem yang lain, dan sebagainya).

3.1.3 Analisis dan Kebutuhan non-fungsional

Analisis dan kebutuhan non-fungsional meliputi analisis dan kebutuhan pengguna, analisis dan kebutuhan perangkat keras, serta analisis dan kebutuhan perangkat lunak.

3.1.3.1 Analisis dan Kebutuhan pengguna (*user*)

Analisis pengguna diartikan sebagai orang yang mengakses dan yang akan menggunakan aplikasi kompresi algoritma ZLIB, dalam hal ini pengguna harus memiliki kemampuan dasar untuk mengoperasikan komputer dan memiliki pengetahuan dalam menggunakan aplikasi komputer.

3.1.3.2 Analisis dan Kebutuhan Perangkat Keras

Perangkat keras adalah seluruh komponen atau peralatan yang digunakan untuk membangun aplikasi kompresi ini. Analisis perangkat keras yang digunakan adalah :

Tabel 3.1 Spesifikasi Perangkat Keras

Processor	***Memory (RAM)***	***Hard-disk***	***Monitor (resolution)***
Intel Pentium Core 2 Duo processor 2,33 GHz	Kapasitas memory mulai dari 256 MB sampai 2 GByte	Kapasitas Hard-disk mulai dari 40 – 250 GByte	Pada umumnya monitor yang digunakan memiliki resolusi 1024 x 768 (32 bit) 60 Hz

3.1.3.3 Analisis dan Kebutuhan Perangkat Lunak

1. Sistem Operasi yang digunakan adalah Windows XP SP3
2. Perangkat Lunak yang digunakan untuk membangun aplikasi ini adalah *Visual C# .NET 2008.*

3.1.4 Analisa kompresi terhadap suatu *file* Teks

Analisa kompresi terhadap *file* teks pada Algoritma ZLIB diproses oleh gabungan antara algoritma *LZ77* dan algoritma *Huffman*. Dimana proses kompresi terlebih dahulu dilakukan oleh algoritma *LZ77* dan dilanjutkan dengan algoritma *Huffman*.

Proses kompresi pertama kali dilakukan dengan menggunakan algoritma *LZ77*, *file* yang akan dikompres akan dibaca dan hasilnya akan dimasukkan dalam sebuah variabel *string* yang dinamakan *frase*. Dari *frase* tersebut akan dibentuk dua buah *string* yaitu *Sliding Window (SW)* dan *Read Aheead (RA).* SW adalah sebuah *string* yang berisi rangkaian karakter sebesar n *byte* yang dimulai dari karakter awal *frase* sampai akhir. RA adalah sebuah *string* yang berisi rangkaian

karakter yang besarnya sama atau lebih kecil dari SW yang dimulai dari *index* akhir SW + 1. RA kemudian dibandingkan dengan SW. Jika sama, maka dibentuk *offset* dan *length* yang merupakan index awal dari *string* yang cocok dan panjang dari *string* yang cocok. Jika tidak sama, maka *index* akhir pada RA dikurangi 1 dan terus melakukan proses *string matching* sampai panjang RA = 2. Setelah semua proses *string matching* selesai dilakukan, maka perlu dibuat sebuah *prefix code* untuk menandai yang mana *literal* (karakter) dan *offset/length. Prefix code* akan disisipkan setiap 8 *bytes* agar lebih mudah untuk menghitungnya.

Contoh kompresi jika pada *file* yang akan dikompresi terdapat teks "AAAAAAAAAAABCABCAAAAA".

Sliding Window (SW) : AAAAAAAAAAABCA

Read Ahead (RA) : BCAAAAA

Isi RA tidak sama dengan dengan SW, maka *index* akhir RA dikurangi 1 menjadi BCAAAA. Jika masih tidak ditemukan maka *index* akhir RA akan dikurangi terus menjadi :

SW : AAAAAAAAAAABCA

RA : BCA

Setelah isi RA memiliki kesamaan dengan isi SW maka dicari nilai *offset* dan *length* nya yang terdiri dari 2 buah kode. Setelah itu *index* awal SW akan bergeser sebanyak isi dari RA.

Frase : AAAAAAAAAAABCABCAAAAA

Hasil : AAAAAAAAAAABCA22

Proses kompresi berhenti bila akhir *index* dari SW sama dengan panjang dari *frase*. Proses kemudian dilanjutkan dengan menambah *flag* pada setiap 8 *byte* pada variabel hasil dimana *flag* 0 untuk *literal* (huruf) dan *flag* 1 untuk kode (angka).

Frase : AAAAAAAAAAABCABCAAAAA

Hasil : AAAAAAAAABCA2293

Flag : 000000000000001111

Flag kemudian dipotong tiap 8 bit hingga menjadi 1 *byte* yang akan digunakan sebagai *prefix code.*

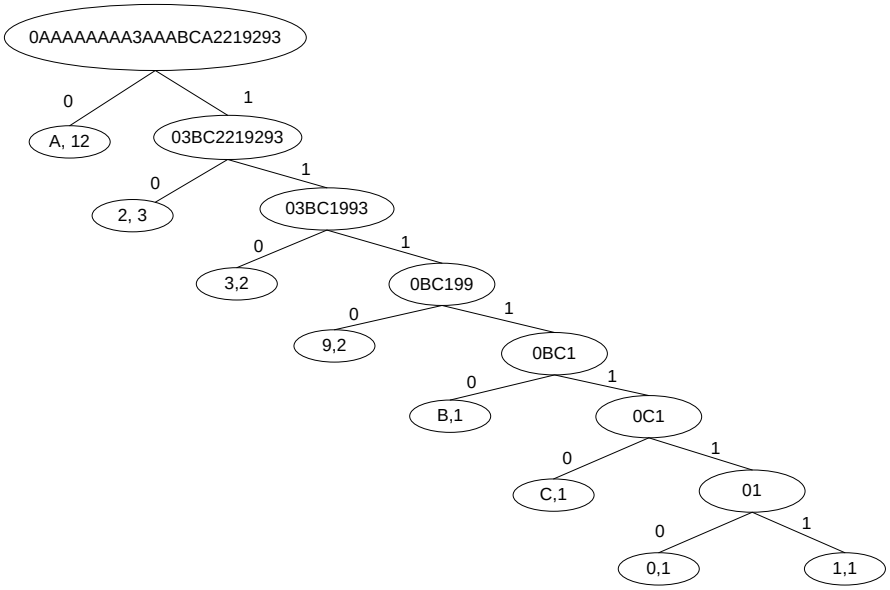
Flag : 00000000 00000011 11000000

Prefix code : 0 3 192

Prefix code yang diselipkan pada hasil kompresi menjadi :

0AAAAAAAA3AAABCA2219293

Hasil dari proses kompresi yang telah diselipkan *prefix code* ini akan 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 *value* diperoleh dari jumlah kemunculan karakter tersebut.



Gambar 3.1 Pohon *Huffman string* AAAAAAAAAAABCABCAAAAA

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 pada sebuah variabel ’*path*’ dan kemudian akan disimpan ke dalam variabel *dictionary* dapat dilihat pada tabel 3.2.

Gambar 3.2 Tabel Kode Huffman

Karakter	**Frekuensi**	**Peluang**	**Kode *Huffman***
A	12	12/21	0
3	2	2/21	10
2	2	2/21	110
9	1	1/21	1110
B	1	1/21	11110
C	1	1/21	111110
0	1	1/21	1111110
1	1	1/21	11111110

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

01011011101111011111011111011111110

Setelah kode *dictionary* telah diperoleh, maka langkah terakhir yaitu menyalin *dictionary* ke *file output*.

