PERANCANGAN DAN ANALISIS KOMPRESI *AUDIO* WAV DENGAN MENGGUNAKAN METODE *HUFFMAN*

TUGAS AKHIR

Diajukan untuk memenuhi salah satu mata kuliah Tugas Akhir pada jurusan Teknik Informatika di STMIK- IM

Oleh :

Eko Surahman
(NIM : 361501013)

PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TINGGI MANAGEMEN INFORMATIKA DAN KOMPUTER

INDONESIA MANDIRI

BANDUNG

2020

LEMBAR PENGESAHAN

PERANCANGAN DAN ANALISIS KOMPRESI *AUDIO* WAV DENGAN MENGGUNAKAN METODE *HUFFMAN*

DESIGN AND ANALYSIS COMPRESSION AUDIO WAV USING HUFFMAN METHOD

Oleh
Eko Surahman
361501013

Tugas Akhir ini telah diterima dan disahkan untuk
memenuhi persyaratan mencapai gelar
Sarjana Teknik Informatika

Pada

PROGRAM STUDI TEKNIK INFORMATIKA SEKOLAH TINGGI
MANAJEMEN INFORMATIKA & KOMPUTER INDONESIA MANDIRI

Bandung, 29 September 2020
Disetujui Oleh

Ketua Program Studi	Pembimbing
Chalifa Chazar, S.T., M.T.	Patah Herwanto, S.T., M.Kom.
NIDN: 0421098704	NIDN: 0027107501

LEMBAR PERSETUJUAN REVISI

PERANCANGAN DAN ANALISIS KOMPRESI *AUDIO* WAV DENGAN MENGGUNAKAN METODE *HUFFMAN*

Oleh
Eko Surahman
361501013

Telah melakukan sidang tugas akhir dan telah melakukan revisi sesuai dengan perubahan dan perbaikan yang diminta pada saat sidang tugas akhir.

Bandung, 9 Oktober 2020
Menyetujui

No	Nama Dosen	Keterangan	Tanda Tangan
1.	Patah Herwanto, S.T., M.Kom.	Pembimbing	
2.	Chairuddin, Ir., M.T., M.M., Dr.	Penguji 1	
3.	Haryoso wicaksono, S.SI., M.M., M.Kom.	Penguji 2	

Mengetahui

Ketua Program Studi Teknik Informatika

Chalifa Chazar, S.T., M.T.
NIDN: 0421098704

SURAT PERNYATAAN

Dengan ini saya menyatakan bahwa :

1) Tugas akhir ini adalah asli dan belum pernah diajukan untuk mendapatkan gelar akademik. Baik di Sekolah Tinggi Manajemen Informatika dan Komputer Indonesia Mandiri maupun perguruan tinggi lainnya.
2) Tugas akhir ini murni merupakan karya penelitian saya sendiri dan tidak menjiplak karya pihak lain. Dalam hal ada bantuan atau arahan dari pihak lain maka telah saya sebutkan identitas dan jenis bantuannya di dalam lembar ucapan terimakasih.
3) Seandainya ada karya pihak lain yang ternyata memiliki kemiripan dengan karya seni saya ini, maka hal ini adalah diluar pengetahuan saya dan terjadi tanpa kesengajaan dari pihak saya.

Pernyataan ini saya buat dengan sesungguhnya dan apabila dikemudian hari terbukti adanya kebohongan dalam pernyataan ini, maka saya bersedia menerima sanksi akademik sesuai norma yang berlaku di Sekolah Tinggi Manajemen.

Bandung , 29 September 2020

Yang membuat penyataan

Eko surahman

361501013

ABSTRAK

PERANCANGAN DAN ANALISIS KOMPRESI *AUDIO* WAV DENGAN MENGGUNAKAN METODE *HUFFMAN*

Oleh

Eko Surahman
361501013

Permasalahan yang dihadapi oleh pengguna teknologi informasi salah satunya adalah ukuran *file* yang besar sehingga diperlukan suatu media penyimpanan yang besar serta diperlukan waktu yang cukup lama untuk menyimpannya. Solusi untuk masalah diatas adalah dengan melakukan pemapatan/ kompresi. Ada banyak sekali metode kompresi yang ada saat ini, salah satunya adalah metode kompresi *Huffman*. Metode ini menggunakan prinsip dengan memperkecil bit yang sering muncul dan memperbesar bit yang jarang muncul. Kompresi dilakukan dengan cara membuat pohon biner *Huffman*. Analisis metode algoritma *Huffman* ini bertujuan untuk membuat aplikasi yang dapat mengkompresi *file audio* WAV dengan menggunakan metode *Huffman*. Serta untuk mengetahui hasil *file audio* WAV setelah dikompresi. Dalam pengembangannya aplikasi kompresi ini menggunakan metode *waterfall* dimana ia merupakan metode yang umum digunakan dalam pengembangan *software*. Dari hasil pengujian proses kompresi didapat bahwa rasio mempunyai besaran antara 65 % untuk nilai terkecil dan 94% untuk nilai tertinggi. Jika dicari hasil rasio kompresi tersebut secara rata-rata adalah sebesar 82,85%. Ini berarti ukuran *file* hasil adalah 0,8285 kali ukuran *file*. Dengan rata-rata pengurangan *file* sekitar 17,15%. *Audio* WAV yang telah di kompresi dengan algoritma *Huffman* ini dapat di kembalikan lagi dengan cara melakukan dekompresi.

Kata Kunci *: Audio*, *Huffman*, Kompresi,Dekompresi, WAV, *Waterfall*.

ABSTRACT

DESIGN AND ANALYSIS COMPRESSION AUDIO WAV USING HUFFMAN METHOD

Oleh

Eko Surahman
361501013

One of the problems faced by information technology users is the large file size so that a large storage medium is needed and it takes a long time to store it. The solution to the problem above is to do compression. There are many compression methods available today, one of which is the Huffman compression method. This method uses the principle of reducing the frequently occurring bits and increasing the rarely occurring bits. Compression is done by creating a Huffman binary tree. The analysis of the Huffman algorithm method aims to create an application that can compress WAV audio files using the Huffman method. And to know the results of the WAV audio file after compression. In its development, this compression application uses the waterfall method where it is a method commonly used in software development. From the test results of the compression process, it is found that the ratio has a magnitude between 65% for the smallest value and 94% for the highest value. If you look for the compression ratio, the average result is 82.85%. This means that the resulting file size is 0.8285 times the file size. With an average file reduction of about 17.15%. WAV audio that has been compressed with the Huffman algorithm can be returned again by decompressing it.

Keywords*: Audio, Huffman, Compression, Decompression, WAV, Waterfall.*

UCAPAN TERIMAKASIH

Puji Syukur kehadirat Allah SWT yang maha pengasih dan maha penyayang karena dengan rahmat, karunia serta taufik dan hidayah-Nya penulis dapat menyelesaikan tugas akhir ini. Laporan penelitian tugas akhir ini di ajukan untuk memenuhi dan melengkapi salah satu syarat akademik dalam kelulusan jenjang Strata Satu (S1) jurusan Teknik Informatika pada Sekolah Tinggi Manajemen Informatika dan Komputer Indonesia Mandiri.

Penulis sadari tugas akhir ini tidak akan selesai tanpa doa, dukungan dan dorongan dari berbagai pihak. Adapun dalam kesempatan ini penulis ingin mengucapkan banyak terima kasih kepada:

1. Bapak Patah Herwanto. S.T., M.Kom. selaku Dosen pembimbing yang selalu meluangkan waktu, fikiran dan tenaga dalam memberikan bimbingan, masukan dan saran-sarannya.
2. Bapak Dr. Chairudin, M.T., M.M. selaku Ketua Sekolah Tinggi Manajemen Informatika dan Komputer Indonesia Mandiri (STMIK-IM).
3. Ibu Chalifa Chazar, S.T., M.T. selaku Ketua program studi Teknik Informatika Sekolah Tinggi Manajemen Informatika dan Komputer Indonesia Mandiri (STMIK-IM).
4. Seluruh Dosen, Staff dan Karyawan Sekolah Tinggi Manajemen Informatika dan Komputer Indonesia Mandiri (STMIK-IM) yang telah mendidik dan membantu

dalam memberikan informasi serat motivasi dalam proses studi maupun tugas akhir berlangsung.

5. Teruntuk kedua orang tua penulis Bapak Sugandi dan Ibu Susilawati tercinta, orang yang paling hebat didunia ini, orang yang selalu tidak pantang menyerah dalam memberikan doa, bantuan, dukungan, kasih sayang, pengorbanan dan semangat di setiap langkah pejalanan penulis dalam menuntut ilmu, sekaligus orang yang banyak mengetahui keluh kesahku pada saat menyusun tugas akhir ini.
6. Sahabat-sahabat seperjuangan bimbingan Ahmedi Firdaus, Febriyanto orang yang tidak pernah mengeluh dan sama-sama berjuang demi mendapatkan hasil terbaik dalam penulisan tugas akhir ini.
7. Sahabat-sahabat penulis yaitu Ahmad Riyadi, Febriyanto, Ahmedi Firdaus yang sama-sama berjuang untuk menyelesaikan penulisan tugas akhir ini.
8. Teman-temanku Firmansyah, Rudi, Agung dharma, Imam Maulana, Widia Lestari yang telah memberikan semangat untuk tidak menyerah dalam penulisan tugas akhir ini.
9. Seluruh rekan STMIK-IM angkatan 2015 khususnya pada program studi Teknik Informatika yang sama-sama berjuang untuk terus meraih impian, yang saling membantu juga memberikan semangat, dan seringkali menjadi tempat sharing, baik yang berhubungan dengan materi perkuliahan maupun lainnya.
10. Seluruh rekan sahabat, dan pihak-pihak yang tidak dapat penulis sebutkan satu persatu yang telah membantu Penulis baik secara langsung maupun tidak langsung

memberikan semangat kepada Penulis dalam menyelesaikan laporan penelitian tugas akhir.

Penulis menyadari bahwa masih banyak kekurangan yang mendasar pada laporan penelitian tugas akhir ini. Oleh karena itu penulis mengundang pembaca untuk memberikan saran serta kritik yang dapat membangun penulis. Penulis berharap adanya kritik konstruktif dan saran yang membangun dari semua pihak.

Akhir kata saya, berharap semoga dengan selesainya laporan penelitian Tugas Akhir ini dapat memberikan manfaat bagi semua pihak serta menambah wawasan bagi pemikiran kita semua. Terimakasih.

KATA PENGANTAR

Puji Syukur kehadirat Allah SWT yang maha pengasih dan maha penyayang karena dengan rahmat, karunia serta taufik dan hidayah-Nya sehingga penulis dapat menyelesaikan laporan penelitian tugas akhir ini dengan baik.