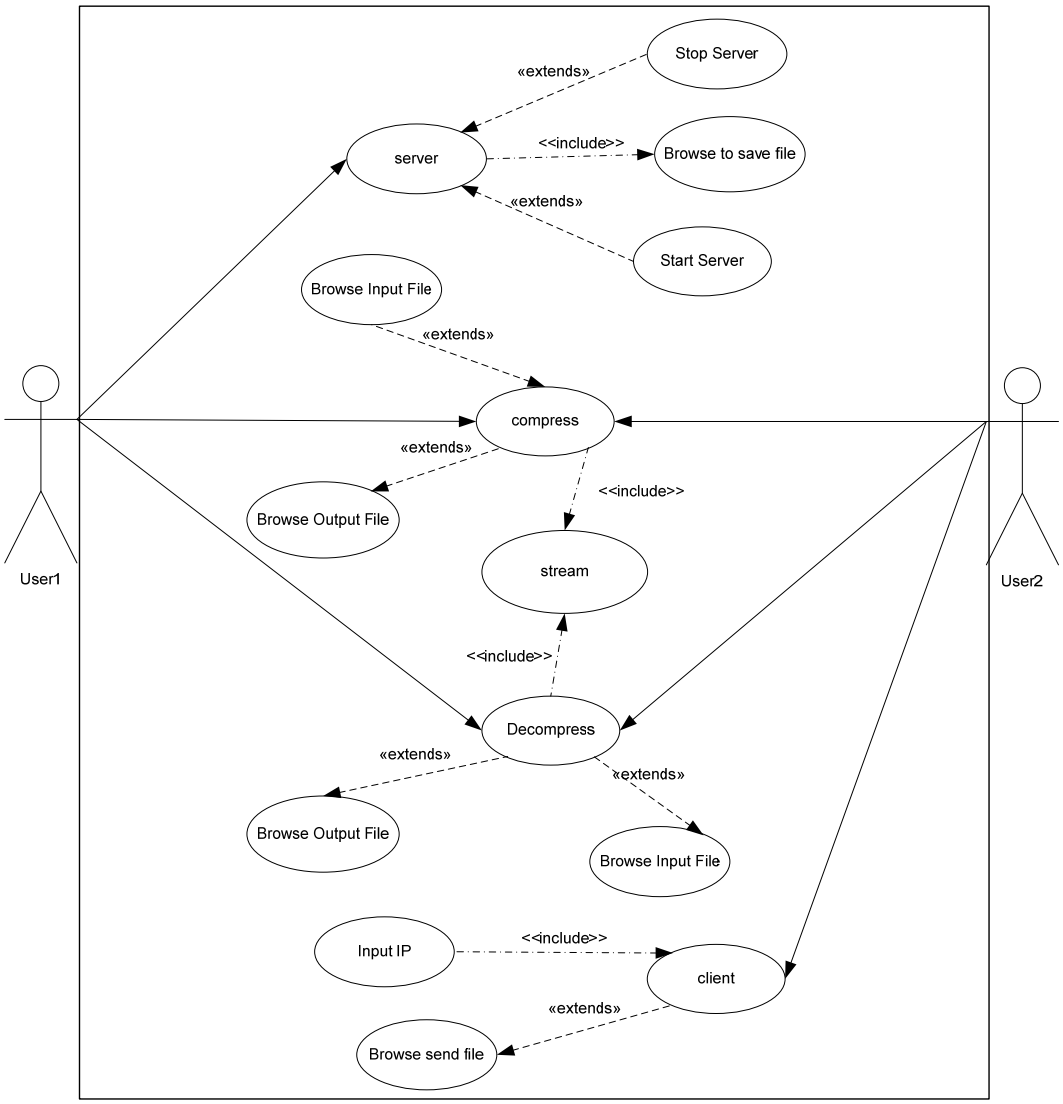
3.2 Use case Diagram

Use case diagram adalah konstruksi untuk mendeskripsikan bagaimana sistem akan terlihat dimata pengguna potensial. *Use case* terdiri dari sekumpulan skenario yang dilakukan oleh seorang actor (orang, perangkat keras, urutan waktu atau system yang lain). *Use case* dibuat berdasarkan ‘apa’ yang dikerjakan oleh software aplikasi, bukan ‘bagaimana’ *software* aplikasi mengerjakannya. *Use case*

yang akan dihasilkan pada aplikasi ini dapat menjalani 2 proses secara terpisah, yaitu :

1. Seorang *user* dapat melakukan kompresi data secara terpisah, yaitu *user* hanya dapat melakukan kompresi data saja ataupun *user* dapat melakukan dekompresi data saja tanpa perlu mengirimkan kepada *user* lain.
2. Seorang *user* dapat melakukan proses pengiriman *file* kepada *user* lain, dimana sebelum melakukan proses pengiriman kedua *user* harus menjadi *server* dan lainnya menjadi *client*. Pada prosesnya, data yang sedang dikirimkan terjadi proses kompresi dan dekompresi didalamya. Sehingga data yang samapai pada *user* yang menerima dapat langsung dibuka tanpa harus melakukan dekompresi lagi.

Use case pada gambar 3.1 menggambarkan proses yang terjadi pada aplikasi kompresi data pada saat transfer data serta bagaimana pengirim melakukan interaksi.



Gambar 3.2 *use case* diagram aplikasi kompresi data dan transfer data

Sistem ini melayani dua aktor, yaitu *user*1 dan *user*2. Transaksi yang dapat dilakukan adalah salah satu dari seorang *user* dapat menjadi seorang *server* yang menerima *file* yang dikirimkan dan *user* yang satu nya lagi menjadi seorang *client* yang mengirimkan *file* kepada *server*.

3.2.1 Skenario *Use case*

Skenario setiap bagian pada *use case* menunjukkan proses apa yang terjadi pada setiap bagian didalam *use case* tersebut, dimana *user* memberikan perintah pada setiap bagian dan respon apa yang diberikan oleh sistem kepada *user* setelah *user* memberikan perintah pada setiap bagian – bagian *use case*.

3.2.1.1 *Server*

Identifikasi	
Nomor	1
Nama	*Server*
Tujuan	Memastikan status seorang *user* menjadi seorang *server*
Deskripsi	Sistem menerima permintaan seorang *user* untuk dapat menjadi *server* agar dapat menerima *file* yang akan dikirimkan.
Tipe	
Aktor	*User*1
Skenario Utama	
Kondisi awal	Aplikasi kompresi dalam keadaan baru dibuka

Aksi Aktor	**Reaksi Sistem**
Memberikan *request* menjadi *server* dengan menekan tombol *start server*	Sistem meListen permintaan *user* dan melakukan pembukaan *port*.
-	Sistem meminta *user* menentukan tempat penyimpanan *file* yang akan diterima.
Menentukan tempat penyimpanan *file* yang akan diterima.	Sistem menetapkan tempat penyimpanan *file* yang telah dipilih oleh *user*. Jika OK, sistem menampilkan status *user*.
-	Sistem mengirim *file* yang dikirimkan oleh *Client*.
Server menerima *File* yang telah dikirimkan oleh *Client*	Sistem memberikan status pengiriman pengiriman.

Memberikan *request* kepada sistem untuk menyudahi status sebagai *server*.	Sistem *meListen* permintaan *user* dan melakukan penutupan *port*.
Skenario Alternatif – Autentikasi Gagal	
Aksi Aktor	**Reaksi Sistem**
Memberikan *request* menjadi *server* dengan menekan tombol *start server*	
Kondisi akhir	Hak akses *user* yang akan berperan sebagai *server* sudah dipastikan

3.2.1.2 *Start Server*

Identifikasi	
Nomor	2
Nama	*Start Server*
Tujuan	Menjadikan *user* seorang *Server*
Deskripsi	Sistem membuka *port* 5656 agar *user* menjadi *server*
Tipe	
Aktor	*User*1
Skenario Utama	
Kondisi awal	Aplikasi kompresi dalam keadaan baru dibuka
Aksi Aktor	**Reaksi Sistem**
Memberikan *request* menjadi *server* dengan menekan tombol *start server*	Sistem *meListen* permintaan *user* dan melakukan pembukaan *port* 5656.
Skenario Alternatif – Autentikasi Gagal	
Aksi Aktor	**Reaksi Sistem**
Memberikan *request* menjadi *server* dengan menekan tombol *start server*	
Kondisi akhir	*User* akan menjadi *server*