Tugas akhir ini berjudul "*PERANCANGAN DAN ANALISIS KOMPRESI AUDIO WAV DENGAN MENGGUNAKAN METODE HUFFMAN*" disusun untuk melengkapi tahapan akhir studi yang dijalani di Sekolah Tinggi Manajemen Informatika dan Komputer Indonesia Mandiri.

Tugas akhir ini berisi mengenai perancangan sebuah aplikasi sederhana untuk melakukan kompresi dan dekompresi untuk *file audio* berektensi WAV.

Dengan segala keterbatasan tentunya diharapkan aplikasi ini dapat bermanfaat bagi berbagai pihak, khususnya bagi penulis sendiri.

Bandung , 29 September 2020

Penulis

Eko surahman

361501013

DAFTAR ISI

LEMBAR PENGESAHAN ..... **i**
LEMBAR KETERANGAN REVISI ..... **ii**
SURAT PERNYATAAN ..... **ii**
ABSTRAK ..... **iv**
ABSTRACT ..... **v**
UCAPAN TERIMAKASIH ..... **vi**
KATA PENGANTAR ..... **ix**
DAFTAR ISI ..... **x**
DAFTAR GAMBAR ..... **xiii**
DAFTAR TABEL ..... **xv**
BAB I PENDAHULUAN 1
1.1. Latar Belakang 1
1.2. Identifikasi Masalah 2
1.3. Tujuan Penulisan 2
1.4. Batasan Masalah 3
1.5. Metode Penelitian 3
1.5.1. Metode Pengumpulan Data 3
1.5.2. Metode Pengembangan Perangkat Lunak 3
1.6. Sistematika Penulisan 5
BAB II LANDASAN TEORI 7
2.1. Kompresi Data 7
2.2. Dekompresi Data 9
2.3. Algoritma 9
2.4. Algoritma *Huffman* 10
2.5. Rasio Kompresi 11
2.6. *Audio* 11
2.7. Reprentasi Suara 12

2.8. *File* WAV 13
2.9. Metode Perancangan 14
2.10. *Flowchart* 16
2.11. UML (*Unified Modelling Language*) 21
2.12. *Testing* 27
2.12.1. Pengertian *Testing* 27
2.12.2. *Behavioral Test* (*Black-Box Test*) 28
BAB III ANALISA MASALAH DAN PERANCANGAN PROGRAM 29
3.1. *Communication* 29
3.1.1. Pengumpulan Data 29
3.1.1.1. Studi Literatur 29
3.1.2. Analisis Sistem 31
3.1.3. Analisis Identifikasi Masalah 31
3.1.4. Analisis Kebutuhan 32
3.1.4.1. Analisis Perangkat Keras 33
3.1.4.2. Analisis Perangkat Lunak 33
3.1.5. Analisis Proses Kompresi Algoritma *Huffman* 33
3.1.6. Analisis Proses Dekompresi Algoritma *Huffman* 41
3.2. *Planning* 43
3.3. *Modelling* 44
3.3.1. Perancangan Sistem 45
3.3.1.1. *Use Case Diagram* 45
3.3.2. Desain Atar muka 54
3.3.2.1. Rancangan Antarmuka Halaman Utama 54
3.3.2.2. Rancangan Antarmuka Halaman Kompresi 55
3.3.2.3. Rancangan Antarmuka Halaman Dekompresi 56
3.3.2.4. Rancangan Antarmuka Halaman *About* 57
BAB IV IMPLEMENTASI DAN UJI COBA 58
4.1. *Construction (Code & Test)* 58
4.1.1. Implemetasi *Hardware & Software* 58

4.1.1.1. Perangkat Keras (*Hardware*) 58
4.1.1.2. Perangkat Lunak (*Software*) 58
4.1.2. Implementasi Algoritma 59
4.1.2.1. Implementasi Proses kompresi 59
4.1.2.2. Implemtasi Proses Dekompresi 61
4.1.3. Implemetasi Antar Muka 63
4.1.4. *Testing* 67
4.1.4.1. *Testing Black Box* 67
4.1.4.2. *Testing* Proses kompresi 71
4.1.4.3. *Testin*g Proses Dekompresi 72
4.1.4.4. Hasil Pengujian Proses Kompresi 73
4.1.4.5. Hasil Pengujian Proses Dekompresi 75
4.1.5. Hasil Analisis 76
BAB V PENUTUP 77
5.1. Kesimpulan 77
5.2. Saran 77
DAFTAR PUSTAKA 79
LISTING PROGRAM

DAFTAR GAMBAR

GAMBAR : 1.1. Model *Waterfall* (Pressman, 2015 : 42).....4

GAMBAR : 2.1. Alur gelombang suara (Binanto, 2010).....12

GAMBAR : 2.2. Model *Waterfall* (Pressman, 2015 : 42).....14

GAMBAR : 3.1. Urutan Sampel secara *Ascending*.....35

GAMBAR : 3.2. Penggabungan dua buah simpul (*node*).....36

GAMBAR : 3.3. Pemindahan dua buah simpul urutan sampel secara *Ascending*....36

GAMBAR : 3.4. Penggabungan dua buah simpul 37

GAMBAR : 3.5. Pemindahan dua buah simpul urutan secara *Ascending* 37

GAMBAR : 3.6. Penggabungan kembali dua buah simpul.....37

GAMBAR : 3.7. Penggabungan dua buah simpul sampai tersisa satu simpul.....38

GAMBAR : 3.8. Pemberian bit 0 dan bit 1 38

GAMBAR : 3.9. Proses Dekompresi denga pohon *Huffman*.....42

GAMBAR : 3.10. *Use Case Diagram* 45

GAMBAR : 3.11. *Activity Diagram* Kompresi 49

GAMBAR : 3.12. *Activity Diagram* Dekompresi 50

GAMBAR : 3.13. *Sequence Diagram* Kompresi 51

GAMBAR : 3.14. *Sequence Diagram* Dekompresi 51

GAMBAR : 3.15. *Flowchart* Proses Kompresi.....52

GAMBAR : 3.16. *Flowchart* Proses Dekompresi.....53

GAMBAR : 3.17. Rancangan Halaman Utama.....54

GAMBAR : 3.18. Halaman Kompresi 55

GAMBAR : 3.29. Halaman Dekompresi 56

GAMBAR : 3.20. Halaman *About* 57

GAMBAR : 4.1. Tampilan Utama 64

GAMBAR : 4.2. Tampilan Menu Kompresi 65

GAMBAR : 4.3. Tampilan Menu Dekompresi 66

GAMBAR : 4.4. Tampilan Menu *About* 67

GAMBAR : 4.5. Testing proses Kompresi 71

GAMBAR : 4.6. Testing proses Dekompresi.....72

DAFTAR TABEL

TABEL : 2.1. Tabel Simbol Penghubung/Alur *Flowchart* (Indrajani, 2011 : 22)... 18

TABEL : 2.2. Tabel Proses *Flowchart* (Indrajani, 2011 : 22) 19

TABEL : 2.3. Tabel *Input/Output Flowchart* (Indrajani, 2011 : 22) 20

TABEL : 2.4. Tabel *Use case Diagram* (Windu, 2013 : 4) 21

TABEL : 2.5. Tabel *Activity Diagram* (Windu, 2013 : 4) 23

TABEL : 2.6. Tabel *Mutlipicity Class Diagram* (Windu, 2013 : 4) 25

TABEL : 2.7. Tabel *Sequence Diagram* (Windu, 2013 : 4)..... 25

TABEL : 2.8. Tabel *Deployment Diagram* (Windu, 2013 : 4) 27

TABEL : 3.1. Referensi Penelitian 30

TABEL : 3.2. Tabel Frekuensi *Sample audio* 35

TABEL : 3.3. Tabel Hasil kompresi *Huffman* 39

TABEL : 3.4. Tabel Penjadwalan Penelitian 44

TABEL : 3.5. Spesifikasi *Use Case* Kompresi 46

TABEL : 3.6. Spesifikasi *Use Case* Dekompresi 46

TABEL : 3.7. Spesifikasi *Use Case* Buka *File* 47

TABEL : 3.8. Spesifikasi *Use case* Kompres 47

TABEL : 3.9. Spesifikasi *Use case* Buka *File* Kompresi/Dekompresi..... 48

TABEL : 3.10. Spesifikasi *Use case* Dekompres 48

TABEL: 4.1. Tabel pengujian *black box* pada halaman beranda..... 68

TABEL : 4.2. Tabel pengujian *black box* pada *form* kompresi..... 69

TABEL : 4.3. Tabel pengujian *black box* pada *form* dekompresi..... 70

TABEL : 4.4. Hasil pengujian proses Kompresi..... 73

TABEL : 4.5. Hasil pengujian proses Dekompresi..... 75

BAB I

PENDAHULUAN

1.1. Latar Belakang

Pesatnya perkembangan teknologi informasi dan komunikasi pada saat ini sangat memungkinkan manusia untuk melakukan pengiriman data dan informasi dengan cepat. Informasi tidak hanya disajikan dalam bentuk teks saja, melainkan data dan informasi dapat juga berupa gambar, suara dan *video* yang lebih dikenal dengan multimedia. *File audio* WAV merupakan salah satu *format file* suara yang banyak dipakai dalam sistem operasi *Windows* untuk keperluan *game* dan multimedia. *File audio* WAV sebenarnya merupakan *format* kasar (*raw format*) dimana *signal* suara langsung direkam dan dikuantisasi menjadi data *digital*. *Format* dasar dari *file* ini secara *default* tidak mendukung kompresi dan dikenal dengan nama PCM (*Pulse Code Modulation*).

Semakin lama durasi sebuah *file audio* WAV, semakin besar kapasitas media penyimpanan yang dibutuhkan untuk menyimpan data *audio file* tersebut. Media penyimpanan yang semakin besar tidak akan menjawab kebutuhan teknologi informasi jika data berkas (*file*) yang digunakan juga semakin besar. Masalah tersebut dapat diatasi bila *file audio* WAV tersebut dikompresi untuk mengurangi ukurannya. Salah satunya adalah dengan menggunakan metode kompresi *Huffman*.

Metode *Huffman* adalah salah satu metode kompresi yang paling terkenal untuk mengkompres data. Terdapat tiga *fase* untuk menggunakan metode *Huffman* untuk mengkompres sebuah data, pertama adalah *fase* pembentukan pohon *Huffman*, kedua adalah *fase encoding* (kompresi) dan ketiga *fase decoding* (dekompresi) . Prinsip kerja metode ini adalah dengan memperkecil bit yang sering muncul dan memperbesar bit yang jarang muncul. Sebagai contoh, data yang awalnya memiliki 56 bit, bisa di kompresi menjadi 11 bit tanpa ada informasi yang hilang.

Berdasarkan masalah di atas, penulis tertarik membuat tugas akhir dengan judul "*PERANCANGAN DAN ANALISIS KOMPRESI AUDIO WAV DENGAN MENGGUNAKAN METODE HUFFMAN*".

1.2. Identifikasi Masalah

Dapat disimpulkan bahwa berdasarkan latar belakang yang telah di uraikan diatas, identifikasi masalah yang terdapat pada tugas akhir ini adalah :

1. Bagaimana cara merancang aplikasi kompresi *file audio* WAV dengan menggunakan metode *Huffman*?
2. Bagaimana hasil *file audio* WAV setelah di kompresi?

1.3. Tujuan Penulisan

Berdasarkan rumusan masalah diatas, tujuan penulisan ini adalah :

1. Merancang aplikasi yang dapat mengkompresi *file audio* WAV dengan menggunakan metode *Huffman*.
2. Untuk mengetahui hasil *file audio* WAV setelah di kompresi.

1.4. Batasan Masalah

Batasan masalah ini dimaksudkan untuk lebih memfokuskan pada masalah-masalah apa saja yang akan dibahas pada tugas akhir ini. Adapun yang menjadi batasan masalah dalam tugas akhir ini adalah :

1. Metode yang digunakan adalah metode *Huffman*.
2. Bahasa pemograman yang digunakan adalah *Microsoft Visual Basic* 2010.
3. *File* yang dikompresi adalah *file audio* berekstensi WAV berukuran maksimum 10 MB.
4. *File* dapat didengarkan kembali setelah proses dekompresi.
5. Penelitian ini dilakukan hanya sampai proses *construction* (tidak melakukan proses *deployment*).

1.5. Metode Penelitian

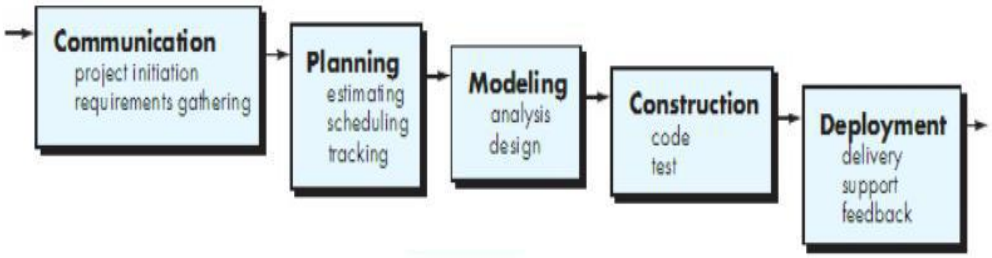
1.5.1. Metode Pengumpulan Data

Metode pengumpulan data yang digunakan dalam penelitian ini adalah Studi Pustaka. Dalam hal ini penulis mempelajari buku, artikel, jurnal maupun hasil penelitian terdahulu sebagai referensi yang diperlukan dalam melakukan penelitian ini. Tujuannya adalah untuk mendapatkan informasi mengenai metode *Huffman* serta pemograman *Visual Basic.*

1.5.2. Metode Pengembangan Perangkat Lunak

Dalam penelitian ini peneliti menggunakan model *waterfall* sebagai metode pengembangan perangkat lunak. Model *waterfall* adalah model klasik yang bersifat

sistematis, berurutan dalam membangun *software* waktu tertentu. Nama model ini sebenarnya adalah "*Linear Sequential Model*". Model ini sering disebut juga dengan "*classic life cycle*" atau metode *waterfall*. Model ini termasuk ke dalam model *generic* pada rekayasa perangkat lunak dan pertama kali diperkenalkan oleh Winston Royce sekitar tahun 1970 sehingga sering dianggap kuno, tetapi merupakan model yang paling banyak dipakai dalam *Software Engineering* (SE). Model ini melakukan pendekatan secara sistematis dan berurutan. Disebut dengan *waterfall* karena tahap demi tahap yang dilalui harus menunggu selesainya tahap sebelumnya dan berjalan berurutan (Pressman, 2015:42). Metode *waterfall* mempunyai tahapan-tahapan yang dijelaskan pada gambar 1.1.



GAMBAR : 1.1. Model *Waterfall* (Pressman, 2015:42)

1. *Communication* (*Project Initiation & Requirements Gathering*)
2. *Planning* (*Estimating, Scheduling, Tracking*)
3. *Modeling* (*Analysis & Design*)
4. *Construction* (*Code & Test*)
5. *Deployment* (*Delivery, Support, Feedback*)

Dalam penelitian ini hanya akan dilakukan hingga tahap 4 sehingga tahap penelitian ini tidak dilakukan dan menjadi batasan masalah.

1.6. Sistematika Penulisan

BAB I PENDAHULUAN

Bab ini berisikan Latar Belakang, Identifikasi Masalah, Tujuan Penelitian, Manfaaat Penelitian, Batasan Masalah, Metode Penelitian dan Sistematika Penulisan.

BAB II LANDASAN TEORI

Bab ini berisikan referensi dengan topik penelitian yang ditujukan untuk menunjang penulisan skripsi. Referensi yang dicantumkan berasal dari jurnal, *paper* ilmiah, *prosiding*, buku teks, *white paper*, majalah ilmiah dan tesis yang relevan dengan topik penelitian.

BAB III ANALISA MASALAH DAN PERANCANGAN PROGRAM

Bab ini berisikan tentang analisis kebutuhan informasi yang diperlukan untuk membangun aplikasi, gambaran sistem yang sedang berjalan dan sistem yang akan dibangun. Bab ini juga berisi perancangan antar muka aplikasi.

BAB IV IMPLEMENTASI DAN UJI COBA

Bab ini berisikan tentang penerapan sistem dari tahapan analisis dan perancangan sistem kedalam bentuk bahasa pemograman, serta perangkat keras dan perangkat

lunak yang dibutuhkan dalam membangun sistem. Bab ini juga berisi pengujian terhadap sistem apakah sudah benar- benar berjalan seperti yang diharapkan.

BAB V PENUTUP

Bab ini berisikan kesimpulan dari keseluruhan uraian bab – bab sebelumnya dan saran – saran dari hasil yang diperoleh yang diharapkan dapat bermanfaat untuk pengembangan selanjutnya.

BAB II

LANDASAN TEORI

2.1. Kompresi Data

Kompresi Data adalah proses pemadatan data yang bertujuan untuk memperkecil ukuran data sehingga selain dapat menghemat media penyimpanan data dalam jaringan (Howe,1993). Kompresi data juga didefinisikan sebagai suatu proses mengubah suatu input data menjadi data lain dengan *format* dan dengan ukuran yang lebih kecil (Solomon, 2004).

File merupakan data *digital* yang berupa representasi atas bit'0' dan '1' sering kali dalam sebuah *file* terjadi perulangan data atau *redundancy* . semua metode kompresi data melakukan pemadatan terhadap data berulang tersebut.

Berdasarkan hasil kompresi, algoritma kompresi terbagi atas *lossless compression* dan *lossy compression* (Sayood, 2005).

1. *Lossless compression* ,pada teknik ini pada umumnya digunakan penerapan yang tidak bisa mentoleransi setiap perbedaan antara data yang asli dan data yang telah dikontruksikan. Pada metode ini tidak ada terjadi kehilangan data. Data yang telah dikompresi akan dapat kembali persis seperti data yang asli, dengan kata lain data sesudah dikompres sama dengan sebelum dikompres. Data berbentuk teks harus dikompres dengan metode ini, karena kehilangan salah satu huruf dapat

mengakibatkan kesalah pahaman yang sangat berarti. Contoh metode ini adalah *Shannon-Fano Coding*, *Huffman Coding*, *Arithmetic Coding*, *Run length Encoding* dan lain sebagainya (Arizka, R. U.2011).

2. *Lossy compression*, pada teknik ini terjadi kehilangan pada beberapa data. Data yang telah dikompres tidak akan sama atau persis dengan data aslinya. Di dalam penerapan, banyak rekontruksi yang tepat untuk mengatasi suatu masalah. Kelebihan metode ini adalah rasio kompresinya yang lebih tinggi dibandingkan metode *lossless*. (Sayood, 2005).

   Pada gambar dan MP3, contoh metode ini adalah *Transform Coding*, *Wavelet*, dan lain-lain. Pada metode *lossy compression* tidak memungkinkan mengembalikan data aslinya seperti semula, atau disebut dengan *irreversible*. Keuntungan pada metode ini adalah kompresi yang dihasilkan lebih kecil dibandingkan metode *lossless*. Pada umumnya metode *lossy* digunakan pada *file* berbentuk suara, gambar, dan *video* karna dalam mengilangkan bagian-bagian tertentu pada *file* ini masih bisa ditoleransi oleh pengindraan manusia sehingga perbedaan yang didapat tidak terlalu menonjol, bahkan sekilas hampir mirip dengan aslinya.

2.2. Dekompresi Data

Kebalikan dari proses kompresi data yaitu proses dekompresi. Dekompresi adalah sebuah proses untuk mengembalikan data baru yang telah dihasilkan oleh proses kompresi menjadi data awal. Dekompresi yang menghasilkan data sama persis dengan data aslinya sebelum kompresi, maka data tersebut disebut *lossless compression*. Sebaliknya, jika hasil dekompresi menghasilkan data tidak sama persis dengan data aslinya sebelum dekompresi, karena ada data yang dihilangkan karena dirasa tidak terlalu penting tetapi tidak mengubah informasi yang dikandungnya, disebut *lossy compression*. Setelah dilakukan proses kompresi terhadap sebuah *file*, maka sebuah *file* akan dapat dikembalikan ke dalam bentuk semula yaitu dengan melakukan dekompresi (*decompression*) pada data *file* tersebut (Kharisma, 2010).

2.3. Algoritma

Dalam matematika dan komputasi algoritma merupakan kumpulan perintah yang saling berkaitan untuk menyelesaikan suatu masalah. Perintah-perintah ini dapat diterjemahkan secara bertahap dari awal hingga akhir. Dalam penyusunannya diperlukan urutan serta logika agar algoritma yang dihasilkan sesuai dengan yang diharapkan. Algoritma merupakan bagian terpenting yang tidak dapat dipisahkan dari pemograman. Meskipun sintaksis dan semantik yang dibuat benar adanya, dengan algoritma yang keliru, permasalahan yang ingin dipecahkan dengan teknik pemograman tidak akan berhasil. Oleh karena itu, sebelum membuat program aplikasi, hal pertama yang harus kita pahami algoritma atau prosedur pemecahannya.

Hal ini bertujuan agar *program* yang telah dibuat dapat sesuai dengan yang diharapkan (Ramadhani, 2015).