3.2.1.3 *Browse to save file*

Identifikasi		
Nomor	3	
Nama	*Browse to save File*	
Tujuan	Memilih tempat *server* menyimpan *file* yang diterima oleh *client*	
Deskripsi	Sistem meminta menentukan tempat penyimpanan *file* yang dipilih *server*	
Tipe		
Aktor	*User*1	
Skenario Utama		
Kondisi awal	Aplikasi kompresi dalam keadaan baru dibuka	
Aksi Aktor		**Reaksi Sistem**
Memberikan *request* menjadi *server* dengan menekan tombol *start server*		Sistem meListen permintaan *user* dan melakukan pembukaan *port*.
-		Sistem meminta *user* menentukan tempat penyimpanan *file* yang akan diterima.
Menentukan tempat penyimpanan *file* yang akan diterima.		Sistem menetapkan tempat penyimpanan *file* yang telah dipilih oleh *user*. Jika OK, sistem menampilkan status *user*.
Skenario Alternatif – Autentikasi Gagal		
Aksi Aktor		**Reaksi Sistem**
Memberikan *request* menjadi *server* dengan menekan tombol *start server*		
Kondisi akhir	Tempat penyimpanan *file* yang diterima telah dipilih	

3.2.1.4 *Stop Server*

Identifikasi		
Nomor	4	
Nama	*Stop Server*	
Tujuan	Menghentikan status *user* sebagai *server*	
Deskripsi	Sistem menerima permintaan seorang *user* untuk berhenti menjadi *server* dan menutup *port* 5656	
Tipe		
Aktor	*User*1	
Skenario Utama		
Kondisi awal	Aplikasi kompresi dalam keadaan baru dibuka	
Aksi Aktor		**Reaksi Sistem**
Memberikan *request* kepada sistem untuk menyudahi status sebagai *server*.		Sistem meListen permintaan *user* dan melakukan penutupan *port*.
Skenario Alternatif – Autentikasi Gagal		
Aksi Aktor		**Reaksi Sistem**
Memberikan *request* menjadi *server* dengan menekan tombol *start server*		
Kondisi akhir	Hak *user* sebagai *server* akan berakhir	

3.2.1.5 *Compress*

Identifikasi	
Nomor	5
Nama	*Compress*
Tujuan	*User* menerima *file* yang telah dikompresi
Deskripsi	Sistem menerima masukkan *file* yang diinputkan oleh *user* dan melakukan proses kompresi serta proses *encoding*.
Tipe	
Aktor	*User*

Skenario Utama	
Kondisi awal	Aplikasi kompresi dalam keadaan baru dibuka
Aksi Aktor	**Reaksi Sistem**
Menginputkan *file* yang akan dikompresi.	Meminta *user* memasukkan nama dan tempat *file* hasil kompresi akan disimpan
Memasukkan nama *file* kompresi dan memilih tempat *file* kompresi akan disimpan.	Melakukan kompresi *file* menggunakan Algoritma *LZ77* dengan memilih *string* – *string* didalam *file* yang sama.
	String – *string* yang sama yang telah ditentukan oleh Algoritma *LZ77* pada sistem akan dilanjutkan ke proses *encoding* yang dilakukan oleh Algoritma *Huffman* dengan mengubah *string* – *string* tersebut kedalam format biner.
	String yang telah dirubah kedalam format biner tersebut akan diberikan *header file* oleh bagian steram pada system
Menerima *file* yang telah terkompresi. *File* kompresi tersebut akan memiliki ekstensi *file* .zkomp.	Memberikan informasi bahwa kompresi telah berhasil.
Skenario Alternatif – Autentikasi Gagal	
Aksi Aktor	**Reaksi Sistem**
Menginputkan *file* yang akan dikompresi	
Kondisi akhir	*User* akan menerima *file* yang telah terkompresi

3.2.1.6 *Browse input File*

Identifikasi	
Nomor	6, 10
Nama	*Browse Input File*
Tujuan	*User* menentukan *file* yang akan dikompresi dan yang akan didekompresi
Deskripsi	Sistem menerima *file* yang dipilih *user* dan melakukan kompresi atau

	dekompresi	
Tipe		
Aktor	*User*	
Skenario Utama		
Kondisi awal	1. Aplikasi kompresi dalam keadaan baru dibuka 2. Terdapat *file* kompresi	
Aksi Aktor		**Reaksi Sistem**
Memilih dan menginputkan *file* yang akan dikompresi.		Meminta *user* memasukkan nama dan tempat *file* hasil kompresi akan disimpan
Menginputkan *file* zkomp		Meminta *user* memasukkan nama *file* hasil dekompresi dan tempat *file* hasil dekompresi akan disimpan
Skenario Alternatif – Autentikasi Gagal		
Aksi Aktor		**Reaksi Sistem**
Menginputkan *file* yang akan dikompresi		
Kondisi akhir	1. *User* telah memilih *file* yang akan dikompresi 2. *User* telah memilih *file* yang akan didekompresi	

3.2.1.7 *Browse Output File*

Identifikasi	
Nomor	7, 11
Nama	*Browse Output File*
Tujuan	Menentukan tempat *file* hasil kompresi dan hasil dekompresi akan disimpan
Deskripsi	Sistem meminta dimana *file* yang telah dikompresi ataupun *file* hasil dekompresi akan disimpan
Tipe	
Aktor	*User*
Skenario Utama	
Kondisi awal	1. Aplikasi kompresi dalam keadaan baru dibuka

	2. Terdapat *file* kompresi
Aksi Aktor	**Reaksi Sistem**
Menginputkan *file* yang akan dikompresi atau didekompresi.	Meminta *user* memasukkan nama dan tempat *file* hasil kompresi akan disimpan
Memasukkan nama *file* kompresi dan memilih tempat *file* kompresi akan disimpan.	Melakukan kompresi *file* menggunakan Algoritma *LZ77* dengan memilih *string – string* didalam *file* yang sama.
Memasukkan nama *file* dekompresi dan memilih tempat *file* dekompresi akan disimpan.	Pembacaan *length* pada *file* kompresi.
Skenario Alternatif – Autentikasi Gagal	
Aksi Aktor	**Reaksi Sistem**
Menginputkan *file* yang akan dikompresi	
Kondisi akhir	*User* telah memilih tempat *file* hasil kompresi akan disimpan

3.2.1.8 Stream

Identifikasi	
Nomor	8
Nama	Stream
Tujuan	Memberikan *header file*
Deskripsi	Sistem akan memberikan *header file* pada *file* komperi yang telah di*encoding* oleh Algoritma *Huffman*.
Tipe	
Aktor	*User*
Skenario Utama	
Kondisi awal	*File* telah di*encoding* oleh Algoritma *Huffman*
Aksi Aktor	**Reaksi Sistem**
Memasukkan *file* yang telah di*encoding*	Sistem memberikan *header – header* pada *file* yang telah di *encoding* oleh *Huffman*.
	Header file akan dikelompokkan menjadi

	sebuah *dictionary* pada *file*.
Memasukkan *file* yang akan didekompresi.	Sistem akan menghilangkan *header file* pada *file* kompresi dan juga akan mengembalikan *header* pada *file* asli.
Skenario Alternatif – Autentikasi Gagal	
Aksi Aktor	**Reaksi Sistem**
Memasukkan *file* yang telah di*encoding*	
Kondisi akhir	1. *File* asli akan menjadi *file* kompresi 2. *File* kompresi akan menjadi *file* asli

3.2.1.9 *Decompress*

Identifikasi	
Nomor	9
Nama	*Decompress*
Tujuan	*User* menerima *file* asli
Deskripsi	Sistem menerima masukkan *file* zkomp yang diinputkan oleh *user* dan akan melakukan proses *decoding*.
Tipe	
Aktor	*User*
Skenario Utama	
Kondisi awal	Terdapat *file* kompresi .zkomp
Aksi Aktor	**Reaksi Sistem**
Menginputkan *file* zkomp	Meminta *user* memasukkan nama *file* hasil dekompresi dan tempat *file* hasil dekompresi akan disimpan
Memasukkan nama *file* dekompresi dan memilih tempat *file* dekompresi akan disimpan.	Pembacaan *length* pada *file* kompresi.
	Melakukan proses *decoding* pada *file* kompresi dengan menggunakan algoritma *Huffman*.

	Melakukan proses dekompresi pada *file* yang telah di*decoding* oleh Algoritma *Huffman*. Proses dekompresi *file* akan dilakukan oleh Algoritma *LZ77*
	Menghilangkan *file header* pada *file* yang telah di*decoding* Algoritma *Huffman* oleh bagian stream pada sistem.
Menerima *file* yang telah didekompresi. *File* hasil dekompresi tersebut akan menjadi *file* asal.	Memberikan informasi bahwa kompresi telah berhasil.
Skenario Alternatif – Autentikasi Gagal	
Aksi Aktor	**Reaksi Sistem**
Menginputkan *file* yang akan didekompresi	
Kondisi akhir	*User* akan menerima *file* asli.

3.2.1.10 *Client*

Identifikasi	
Nomor	12
Nama	*Client*
Tujuan	Mengirimkan *file* kepada *Server*
Deskripsi	Pada saat aplikasi kompresi dibuka seorang *user* otomatis akan menjadi seorang *Client* dan ser tersebut dapat mengirimkan *file* kepada *server*.
Tipe	
Aktor	*User*2
Skenario Utama	
Kondisi awal	*Server* Aktif
Aksi Aktor	**Reaksi Sistem**
Memasukkan *file* yang akan dikirimkan kepada *server*.	Meminta *client* memasukkan alamat IP *server*.
Memasukkan alamat IP *server*	Memeriksa alamat IP *server* aktif atau tidak.

	Memberikan informasi status pengiriman pada *client*.
Skenario Alternatif – Autentikasi Gagal	
Aksi Aktor	**Reaksi Sistem**
Memasukkan *file* yang akan dikirimkan kepada *server* jika *server* sedang tidak aktif atau alamat IP yang dimasukkan salah.	Memberikan informasi bahwa pengiriman gagal.
Kondisi akhir	1. *File* yang dikirimkan akan berhasil jika *server* aktif 2. *File* yang dikirmkan akan gagal jika *server* tidak aktif atau alamat IP yang dimasukkan tidak ada.

3.2.1.11 *Browse Send File*

Identifikasi	
Nomor	13
Nama	*Browse Send File*
Tujuan	Memilih *file* yang akan dikirimkan ke *server*
Deskripsi	Mengirimkan *file* yang dipilih *client* untuk dikirimkan ke *server*
Tipe	
Aktor	*User*2
Skenario Utama	
Kondisi awal	*Server* Aktif
Aksi Aktor	**Reaksi Sistem**
Memasukkan *file* yang akan dikirimkan kepada *server*.	Meminta *client* memasukkan alamat IP *server*.
Skenario Alternatif – Autentikasi Gagal	
Aksi Aktor	**Reaksi Sistem**
Memasukkan *file* yang akan dikirimkan kepada *server*.	Memberikan informasi bahwa pengiriman gagal.
Kondisi akhir	*File* yang telah dipilih akan dikirim

3.2.1.12 *Input* IP

Identifikasi		
Nomor	14	
Nama	*Input* IP	
Tujuan	Menentukan alamat IP yang akan dikirimin *file*	
Deskripsi	Sistem akan mencari alamat IP yang dimasukkan dan mencocokkannya	
Tipe		
Aktor	*User*2	
Skenario Utama		
Kondisi awal	*Server* Aktif	
Aksi Aktor		**Reaksi Sistem**
Memasukkan alamat IP *server*		Memeriksa alamat IP *server* aktif atau tidak.
Skenario Alternatif – Autentikasi Gagal		
Aksi Aktor		**Reaksi Sistem**
Alamat IP yang dimasukkan salah.		Memberikan informasi bahwa pengiriman gagal.
Kondisi akhir	*File* yang dikirmkan akan gagal jika alamat IP yang dimasukkan tidak ada.	

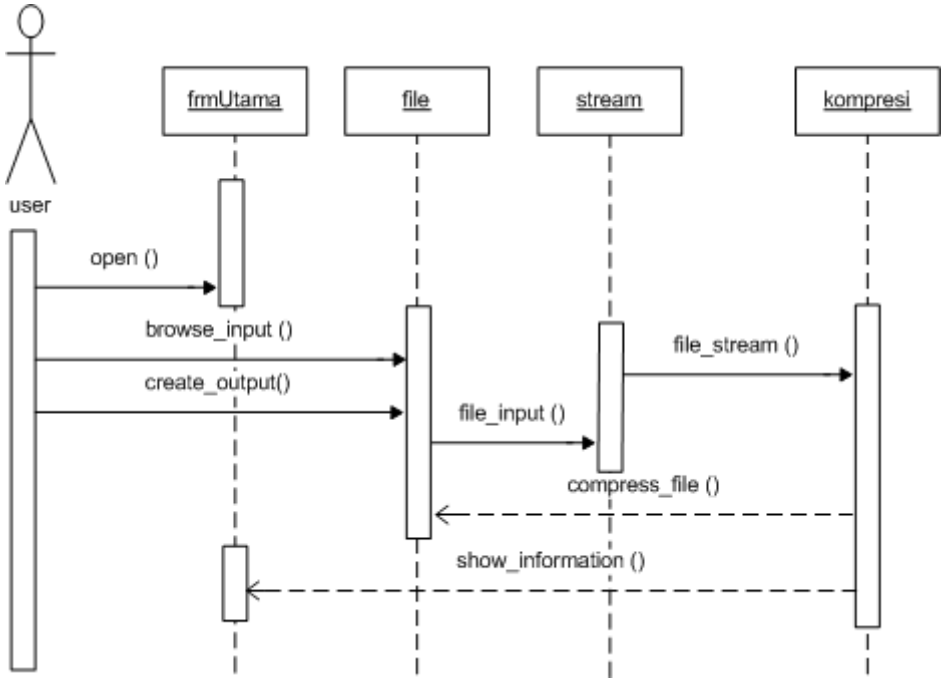
3.3 Sequence Diagram

Data sequence diagram menggambarkan bagaimana suatu operasi itu dilakukan sebagai respons dari sebuah *event* untuk menghasilkan *output* tertentu, berupa *message* (pesan) apa yang dikirim dan kapan pelaksanaannya. Diagram ini diatur berdasarkan waktu. Obyek – obyek yang berkaitan dengan proses berjalannya operasi diurutkan dari kiri ke kanan berdasarkan waktu terjadinya dalam pesan yang terurut.

Sequence diaram yang digambarkan dalam perancangan sistem ini adalah sebagai berikut.

3.3.1 Sequence Diagram *Compress*

Sequence diagram *compress* pada aplikasi ditunjukkan pada gambar 3.2 :



Gambar 3.3 *Sequence diagram compress*

Pada sequence diagram *compress*, skenario yang terjadi pada bagian kompresi *file* yaitu:

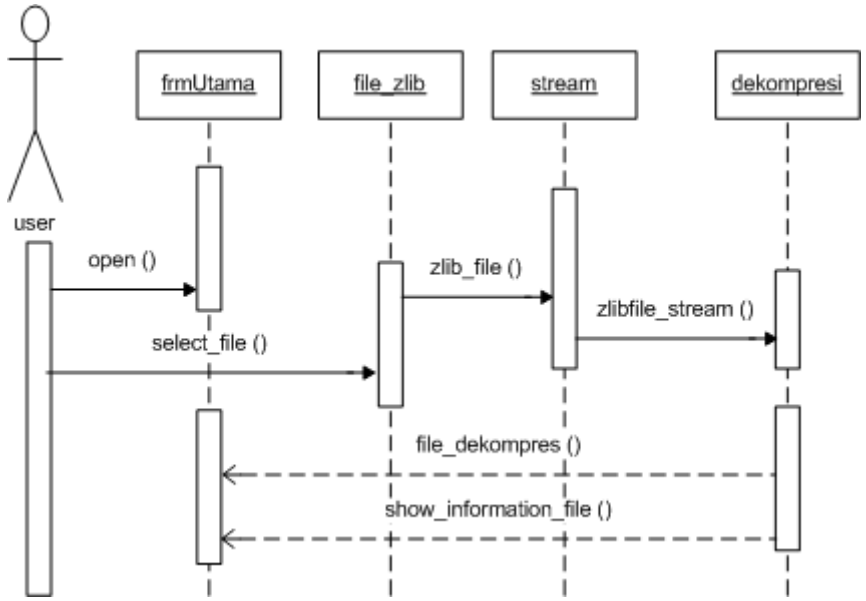
1. *User* membuka program aplikasi.
2. Sebelum melakukan kompresi sistem akan meminta *user* untuk mencari *file* yang akan dikompresi.
3. Setelah *file* yang akan dikompresi telah ditentukan oleh *user*, setelah itu *user* diminta untuk mebuat *file* kompresi dengan cara *user* diminta memasukkan nama *file* hasil kompresi atau *file* yang telah dikompresi

disimpan sesuai dengan nama *file* asliny serta menentukan tempat penyimpanan *file* hasil kompresi.