2.4. Algoritma *Huffman*

Algoritma *Huffman* ditemukan oleh *Huffman* pada tahun 1952 dan dipublikasikan pada *paper* yang berjudul *" A method for the construction of minimum-redudancy codes"* Algoritman *Huffman* dapat dipakai sebagai algoritma pencarian (*Searching*) dan algoritma pemapat data (*Data Compression*) (Widyawardhana, 2000).

Algoritma *Huffman* dapat diterapkan untuk melakukan pencarian karena pohon *Huffman* di bentuk dengan meletakkan *item* yang paling sering muncul di daun yang paling dekat dengan akar dan *item* yang paling jarang muncul di tempatkan pada daun yang paling jauh dengan akar. Pencarian dilakukan dengan membaca pohon tersebut satu persatu mulai dari akar, sehingga dalam proses pencarian *item* yang sering muncul dapat ditemukan. Dengan asumsi bahwa pencarian akan sering diperlukan untuk *item* yang sering muncul maka algoritma ini menjadi efektif.

Berdasarkan tipe peta kode yang digunakan untuk mengubah pesan awal (Isi data yang dimasukkan) menjadi sekumpulan *codeword*, algoritma *Huffman* termasuk kedalam kelas algoritma yang menggunakan metode statis. Metode statis adalah metode yang selalu menggunakan peta kode yang sama, metode membutuhkan dua *fase* (*two-pass*), *fase* pertama untuk menghitung probabilitas kemunculan tiap dan simbol dan menentukan peta kodenya, dan *fase* kedua untuk mengubah pesan menjadi kode yang akan di tranmisikan.

Sedangkan berdasarkan teknik pengodean simbol yang digunakan, algoritma *Huffman* menggunakan metode *symbolwise*. Metode *symbolwise* adalah metode yang menghitung peluang kemunculan dari setiap simbol dalam satu waktu, dimana simbol yang lebih sering muncul diberi kode lebih pendek dibandingkan simbol yang jarang mucul.

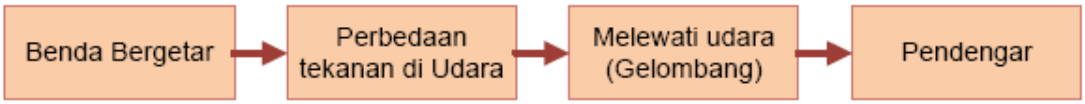
2.5. Rasio Kompresi

Dalam kompresi data, tujuan utama yang perlu di perhatikan adalah rasio kompresi yang semakin baik, dan proses kompresi dan pengembalian yang semakin cepat (Aburas, 2008). Rasio kompresi secara matematis dapat ditulis sebagai berikut :

$$\textbf{Rasio Kompresi} = \frac{\text{Ukuran File Asli - Ukuran File setelah dikompresi}}{\text{Ukuran File Asli}} \text{ x } 100$$

2.6. *Audio*

Audio (Suara) adalah fenomena fisik yang dihasilkan oleh getaran suatu benda yang berupa sinyal *analog* dengan *amplitude* yang berubah secara kontiniu terhadap satuan waktu yang disebut frekuensi. Selama bergetar, perbedaan tekanan terjadi di udara sekitarnya. Pola osilasi yang terjadi dinamakan sebagai gelombang. Gelombang mempunyai pola sama yang berulang pada *interval* tertentu, yang disebut sebagai periode. Contoh suara periodik adalah instrumen musik, nyanyian burung sedangkan contoh suara non periodik adalah batuk, percikan ombak dan lain-lain (Binanto, 2010).



GAMBAR : 2.1. Alur gelombang suara (Binanto, 2010)

2.7. Reprentasi Suara

Gelombang suara *analog* tidak dapat langsung direpresentasikan pada komputer. Komputer mengukur amplitudo pada satuan waktu tertentu untuk menghasilkan sejumlah angka. Tiap satuan pengukuran ini dinamakan "*sample*". *Analog To Digital Conversion* (ADC) adalah proses mengubah amplitudo gelombang bunyi ke dalam waktu *interval* tertentu (*sampling*), sehingga menghasilkan representasi *digital* dari suara. Dalam teknik *sampling* dikenal istilah *sampling rate* yaitu beberapa gelombang yang diambil dalam satu detik. Sebagai contoh jika kualitas CD *Audio* dikatakan memiliki frekuensi sebesar 44100 Hz, berarti jumlah sampel sebesar 44100 per detik (Pu IM, 2006).

Teknik *sampling* yang umum pada *file audio* seperti *Nyquist Sampling Rate* dimana untuk memperoleh representasi akurat dari suatu sinyal *analog* secara *lossless*, amplitudonya harus diambil *sample*-nya setidaknya pada kecepatan (*rate*) sama atau lebih besar dari 2 kali lipat komponen frekuensi maksimum yang akan didengar. Misalnya untuk sinyal analog dengan *bandwith* 15Hz – 10 kHz→*sampling rate* = 2 x 10kHz = 20 kHz.

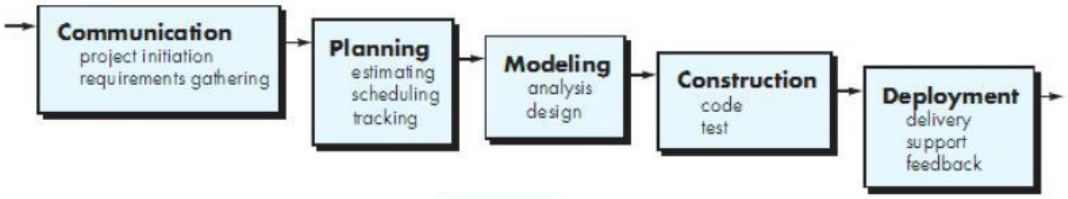
2.8. *File* WAV

WAV adalah *format audio* standar *Microsoft* dan IBM untuk personal *computer* (PC), biasanya menggunakan *coding* PCM (*Pulse Code Modulation*). WAV adalah data tidak terkompres sehingga seluruh sampel *audio* disimpan semuanya di *harddisk. Software* yang dapat menciptakan WAV dari *analog sound* misalnya adalah *Windows Sound Recorder*. *File audio* ini jarang sekali digunakan di internet karena ukurannya yang relatif besar dengan batasan maksimal untuk *file* WAV adalah 2 GB (Meckah, 2005).

Parameter-parameter tersebut menyatakan *setting* yang digunakan oleh ADC (*Analog-to-Digital Converter*) pada saat data *audio* direkam. Biasanya laju sampel juga dinyatakan dengan satuan Hz atau kHz. Sebagai gambaran, data *audio digital* yang tersimpan dalam CD *audio* memiliki karakteristik laju sampel 44100 Hz, 16 bit per sampel, dan 2 kanal (*stereo*), yang berarti setiap satu detik suara tersusun dari 44100 sampel, dan setiap sampel tersimpan dalam data sebesar 16-bit atau 2 byte. Laju sampel selalu dinyatakan untuk setiap satu kanal. Jadi misalkan suatu data *audio digital* memiliki 2 kanal dengan laju sampel 8000 sampel/detik, maka sesungguhnya di dalam setiap detiknya akan terdapat 16000 sampel. Sebagaimana telah dijelaskan sebelumnya bahwa untuk *stream* data *audio* menggunakan *header* berupa struktur *PCMWAVEFORMAT*. PCM merupakan singkatan dari *Pulse Coded Modulation*, yaitu suatu metode yang digunakan untuk mengkonversikan sinyal *audio* dari bentuk *analog* ke bentuk *digital*.

2.9. Metode Perancangan

Metode perancangan yang digunakan dalam tugas akhir ini adalah menggunakan salah satu dari metode pengembangan perangkat lunak *software development life cycle* (SDLC) yaitu metode *waterfall* yang memiliki beberapa tahapan yaitu seperti pada Gambar 2.2.



GAMBAR : 2.2. Model *Waterfall* (Pressman, 2015 : 42)

Berikut ini adalah penjelasan dari tahapan-tahapan yang dilakukan di dalam Model *Waterfall* menurut pressman pada Gambar 2.2 :

1. *Communication* (*Project Initiation & Requirements Gathering*)

Sebelum memulai pekerjaan yang bersifat teknis, sangat diperlukan adanya komunikasi dengan *customer* demi memahami dan mencapai tujuan yang ingin dicapai. Hasil dari komunikasi tersebut adalah inisialisasi proyek, seperti menganalisis permasalahan yang dihadapi dan mengumpulkan data-data yang diperlukan, serta membantu mendefinisikan fitur dan fungsi *software*. Pengumpulan data-data tambahan bisa juga diambil dari jurnal, artikel, dan *internet*.

2. *Planning* (*Estimating, Scheduling, Tracking*)

Tahap berikutnya adalah tahapan perencanaan yang menjelaskan tentang estimasi tugas-tugas teknis yang akan dilakukan, resiko-resiko yang dapat terjadi, sumber daya

yang diperlukan dalam membuat sistem, produk kerja yang ingin dihasilkan, penjadwalan kerja yang akan dilaksanakan, dan *tracking* proses pengerjaan sistem.

3. *Modeling* (*Analysis & Design*)

Tahapan ini adalah tahap perancangan dan permodelan arsitektur sistem yang berfokus pada perancangan struktur data, arsitektur *software*, tampilan *interface*, dan algoritma *program*. Tujuannya untuk lebih memahami gambaran besar dari apa yang akan dikerjakan.

4. *Construction* (*Code & Test*)

Tahapan *Construction* ini merupakan proses penerjemahan bentuk desain menjadi kode atau bentuk/bahasa yang dapat dibaca oleh mesin. Setelah pengkodean selesai, dilakukan pengujian terhadap sistem dan juga kode yang sudah dibuat. Tujuannya untuk menemukan kesalahan yang mungkin terjadi untuk nantinya diperbaiki.

5. *Deployment* (*Delivery, Support, Feedback*)

Tahapan *Deployment* merupakan tahapan implementasi *software ke customer*, pemeliharaan *software* secara berkala, perbaikan *software*, evaluasi *software*, dan pengembangan *software* berdasarkan umpan balik yang diberikan agar sistem dapat tetap berjalan dan berkembang sesuai dengan fungsinya. (Pressman, 2015:17).

2.10. *Flowchart*

Flowchart merupakan penggambaran secara grafik dari langkah-langkah dan urutan prosedur suatu *program*, Biasanya mempengaruhi penyelesaian masalah yang khusunya perlu dipelajari dan dievaluasi lebih lanjut.(Indrajani, 2011:22).

Flowchart di bedakan menjadi 5 jenis *flowchart*, antara lain *system flowchart*, *document flowchart*, *schematic flowchart*, *program flowchart*, *process flowchart*.

Masing-masing jenis *flowchart* akan dijelaskan berikut ini:

1. *System Flowchart*

System Flowchart dapat didefinisikan sebagai bagan yang menunjukkan arus pekerjaan secara keseluruhan dari sistem. Bagan ini menjelaskan urutan-urutan dari prosedur-prosedur yang ada di dalam sistem. Bagan alur sistem menunjukkan apa yang dikerjakan di sistem.

2. *Document Flowchart*

Bagan alur dokumen (*document flowchart*) atau disebut juga bagan alur formulir (*form flowchart*) atau *paperwork flowchart* merupakan bagan alur yang menunjukkan arus dari laporan dan formulir termasuk tembusan-tembusannya.

3. *Schematic Flowchart*

Bagan alur skematik (*schematic flowchart*) merupakan bagan alur yang mirip dengan bagan alur sistem, yaitu untuk menggambarkan prosedur di dalam sistem. Perbedaanya adalah, bagan alur skematik selain menggunakan simbol-simbol bagan alur sistem, juga menggunakan gambar-gambar komputer dan peralatan lainnya yang digunakan. Maksud penggunaa gambar-gambar ini adalah untuk memudahkan

komunikasi kepada orang yang kurang paham dengan simbol-simbol bagian alur. Penggunaan gambar-gambar ini memudahkan untuk dipahami, tetapi sulit dan lama menggambarnya.

4. *Program Flowchart*

Bagan alur *program* (*program flowchart*) merupakan bagan yang menjelaskan secara rinci langkah-langkah dari proses *program*. Bagan alur *program* dibuat dari derivikasi bagan alur sistem.

Bagan alur *program* dapat terdiri dari dua macam, yaitu bagan alur logika *program* (*program logic flowhart*) dan bagan alur *program* komputer terinci (*detailed computer program flowchart*). Bagan alur logika *program* digunakan untuk menggambarkan tiap-tiap langkah di dalam program komputer secara logika. Bagan alat logika *program* ini dipersiapkan oleh analurs sistem. Gambar berikut menunjukkan bagan alur logika *program*. Bagan alur *program* komputer terinci (*detailed computer program flowchart*) digunakan untuk menggambarkan instruksi-instruksi *program computer* secara terinci. Bagan alur ini dipersiapkan oleh pemogram.

5. *Process Flowchart*

Bagan alur proses (*process flowchart*) merupakan bagan alur yang banyak digunakan di teknik industri. Bagan alur ini juga berguna bagi analurs sistem untuk menggambarkan proses dalam suatu prosedur. Berikut ini merupakan notasi atau simbol-simbol yang digunakan dapat dibagi menjadi 3 (tiga) kelompok yaitu :

A. *Flow Direction Symbols* (Simbol Penghubung/alur)

Simbol yang digunakan untuk menghubungkan antara simbol yang satu dengan yang lainnya. Simbol ini juga disebut *connecting line*. Simbol-simbol penghubung/alur terdapat pada tabel dibawah ini.

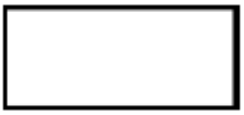
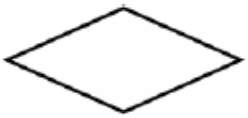
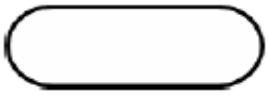
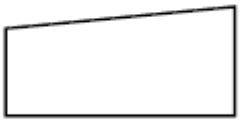
TABEL : 2.1. Tabel Simbol Penghubung/Alur *Flowchart* (Indrajani, 2011:22)

NO	*SYMBO*L	NAMA	KETERANGAN
1		Arus/*flow*	Untuk menyatakan jalannya suatu arus.
2		*Connector*	Untuk menyatakan sambungan dari satu proses ke proses lainnya dalam halaman / lembaran sama.
3		*Offline connector*	Untuk menyatakan sambungan dari satu proses ke proses lainnya dalam halaman atau lembaran yang berbeda.

B. *Processing Symbols* (Simbol Proses)

Simbol yang menunjukkan jenis operasi pengolahan dalam suatu proses / prosedur. Simbol – simbol proses pada *flowchart* terdapat pada tabel dibawah ini.

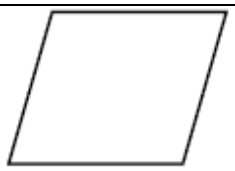
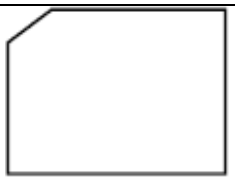
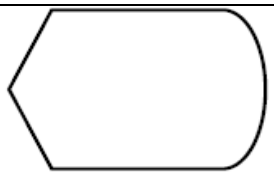
TABEL : 2.2. Tabel Proses *Flowchart* (Indrajani, 2011:22)

NO	*SYMBOL*	NAMA	KETERANGAN
1		Proses	Sebuah fungsi pemrosesanyang dilaksanakan oleh komputer biasanya menghasilkan perubahan terhadap data atau informasi.
2		*Symbol manual*	Untuk menyatakan suatu tindakan (proses) yang tidak dilakukan oleh komputer (manual).
3		*Decision / Logika*	Untuk menunjukkan suatu kondisi tertentu, dgn dua kemungkinan, YA / TIDAK.
4		Terminal	Untuk menyatakan permulaan atau akhir suatu *program*.
5		*Manual Input*	Untuk memasukkan data secara manual dengan menggunakan *online keyword*.
6		*Predefined Process*	Untuk menunjukkan suatu bagian *procedur*.

C. Input/Output Symbol

Simbol ini sering digunakan untuk menunjukan operasi pengolahan dalam suatu proses *input/output di flowchart*, akan dijelaskan pada tabel dibawah ini.

TABEL : 2.3. Tabel *Input/Output Flowchart* (Indrajani, 2011:22)

NO	*SYMBOL*	NAMA	KETERANGAN
1		*Input/Output*	Simbol yang menyatakan proses input dan *output* tanpa tergantung jenis peralatannya.
2		*Punched Card*	Simbol menyatakan *input* berasal dari kartu atau *output* ditulis ke kartu.
3		*Magnetic Tape*	Simbol menyatakan *input* berasal dari pita magnetis atau *output* disimpan ke pita magnetis.
4		*Disk Storage*	Simbol menyatakan *input* berasal dari *disk* atau *output* disimpan ke *disk*.
5		*Document*	Simbol mencetak keluaran dalam bentuk dokumen (melalui *printer*).
6		*Display*	Simbol mencetak keluaran dalam layar monitor.

2.11. UML (*Unified Modelling Language*)

Unified Modeling Language (UML) adalah bahasa spesifikasi standar yang dipergunakan untuk mendokumentasikan, menspesifikasikan dan membangun perangkat lunak. UML merupakan metodologi dalam mengembangkan sistem berorientasi objek dan juga merupakan alat untuk mendukung pengembangan sistem (Windu, 2013 : 4).

Alat bantu yang digunakan dalam perancangan berorientasi objek berbasiskan UML adalah sebagai berikut.

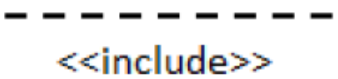
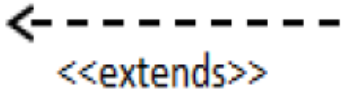
1. *Use Case Diagram*

   Use-case diagram adalah diagram yang mendeskripsikan interaksi antara *user* dengan sistem. Sebuah *use-case diagram* dapat menggambarkan semua kegiatan didalam satu sistem yang berjalan. Simbol-simbol yang digunakan dalam *Use Case Diagram* yaitu :

TABEL : 2.4. Tabel *Use case Diagram* (Windu, 2013 : 4)

No	Gambar	Keterangan
1		*Use case* menggambarkan fungsionalitas yang disediakan sistem sebagai unit-unit yang bertukar pesan antar unit dengan aktor, yang dinyatakan dengan menggunakan kata kerja.

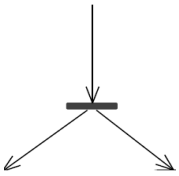
Lanjutan TABEL : 2.4. Tabel *Use case Diagram* (Windu, 2013 : 4)

No	Gambar	Keterangan
2		*Actor* atau Aktor adalah *Abstraction* dari orang atau sistem yang lain yang mengaktifkan fungsi dari target sistem. Untuk mengidentifikasikan aktor, harus ditentukan pembagian tenaga kerja dan tugas-tugas yang berkaitan dengan peran pada konteks target sistem. Orang atau sistem bisa muncul dalam beberapa peran. Perlu dicatat bahwa aktor berinteraksi dengan *Use Case*, tetapi tidak memiliki *control* terhadap *use case.*
3		Asosiasi antara aktor dan *use case*, digambarkan dengan garis tanpa panah yang mengindikasikan siapa atau apa yang meminta interaksi secara langsung dan bukannya mengindikasikan data.
4		Asosiasi antara aktor dan *use case* yang menggunakan panah terbuka untuk mengindikasikan bila *actor* berinteraksi secara pasif dengan sistem.
5		*Include*, merupakan di dalam *use case* lain (*required*) atau pemanggilan *use case* oleh *use case* lain, contohnya adalah pemanggilan sebuah fungsi *program*.
6		*Extend*, merupakan perluasan dari *use case* lain jika kondisi atau syarat terpenuhi.

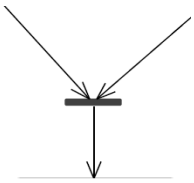
2. *Activity Diagram*

Activity diagram adalah suatu diagram yang digunakan untuk menggambarkan aliran proses usaha secara grafis, langkah-langkah dari suatu *use-case*, atau logika dari suatu tingkah laku objek (*method*). Simbol-simbol yang digunakan dalam *activity Diagram* yaitu :

TABEL : 2.5. Tabel *Activity Diagram* (Windu, 2013 : 4)

NO	GAMBAR	KETERANGAN
1		*Start Point*, diletakkan pada pojok kiri atas dan merupakan awal aktivitas.
2		*End Point*, akhir aktivitas.
3		*Activities*, menggambarkan suatu proses atau kegiatan bisnis.
4		*Fork*/percabangan, digunakan untuk menunjukkan kegiatan yang dilakukan secara paralel atau untuk menggabung kan dua kegiatan paralel menjadi satu.

Lanjutan TABEL : 2.5. Tabel *Activity Diagram* (Windu, 2013 : 4)

No	Gambar	Keterangan
5		*Join* (penggabungan) atau *rake*, digunakan untuk menunjukkan adanya dekomposisi.
6		*Decision Points*, menggambar kan pilihan untuk pengambilan keputusan, *true* atau *false*.
7		*Swimlane*, pembagian *activity diagram* untuk menunjukkan siapa melakukan apa.

3. *Class Diagram*

Merupakan hubungan antar kelas dan penjelasan *detail* tiap-tiap kelas di dalam model desain dari suatu sistem, juga memperlihatkan aturan-aturan dan tanggung jawab entitas yang menentukan perilaku sistem.