4. Setelah proses penginputan *file* dan penamaan *file* hasil kompresi telah selesai. Maka *file* yang telah dipilih tersebut akan masuk kedalam proses stream.
5. Setelah proses stream dilakukan oleh objek stream, maka hasil keluaran dari proses stream tersebut akan dilakukan proses kompresi.
6. Setelah proses kompresi selesai dilakukan maka hasil keluaran dari proses kompresi ini adalah *file* yang telah dikompresi. Setelah proses ini selesai maka *user* akan menerima *file* yang telah dikompresi.
7. Setelah *user* menerima *file* yang telah dikompresi maka sistem akan memberikan informasi berupa ukuran *file* serta waktu yang dibutuhkan sistem untuk melakukan kompresi tersebut. Setelah informasi ini muncul, maka proses kompresi yang dilakukan oleh sistem telah selesai.

3.3.2 Sequence Diagram *Decompress*

Sequence diagram *decompress* pada aplikasi ditunjukkan pada gambar 3.3 :



Gambar 3.4 ***Sequence diagram decompress***

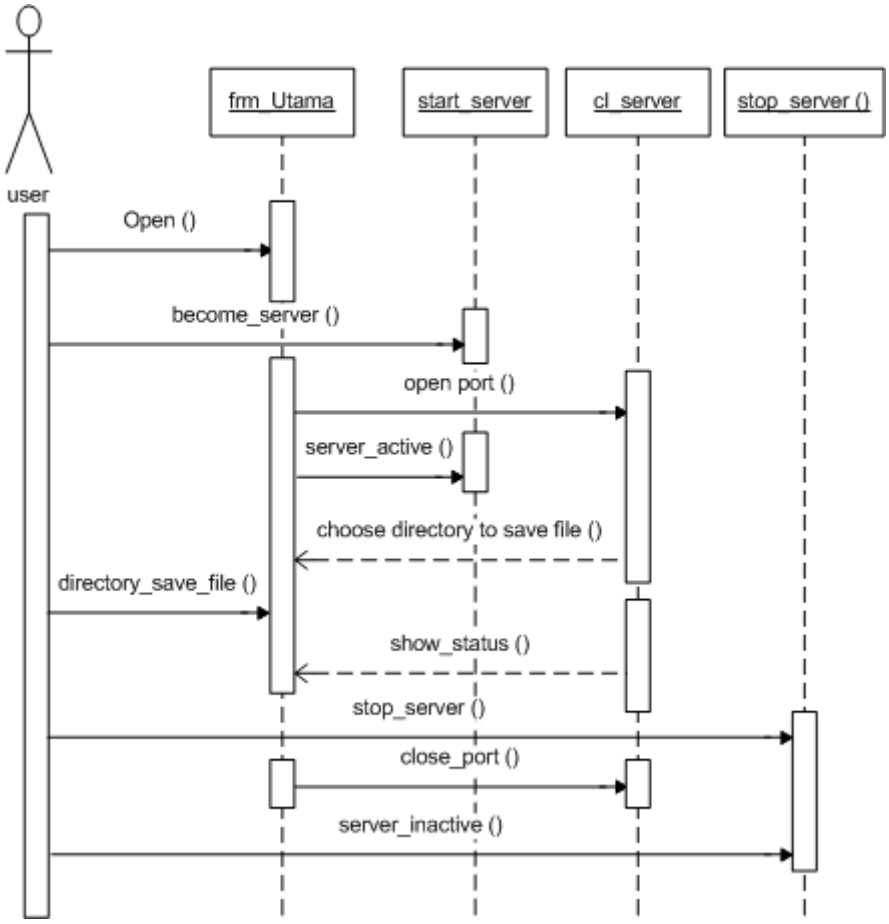
Pada sequence diagram *decompress*, skenario yang terjadi pada bagian dekompresi *file* yaitu :

1. *User* membuka program aplikasi
2. Sebelum melakukan dekompresi data, sistem akan meminta kepada *user* untuk memasukkan *file* kompresi yang akan dilakukan proses dekompresi serta tempat menyimpan *file* yang telah didekompresi.
3. Setelah *file* kompresi telah dipilih oleh *user*, maka sistem akan melakukan proses stream pada *file* kompresi tersebut ke objek stream.
4. Setelah proses stream *file* kompresi selesai dilakukan oleh objek stream, maka keluaran dari proses stream ini akan dilanjutkan ke objek dekompresi

untuk dilakukan proses dekompresi *file* kompresi agar *file* kompresi tersebut kembali ke *file* asal.

5. Setelah proses dekompresi ini selesai dilakukan oleh objek dekompresi, maka selanjutnya sistem akan memberikan informasi berupa ukuran *file* dan waktu yang diperlukan sistem untuk melakukan dekompresi *file* tersebut.

3.3.3 Sequence Diagram *Server*



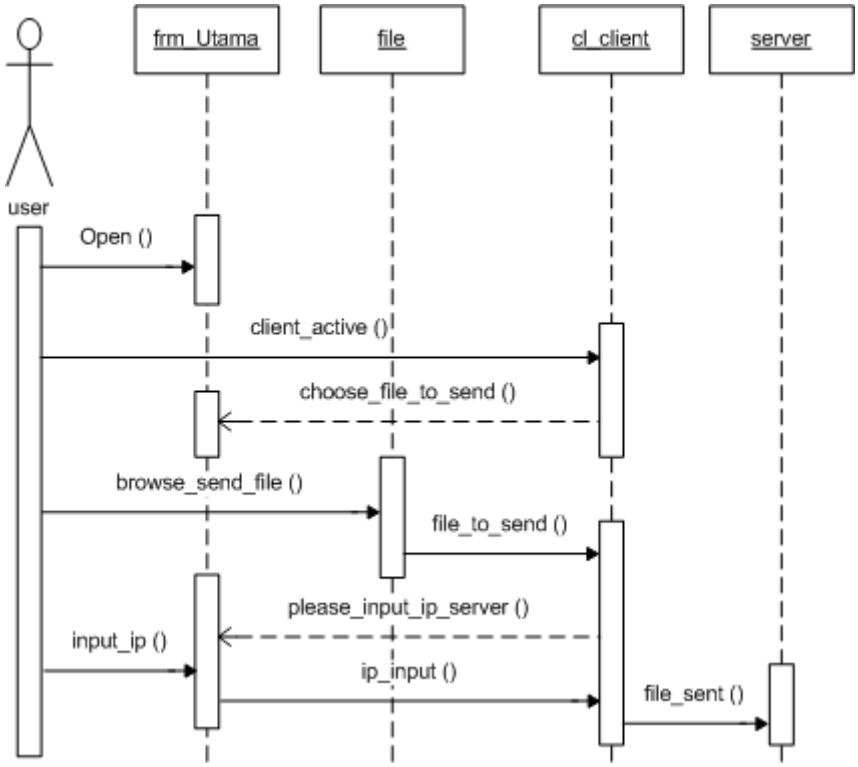
Gambar 3.5 *Sequence diagram server*

Pada sequence diagram *server*, skenario yang terjadi pada bagian proses *server* yaitu :

1. *User* membuka program aplikasi.
2. Agar seorang *user* dapat menjadi *server*, maka *user* tersebut terlebih dahulu harus menjadi *server* dengan melakukan perintah menjadi *server* pada aplikasi.
3. Setelah *user* melakukan *start server*, maka sistem akan melakukan proses open *port* kepada objek cl_*server*. Dimana *port* yang digunakan pada aplikasi ini yaitu *port* 5656.
4. Setelah *port* dibuka oleh objek cl_*server* maka *user* tersebut telah menjadi *server*.
5. Setelah *user* menjadi *server*, maka objek cl_*server* akan meminta kepada *user* melalui objek frm_Utama untuk menentukan dimana tempat *server* akan menerima *file* yang nanti akan dikirimkan oleh *client*.
6. *User* yang telah menjadi *server* memilih tempat penyimpanan *file* yang akan diterima.
7. Setelah *user* memilih tempat penyimpanan *file* yang akan diterima, maka objek cl_*server* akan memberikan informasi bahwa *server* telah aktif dan siap menerima *file*.
8. Apabila seorang *user* ingin mengakhiri status nya sebagai seorang *server*, maka *user* tersebut dapat mengakhiinya dengan melakukan proses stop *server* pada objek *stop_server*.

9. Proses mengakhiri *server* ini dilakukan objek frm_utama dengan memberikan perintah *close_port* kepada cl_*server*.
10. Setelah perintah *close_port* tersebut dilakukan oleh objek cl_*server*, maka status *user* tersebut sebagai *server* telah berakhir dan *user* tersebut tidak dapat menerima *file* dari *client*.

3.3.4 Sequence Diagram *Client*



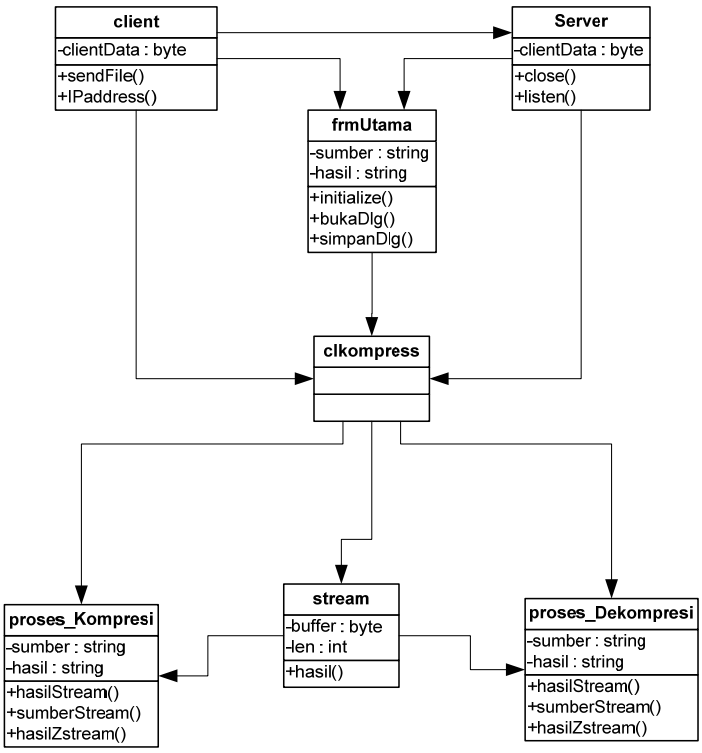
Gambar 3.6 Sequence diagram *client*

Pada sequence diagram *client*, skenario terjadi pada bagian proses *client* yaitu :

1. *User* membuka program aplikasi
2. Setiap *user* membuka program aplikasi, maka secara otomatis *user* tersebut telah bertindak sebagai seorang *client*.
3. Apabila *user* ingin mengirim *file* kepada *server* maka sistem akan meminta kepada *client file* mana yang akan dikirimkan kepada *server*.
4. *Client* memilih *file* yang akan dikirimkan.
5. Setelah *file* telah dipilih, maka *file* tersebut akan dikirimkan ke objek cl_*client*.
6. Setelah *file* masuk ke objek cl_*client*, maka objek cl_*client* ini akan meminta kepada *user* untuk memasukkan alamat ip *server*.
7. Agar pengiriman dapat berlangsung maka *client* diminta memasukkan ip *server* yang sesuai.
8. Setelah ip dimasukkan maka *file* yang telah dipilih tersebut akan dikirimkan oleh objek cl_*client* ke alamat ip tujuannya, disini diartikan alamat ip yang dituju adalah alamat ip *server*.

3.4 Class Diagram

Class Diagram memberikan pandangan secara luas dari suatu sistem dan memberikan pelayanan untuk menyelesaikan keadaan tersebut.



Gambar 3.7 *Class Diagram*

Class Diagram diatas menunjukkan bahwa aplikasi ini terdiri dari 5 *class* yang saling terhubung antara *class* yang satu dengan *class* yang lainnya :

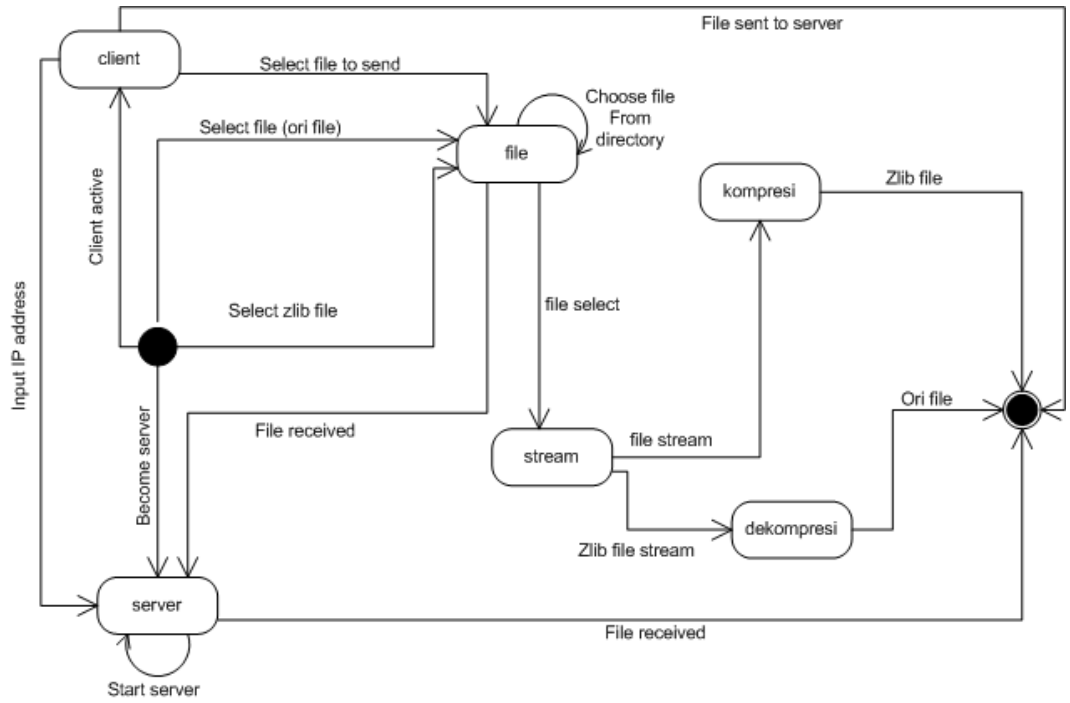
1. *Class Client* : *class client* pada aplikasi digunakan untuk melakukan proses send *file* dan proses untuk memasukkan IP Addres. *Class Client* ini terhubung dengan *Class* frmUtama yang merupakan *class* yang dimana segala proses *class* yang ada di aplikasi ini terhubung, *Class Client* ini jg terhubung dengan *Class Server* dimana *Client Server* ini akan menerima *File* yang dikirim oleh *Class Client.*
2. *Class Server* : *class server* pada aplikasi ini digunakan untuk melakukan proses menghidupkan dan mematikan *server,* karena agar *class server* ini dapat menerima *file* dari *client* maka *server* harus dihidupkan terlebih

dahulu. *Class server* ini terhubung dengan *class client* agar dapat menerima *file* yang dikirimkan dari *client*. *Class server* ini jg terhubung dengan *class* frmUtama.