Class Diagram juga menunjukkan atribut-atribut dan operasi-operasi dari sebuah kelas dan *constraint* yang berhubungan dengan objek yang dikoneksikan.

Class Diagram meliputi : Kelas (*Class*), Relasi *Associations*, *Generalitation*, dan *Aggregation*, Attribut *(Attributes*), Operasi (*Operation/ Method*) dan *visibility*,

tingkat akses objek eksternal kepada suatu operasi atau atrribut. Hubungan antar kelas mempunyai keterangan yang disebut *Mutlipicity* atau *Cardinality*.

TABEL : 2.6. Tabel *Mutlipicity Class Diagram* (Windu, 2013 : 4)

Mutlipicity	Penjelasan
1	Satu dan hanya satu.
0..*	Boleh tidak ada 1 atau lebih.
1..*	1 atau lebih.
0..1	Boleh tidak ada, maksimal 1.
n..n	Batasan antara. Contoh 2..4 mempunyai arti minimal 2 maksimal 4.

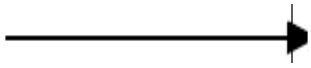
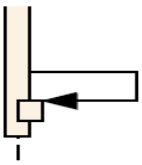
4. *Sequence Diagram*

Sequence diagram adalah diagram yang memodelkan logika sebuah *use-case* dengan cara menggambarkan interaksi pesan di antara objek – objek dalam rangkaian waktu. Simbol-simbol yang digunakan dalam *Sequence Diagram* yaitu:

TABEL : 2.7. Tabel *Sequence Diagram* (Windu, 2013 : 4)

NO	GAMBAR	KETERANGAN
1		*Entity Class*, merupakan bagian dari sistem yang berisi kumpulan kelas berupa entitas-entitas yang membentuk gambaran awal sistem dan menjadi landasan untuk menyusun basis data.

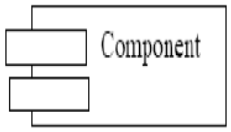
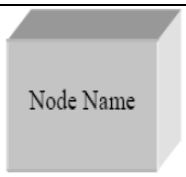
Lanjutan TABEL : 2.7. Tabel *Sequence Diagram* (Windu, 2013 : 4)

No	Gambar	Keterangan
2		*Boundary Class*, berisi kumpulan kelas yang menjadi *interfaces* atau interaksi antara satu atau lebih aktor dengan sistem, seperti tampilan *form entry* dan *form* cetak.
3		*Control class*, suatu objek yang berisi logika aplikasi yang tidak memiliki tanggung jawab kepada entitas, contohnya adalah kalkulasi dan aturan bisnis yang melibatkan berbagai objek.
4		*Message*, simbol mengirim pesan antar *class*.
5		*Recursive*, menggambarkan pengiriman pesan yang dikirim untuk dirinya sendiri.
6		*Activation*,mewa kili sebuah eksekusi operasi dari objek, panjang kotak ini berbanding lurus dengan durasi aktivasi sebuah operasi.
7		*Lifeline*, garis titik-titik yang terhubung dengan objek, sepanjang *lifeline* terdapat *activation*.

5. *Deployment Diagram*

Deployment Diagram digunakan untuk menggambarkan detail bagaimana komponen disusun di infrastruktur sistem.

TABEL : 2.8. Tabel *Deployment Diagram* (Windu, 2013 : 4)

No	Gambar	Keterangan
1		Pada *deployment diagram* komponen – komponen yang ada di letakkan kedalam *node* untuk memastikan keberadaaan posisi mereka.
2		*Node* menggambarkan bagian-bagian *hardware* dalam sebuah sistem. Notasi untuk *node* digambarkan sebagai sebuah kubus 3 dimensi.
3		Sebuah *association* digambarkan sebagai sebuah garis yang menghubungkan dua *node* yang mengindikasikan jalur komunikasi antara elemen-elemen *hardware.*

2.12 Testing

2.12.1 Pengertian *Testing*

Pengujian *software* adalah proses verifikasi dan validasi apakah sebuah aplikasi *software* atau program memenuhi persyaratan bisnis dan persyaratan teknis yang mengarahkan desain dan pengembangan dan cara kerjanya seperti yang

diharapkan dan juga mengidentifikasi kesalahan yang penting yang digolongkan berdasarkan tingkat *severity* pada aplikasi yang harus diperbaiki (Quadri dan Farooq 2010 : 1).

Pengujian *software* adalah teknik yang sering digunakan untuk verifikasi dan validasi kualitas suatu *software*. Pengujian *software* adalah prosedur untuk eksekusi sebuah *program* atau sistem dengan tujuan untuk menemukan kesalahan (Nidhra dan Dondeti 2012:1).

2.12.2. *Behavioral Test* (*Black-Box Test*)

Tester menggunakan *behavioral test* (disebut juga *Black-Box Tests*), sering digunakan untuk menemukan *bug* dalam *high level operations*, pada tingkatan fitur, profil operasional dan skenario *customer* (Black 2009 :3). *Teste*r dapat membuat pengujian fungsional *black box* berdasarkan pada apa yang harus sistem lakukan. *Behavioral testing* melibatkan pemahaman rinci mengenai *domain* aplikasi, masalah bisnis yang dipecahkan oleh sistem dan misi yang dilakukan sistem. *Behavioral test* paling baik dilakukan oleh penguji yang memahami desain sistem, setidaknya pada tingkat yang tinggi sehingga mereka dapat secara efektif menemukan *bug* umum untuk jenis desain.

Black box testing juga disebut *functional testing*, sebuah teknik pengujian fungsional yang merancang *test case* berdasarkan informasi dari spesifikasi (Nidhra dan Dondeti 2012:1).

BAB III

ANALISA MASALAH DAN PERANCANGAN PROGRAM

3.1. *Communication*

Pada tahap ini dilakukan proses pengumpulan informasi berupa data yang berkaitan dengan penelitian.

3.1.1. Pengumpulan Data

Pengumpulan data merupakan suatu hal yang penting dalam penelitian, karena ini merupakan stategi ataupun cara yang dipakai oleh peneliti untuk mengumpulkan data yang berkaitan dengan kompresi *audio* berekstensi WAV guna mendukung penelitian yang sedang dilakukan. Teknik pengumpulan data yang dilakukan dalam penelitian adalah :

3.1.1.1. Studi Literatur

Peneliti melakukan studi literatur dengan cara membaca dan mempelajari secara mendalam buku-buku referensi dan jurnal - jurnal penelitian sejenis mengenai kompresi *audio* WAV dan metode *Huffman*. Berikut inilah tabel literatur yang menjadi referensi dalam penelitian :

TABEL : 3.1. Referensi Penelitian

No	*Literature*	Pembahasan
1	Hengki Tamando Sihotang "Perancangan dan implementasi algoritma *arithmetic coding* untuk aplikasi kompresi data *video* dan *audio*" Jurnal matik penusa Vol.2, No 1, 2018.	Penelitian ini dibuat untuk mengimplementasi algoritma *arithmetic coding* untuk mengkompresi *video* dan *audio*.
2	Eka Kristian Gulo "Perancangan aplikasi kompresi *audio* dengan menerapkan algoritma *golomb*" Jurnal Pelita Informatika, Vol.16, No.4, Oketober 2017.	Penelitian ini dibuat untuk merancang aplikasi kompresi *audio* dengan menggunakan algoritma *Golomb*.
3	Darno Willfrid Midukta Simamora, Garuda Ginting, Yasir Hasan "Implementasi algoritma *run length encoding* pada kompresi *file* mp3" Jurnal Riset Komputer (JARKOM) Vol.3, No.4, Agustus 2016.	Penelitian ini buat untuk Mengkompresi file *audio* mp3 menggunakan Algoritma *Run Legnth Encoding*,
4.	Victor Amrizal "Implementasi Algoritma Kompresi Data *Huffman* Untuk Memperkecil Ukuran *File* MP3 *Player*" Jurnal sistem Informatika, Vol.3, No.2, 2010	Penelitian ini dibuat untuk mengkompresi *file mp3* dengan metode *Huffman*. Disini terdapat bagaimana caranya memproses kompresi data menggunakan metode *Huffman*.

Lanjutan TABEL : 3.1. Referensi Penelitian

No	*Literature*	Pembahasan
5.	Kharisma Mahesa, Karpen "Rancang bangun aplikasi kompresi dan dekompresi pada citra *digital* menggunakan metode *Huffman*" Jurnal *Processor*, Vol.12, No.1, April 2017	Penelitian ini dibuat untuk mengkompresi dan dekompresi citra *digital* menggunakan metode *Huffman*. Disini terdapat pembahasan mengenai proses dekompresi.

3.1.2. Analisis Sistem

Kompresi *file* yang akan dibangun mengimplementasikan metode *Huffman*, pengkompresian *file* yang dapat memampatkan ukuran *file* sesuai dengan jenis *file* yang telah ditentukan. Proses yang dapat dilakukan pada sistem yang akan dirancang mencakup kompresi *file* dan dekompresi *file*, Desain dan implementasi rancangan meliputi desain *interface*, gambaran proses sistem, implementasi desain, dan semua yang diperlukan dalam aplikasi kompresi *file* yang akan dibangun. Aplikasi kompresi *file* ini mampu memanpatkan *file* berekstensi WAV.

3.1.3 Analisis Identifikasi Masalah

Analisis masalah merupakan langkah dimana langkah ini diperlukan untuk mengertahui permasalahan yang akan terjadi di dalam sistem yang dirancang oleh penulis. Indentifikasi masalah dalam penelitian ini adalah bagaimana cara merancang

aplikasi kompresi *file audio* WAV menggunakan metode *Huffman* serta untuk mengetahui berapa hasil dari *file audio* WAV setelah dilakukan proses kompresi.

3.1.4. Analisis Kebutuhan

Analisis kebutuhan yang dibutuhkan untuk memenuhi kebutuhan apa saja yang diperlukan untuk membangun sistem ini. Pada tahap ini terdiri dari 2 jenis kebutuhan yaitu kebutuhan fungsional dan kebutuhan non fungsional.

1. Kebutuhan Fungsional

   Kebutuhan fungsional merupakan kebutuhan yang berisi proses dalam sebuah sistem. Adapun kebutuhan fungsional tersebut adalah :

   a. *Software* dapat mengkompresi *file audio* WAV dengan metode *Huffman*.
   b. *Software* dapat mengdekompresi *file audio* WAV dengan metode *Huffman*.
   c. *Software* dapat menampilkan rasio dan waktu proses kompresi.

2. Kebutuhan Non Fungsional

   Kebutuhan non fungsional merupakan suatu kebutuhan yang menitik beratkan pada alat pendukung prilaku sistem. Kebutuhan non fungsional nantinya akan meliputi beberapa hal penting yang dibutuhkan oleh sistem.

   a. Mudah dimengerti dan mudah dipahami oleh pengguna (*User Friendly*).
   b. Tidak menggunakan koneksi *internet*.
   c. Aplikasi dibangun menggunakan *Microsoft Visual Basic 2010*.

3.1.4.1. Analisis Perangkat Keras

Analisis Kebutuhan perangkat keras bertujuan untuk mengetahui spesifikasi perangkat keras apa saja yang sedang digunakan oleh aplikasi ini diantaranya adalah sebagai berikut.

1. *Processor* AMD A9-9400 (2Cpu) 2.4 GHz.
2. *Memory* (RAM) 4GB.
3. *Harddisk* Minimal 6GB.
4. VGA *card* minimal 128 MB
5. *Mouse*
6. *Keyboard*

3.1.4.2. Analisis Perangkat Lunak

Perangkat lunak yang digunakan dalam aplikasi ini diantaranya adalah sebagai berikut :

1. Sistem Operasi *Windows* 10
2. *Microsoft Visual Basic 2010.*

3.1.5. Analisis Proses Kompresi Algoritma *Huffman*

Kode *Huffman* merupakan angka-angka biner yang digunakan untuk mengkodekan setiap karakter dalam *file*. Pohon *Huffman* (*Huffman Tree*) adalah karakter-karakter yang akan direpresentasikan dalam biner, dipisahkan ke dalam cabang pohon biner dan diberi frekuensinya. Cabang sebelah kiri diberi bit 0 dan bit kanan diberi angka 1 sebagai identitas (Fardisa dan Budiono, 2011). Jadi, bit ini akan

dibaca dari akar sampai ke daun sehingga terbentuk angka biner sebagai identitas setiap karakter. Adapun langkah-langkah untuk membentuk pohon *Huffman* adalah sebagai berikut :

1. Baca semua data untuk menentukan frekuensi kemunculan masing-masing simbol pada sampel *file audio*.
2. Urutkan simbol sampel *audio* berdasarkan frekuensi kemunculannya secara *Ascending* (terkecil ke yang terbesar).
3. Gabung dua simpul bebas yang mempunyai frekuensi kemunculan paling kecil pada kumpulan simpul. Kemudian dibuat simpul pertama pada pohon *Huffman* dimana simpul tersebut mempunyai frekuensi yang merupakan hasil penjumlahan dari dua simpul penyusunnya.
4. Masukkan simpul pertama ke dalam kumpulan simpul dan urutkan kembali berdasarkan frekuensi kemunculannya, dari yang terkecil ke yang terbesar.
5. Ulangi langkah 2-4 sampai tersisa hanya satu pohon biner. Agar pemilihan dua pohon yang akan digabungkan berlangsung cepat, maka semua pohon yang ada selalu terurut menaik berdasarkan frekuensi.
6. Beri label setiap sisi pada pohon biner dengan cara sisi kiri pohon diberi label 0 dan sisi kanan pohon diberi label 1.
7. Telusuri pohon biner dari akar ke daun. Barisan label-label sisi dari akar kedaun menyatakan kode *Huffman* untuk simbol yang bersesuaian.

Berikut ini adalah contoh untuk memperjelas langkah-langkah membentuk pohon *Huffman*. Misalnya ada sampel *audio* seperti berikut.

A1 A1 A1 A1 A1 A1 A1 A1 A1 A1 B2 B2 B2 B2 B2 B2 B2 B2 C3 C3 C3 C3 C3 C3 D4 D4 D4 D4 E5 E5

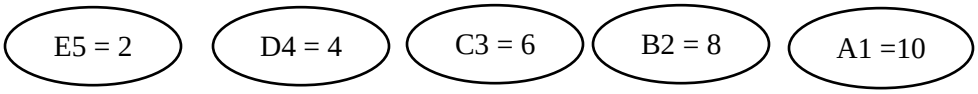
Dari data diatas ikuti langkah-langkah di atas sampai terbentuk pohon *Huffman*.

1. Menentukan frekuensi atau banyaknya sampel *audio* yang muncul, misalnya frekuensi dari setiap sampel *audio* seperti pada tabel 3.2 berikut.

TABEL : 3.2. Tabel Frekuensi *Sample audio*

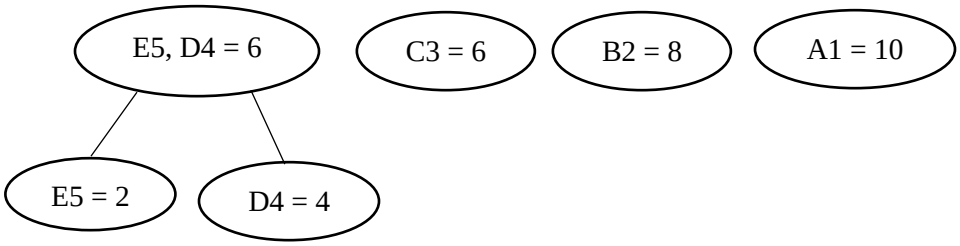
Sample Audio	**Frekuensi**
A1	10
B2	8
C3	6
D4	4
E5	2

2. Setelah frekuensi dari sampel *audio* diketahui maka urutkan data secara *Ascending* (terkecil ke terbesar), dengan masing-masing sampel sebagai simpul seperti pada gambar 3.1 berikut.



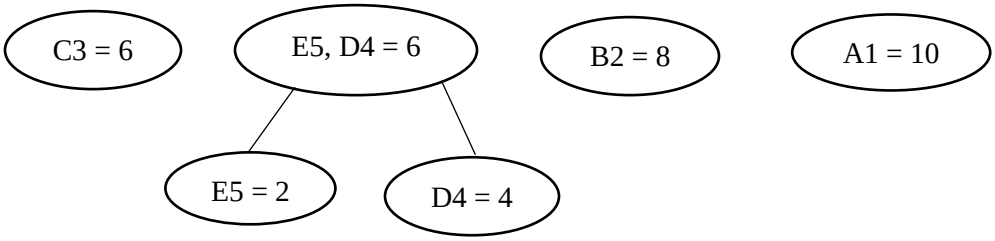
GAMBAR : 3.1. Urutan Sampel secara *Ascending*

3. Dari gambar 3.1 sudah menunjukan sampel secara berurutan sesuai dengan frekuensi terkecil ke terbesar. Maka langkah selanjutnya adalah penggabungan dua buah simpul yang mempunyai frekuensi terkecil, seperti pada gambar 3.2 berikut.



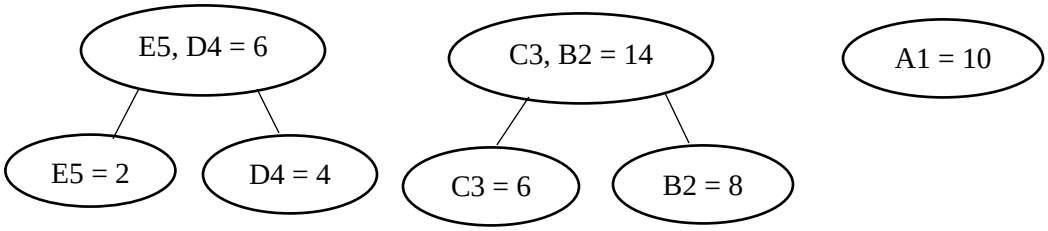
GAMBAR : 3.2. Penggabungan dua buah simpul (*node*)

4. Dari gambar 3.2 sudah menunjukan penggabungan dua buah simpul (*node*) dengan frekuensi terkecil. Lalu langkah selanjutnya adalah mengurutkan kembali seperti pada gambar 3.1 secara *Ascending* lalu penggabungan dua buah simpul yang mempunyai frekuensi terkecil sedangkan gambar 3.2 tidak perlu diperbaharui lagi, seperti pada gambar 3.3 berikut.



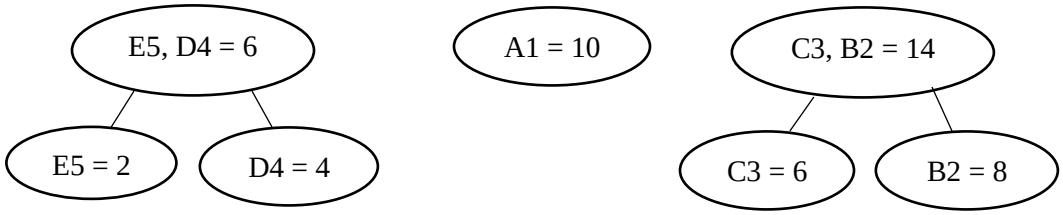
GAMBAR : 3.3. Pemindahan dua buah simpul urutan sampel secara *Ascending*

5. Ulangi langkah-langkah sebelumnya sampai tersisa satu pohon biner. Seperti pada gambar 3.4 berikut.



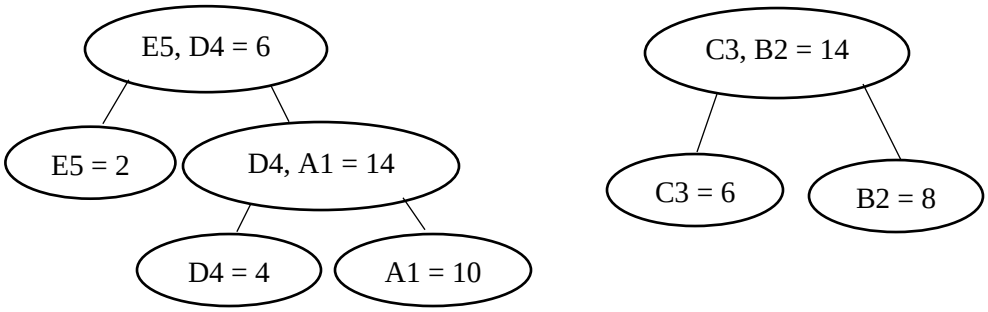
GAMBAR : 3.4. Penggabungan dua buah simpul

Karena ketiga simpul masih belum sesuai dengan frekuensi dari yang terkecil ke yang terbesar, maka dilakukan pemindahan sehingga kembali *Ascending*, seperti pada gambar 3.5 berikut :



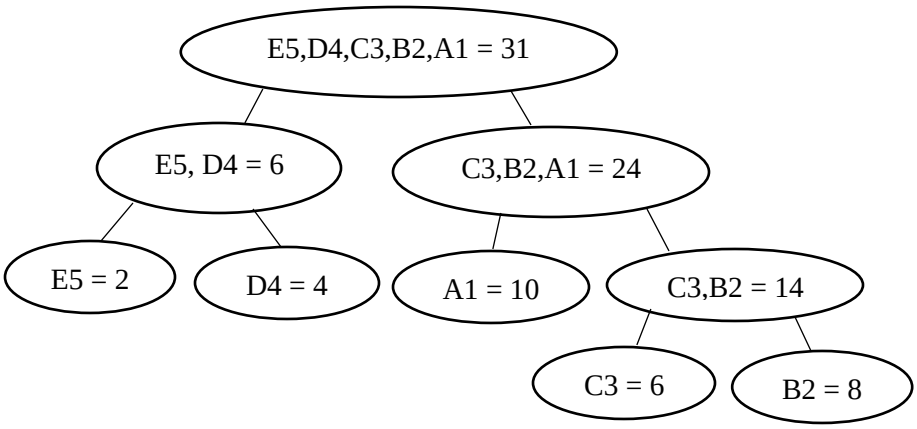
GAMBAR : 3.5. Pemindahan dua buah simpul secara *Ascending*

6. Langkah selanjutnya adalah menggabungkan dua buah simpul yang mempunyai frekuensi kemunculan terkecil, kemudian diurutkan kembali seperti pada gambar 3.6 berikut.