3. *Class* frmUtama : *Class* ini merupakan *class* utama pada aplikasi ini, dimana pada *class* ini semua *class* yang membangun aplikasi ini dihubungkan. *Class* ini terhubung dengan *class client*, *class server* dan *class* clkompress.
4. *Class* clkompress : *Class* ini merupakan *class* untuk melakukan proses kompresi data. Dimana *class* ini terbagi menjadi tiga sub *class*, yaitu *class* proses_kompresi, *class* stream dan *class* proses_dekompresi.

3.5 State Chart

State Chart diagram digunakan untuk memodelkan perilaku dinamis dari kelas individual atau jenis objek lainnya. State chart diagram menampilkan urutan dari *state – state* yang dialami oleh objek, event – event yang menyebabkan transisi dari sebuah state atau aktifitas ke state atau aktiviti lainnya, dan aksi – aksi yang merupakan hasil dari perubahan sebuah state atau aktiviti. Setiap *state* merepresentasikan sebuah nama kondisi dari sebuah objek selama memenuhi suatu kondisi atau menunggu suatu *event*.



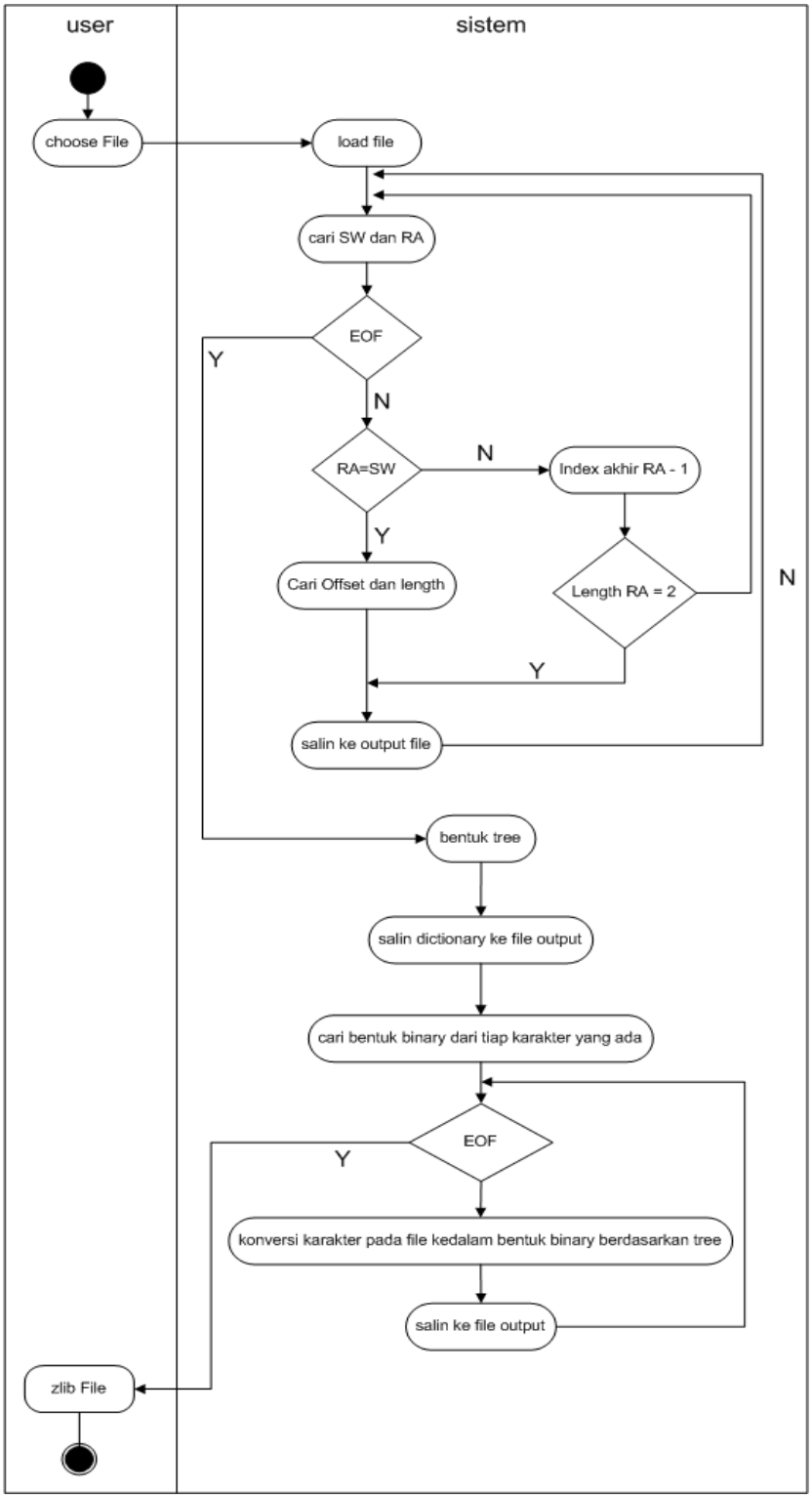
Gambar 3.8 *State Chart* diagram aplikasi kompresi

State chart diagram diatas menunjukkan status yang dilakukan oleh objek serta perilaku yang dilakukan oleh objek tersebut. Dalam aplikasi ini, event yang menyebabkan final state bisa tercapai adalah pada saat status yang diberikan oleh sistem adalah status pada saat *file* telah dikompresi, status pada saat *file* kembali seperti semula (*file original*). Sedangkan jika *state* awal adalah proses menjadi *client* atau pun *server*, final state yang dicapai adalah jika sistem telah memberikan status *file received* pada *server* dan *file* sent pada *client*. Sedangkan perlakuan yang terjadi pada objek *file* yaitu *file* selalu diteruskan pada objek stream sebelum *file* tersebut nantinya akan dilakukan kompresi dan dekompresi.

3.6 Activity Diagram

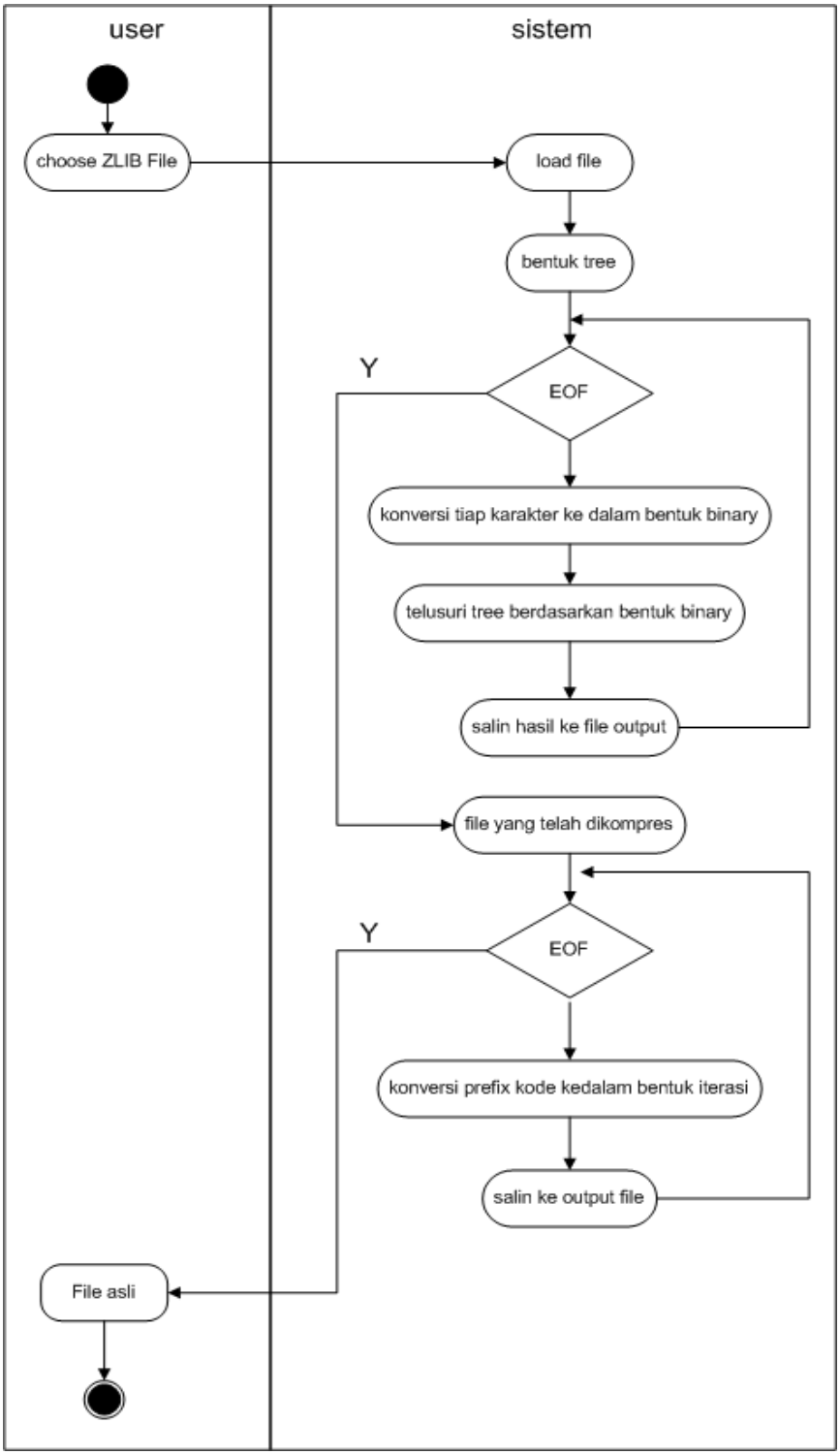
Activity Diagram menggambarkan berbagai alir aktifitas didalam sistem yang sedang dirancang, tentang bagaimana masing – masing alir berawal, *decision* yang mungkin terjadi dan yang berakhir. *Activity diagram* ini berhubungan dengan diagram *statechart*, *activity diagram* berfokus pada aktifitas – aktifitas yang terjadi yang terkait dalam suatu proses tunggal. Jadi dengan kata lain, diagram ini menunjukkan bagaiman aktifitas – aktifitas tersebut bergantung satu sama lain. Gambar 3.10 dan gambar 3.11 menunjukkan *activity diagram* dari setiap alur algoritma yang dilakukan aplikasi.

3.6.1 *Activity Diagram* Proses *encoding* Algoritma Zlib



Gamabr 3.9 *Activity diagram* proses *encoding* algoritma Zlib

3.6.2 *Activity Diagram* Proses *decoding* Algoritma Zlib



Gambar 3.10 *Activity diagram* proses *decoding* algoritma Zlib

3.7 Perancangan Sistem

Perancangan sistem adalah suatu proses yang menggambarkan bagaimana suatu sistem dibangun untuk memenuhi kebutuhan pada fase analisis. Adapun tahapan yang dilakukan dalam perancangan sistem ini membahas mengenai tujuan perancangan sistem, dan perancangan antar muka.

3.7.1 Tujuan Perancangan Sistem

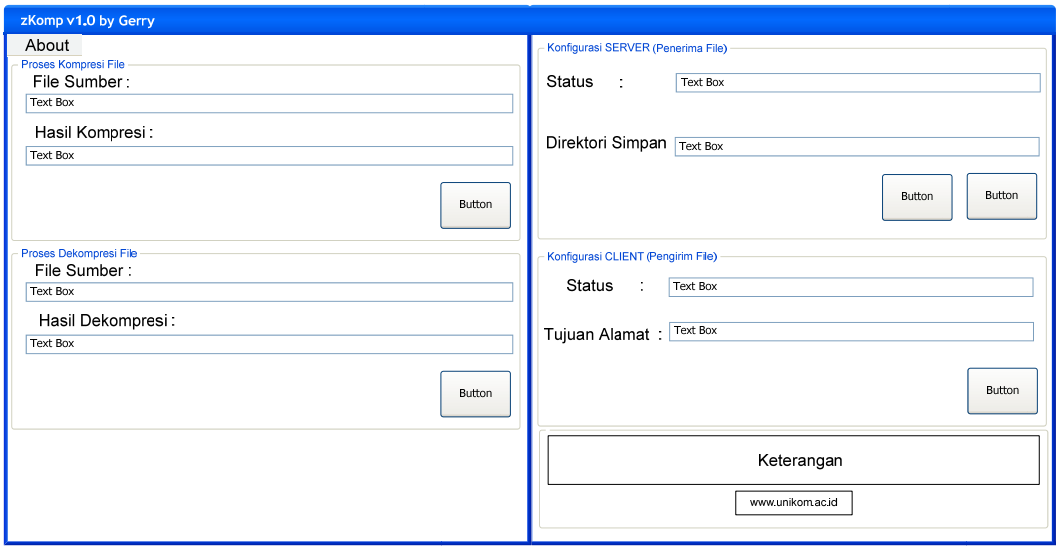
Perancangan sistem merupakan tindak lanjut dari tahap analisa. Perancangan sistem bertujuan untuk memberikan gambaran sistem yang akan dibuat. Dengan kata lain perancangan sistem didefinisikan sebagai penggambaran atau pembuatan sketsa dari beberapa elemen yang terpisah kedalam satu kesatuan yang utuh dan berfungsi. Selain itu juga perancangan bertujuan untuk lebih mengarahkan sistem yang terinci, yaitu pembuatan perancangan yang jelas dan lengkap yang nantinya akan digunakan untuk pembuatan simulasi. Aplikasi kompresi ini dibuat dengan sederhana, sehingga diharapkan *user* dapat dengan mudah menggunakan aplikasi kompresi ini.

3.7.2 Perancangan Antar Muka

Perancangan antarmuka dibutuhkan untuk mewakili keadaan sebenarnya dari aplikasi yang akan dibangun, berikut akan disajikan perancangan antarmuka dari aplikasi yang akan dibangun:

3.7.2.1 Antar muka Utama

Antar muka utama aplikasi digunakan untuk melakukan proses kompresi dan dekompresi *file*. Gambar 3.10 menampilkan rancangan antar muka utama aplikasi kompresi zlib.



Gambar 3.11 Rancangan antar muka utama aplikasi

Tabel 3.3 Keterangan tampilan antar muka utama

No Objek	Jenis Objek	Fungsi Objek
1	GroupBox kompres	Untuk mencantumkan objek – objek kompresi
2	*File* Sumber	Untuk mencantumkan tulisan *File* Sumber
3	Text Box	Untuk memasukkan *file* yang akan dikompres
4	Hasil Kompresi	Untuk mencantumkan tulisan Hasil Kompresi
5	Text Box	Untuk memasukkan hasil *file* yang telah dikompres

6	Button	Untuk melakukan proses kompresi
7	*File* Sumber	Untuk mencantumkan tulisan *File* Sumber
8	Text Box	Untuk memasukkan tempat *file* yang akan didekompresi
9	Hasil Dekompres	Untuk mencantumkan tulisan Hasil Delompresi
10	Text Box	Untuk memasukkan tempat hasil *file* yang telah di Dekompresi
11	Button	Untuk melakukan proses dekompresi
12	Text Box	Untuk memberitahukan keadaan *server* sedang aktif atau tidak
13	Text Box	Menunujukkan direktori tempat *server* menyimpan *file* yang dikirim *client*
14	Button	Untuk mengaktifkan *server*
15	Button	Untuk mematikan *server*
16	Keterangan	Untuk mencantumkan keterangan tentang aplikasi
17	Link label	Untuk mencantumkan tulisan www.unikom.ac.id