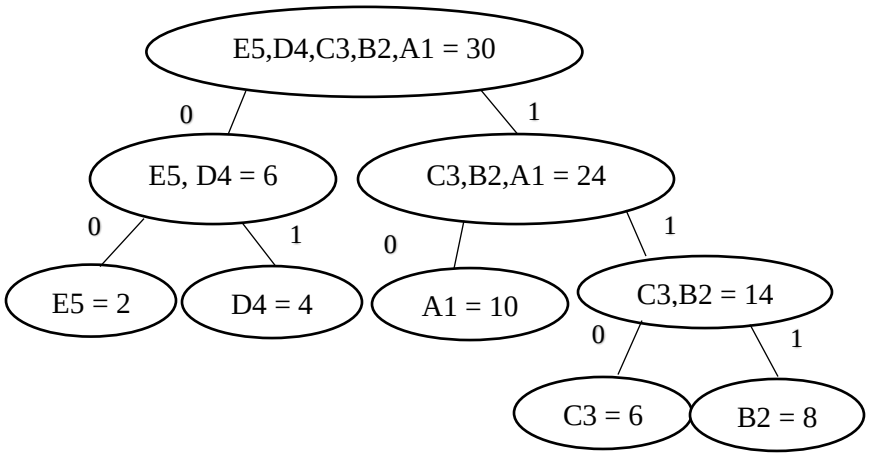
GAMBAR : 3.6. Penggabungan kembali dua buah simpul

7. Langkah selanjutnya adalah menggabungkan dua buah simpul yang mempunyai frekuensi kemunculan terkecil, sehingga hanya memiliki satu simpul seperti pada gambar 3.7 berikut.



GAMBAR : 3.7. Penggabungan dua buah simpul sampai tersisa satu simpul

8. Maka langkah akhir dalam pembentukan pohon *Huffman* adalah dengan memberi bit 0 di sebelah kiri dan bit 1 di sebelah kanan setiap akar ke daun, seperti yang ada pada gambar 3.8 berikut :



GAMBAR : 3.8. Pemberian bit 0 dan bit 1

Setelah melihat hasil dari pohon *Huffman*, maka dapat dibuat sebuah tabel yang berisi identitas dari tiap-tiap sampel *audio* berdasarkan biner. Dalam proses kompresi ini, prosesnya adalah baca urutan biner tiap sampel sampai *edge* dari pohon yang berisi sampel yang dicari. Misalnya, sampel yang dicari adalah A1, maka akan dibaca urutan sampel dari akar sampai ke A1 dengan kode biner 10. Maka gantikan sampel *audio* sebelum di kompresi menjadi kode biner seperti berikut :

A1 A1 A1 A1 A1 A1 A1 A1 A1 A1 B2 B2 B2 B2 B2 B2 B2 B2 C3 C3 C3 C3 C3 C3 D4 D4 D4 D4 E5 E5

Menjadi :

10 10 10 10 10 10 10 10 10 10 111 111 111 111 111 111 111 111 110 110 110 110 110 110 01 01 01 01 00 00

Demikian untuk seluruh sampel-sampelnya, seperti pada tabel 3.3 berikut.

TABEL : 3.3. Tabel Hasil kompresi *Huffman*

Sample *Audio*	**Frekuensi**	**Kode Binary**
A1	10	10
B2	8	111
C3	6	110
D4	4	01
E5	2	00
Total	**30 byte**	**12 bit**

$$T_f = f_1 + f_2 + f_3 + f_4 + f_5 + \ldots + f_n$$

Dimana :

T_f = Total Frekuensi (Byte)

f_n = Frekuensi setiap sampel

Cara menghitung total frekuensi adalah sebagai berikut :

$T_f = 10 + 8 + 6 + 4 + 2 = 30$ Byte

Sedangkan ukuran audio setelah dikompresi menggunakan algoritma *Huffman* adalah sebagai berikut :

$$H_t = h_1 f_1 + h_2 f_1 + h_3 f_1 + h_4 f_1 + h_5 f_1 + \ldots + h_n f_n$$

Dimana :

H_t = Total kode *Huffman* (Bit)

h_n = *Bit Length* (Bit)

f_n= Frekuensi setiap sampel

Cara menghitung total kode *huffman* adalah sebagai berikut :

H_t = (2 bit x 10) + (3 bit x 8) + (3 bit x 6) + (2 bit x 4) + (2 bit x 2)

= 74 bit

= 9.25 Byte

Sebelum dilakukan kompresi ukuran *file audio* adalah 30 byte, diperoleh dari total frekuensi kemunculan sampel dikalikan dengan 1 byte. Sedangkan ukuran *audio* setelah dikompresi (dalam kode *Huffman*): (2 bit x 10) + (3 bit x 8) + (3 bit x 6) + (2 bit x 4) + (2 bit x 2) adalah 74 bit setara dengan 9.25 byte.

Rasio Kompresi = $\frac{File\ Asli - Ukuran\ file\ setelah\ dikompresi}{File\ Asli} X100\%$

Rasio Kompresi = $\frac{30-9.25}{30} X100\%$

$= \frac{20.75}{30} X100\%$ = 0.692 X 100% = 69,2%

Jadi rasio kompresinya adalah 69,2%

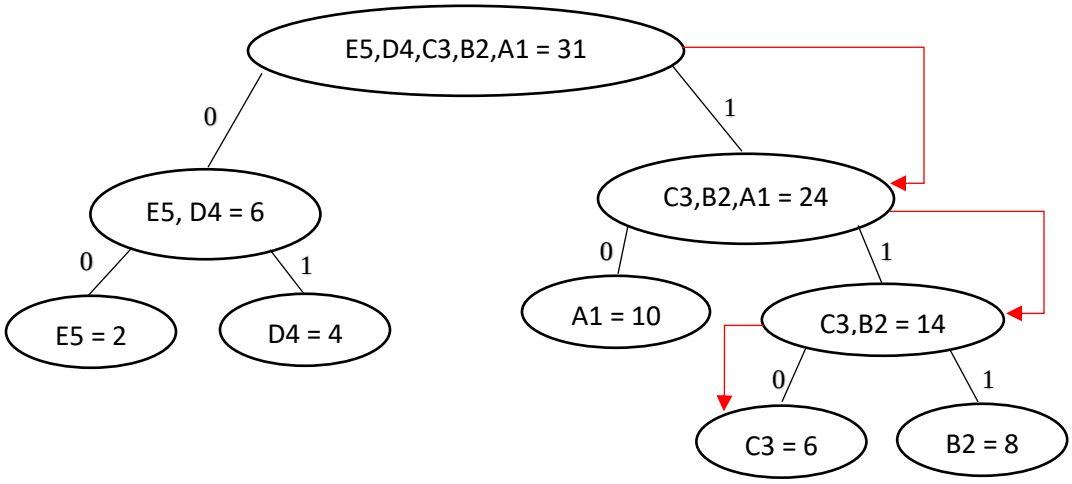
3.1.6. Analisis Proses Dekompresi Algoritma *Huffman*

Dekompresi adalah kebalikan dari kompresi yaitu penyusunan kembali sampel *audio* yang telah dikompresi menjadi sampel semula. Sebelum melakukan dekompresi *file*, maka harus dibentuk kembali pohon *Huffman* berdasarkan informasi yang disimpan dengan *file* yang dikompresi. Proses dekompresi dapat dilakukan dengan dua cara, yang pertama adalah dengan pohon *Huffman* dan yang kedua menggunakan tabel kode *Huffman*. Seperti pada contoh kompresi *Huffman* atau kompresi sebelumnya. Untuk melakukan dekompresi dapat dilakukan dengan pohon *Huffman* sebagai berikut :

1. Proses pembacaan simpul dimulai dari akar pohon *Huffman*, dengan membaca sebuah bit.

2. Untuk setiap bit pada langkah 1, lakukan *traversal* (Kunjungan dalam pohon, dengan setiap *node* hanya dikunjungi tepat satu kali) pada cabang yang bersesuaian.
3. Lakukan pengulangan langkah 1 dan 2 sampai cabang yang bersesuaian ketemu setiap bagian simpul mana sampel itu berada.
4. Lakukan pengulangan langkah 1 sampai seluruh sampel *audio* tersebut di dekompresi.

Sebagi contoh akan meng-dekompresi biner 110, maka gambarnya adalah sebagai berikut :



GAMBAR : 3.9. Proses Dekompresi dengan pohon *Huffman*

Pada gambar 3.9 dimulai dari akar seluruh sampel sampai sampel yang dinginkan ketemu dimana biner 110 adalah C3. Cara kedua adalah dengan menggunakan tabel kode *Huffman* seperti yang ada pada tabel 3.3 di atas.

10 10 10 10 10 10 10 10 10 10 111 111 111 111 111 111 111 111 110 110 110 110 110 110 01 01 01 01 00 00

Pertama-tama baca bit pertama dari biner tersebut adalah 1 karena bit yang dibaca adalah 1 maka akan dilakukan penelusuran terhadap pohon *Huffman* ke arah simpul yang terletak disebelah kanan terlebih dahulu. Kemudian dilakukan penelusuran kembali jika 1 tidak mewakili simbol data maka akan dibaca bit berikutnya sehingga bit yang dibaca manjadi 10 dan karena 10 mewakili simbol data yang bersesuaian dan berakhir pada sebuah daun maka dinyatakan bahwa 10 mewakili simbol A1. Setelah itu dibaca kembali bit berikutnya dan lakukan penelusuran kembali terhadap pohon *Huffman* sehingga didapat bit-bit yang mewakili sebuah simbol data yang bersesuaian. Sampai semua sampel kembali ke bentuk sampel sebelum dikompresi.

3.2. *Planning*

Dalam tahap ini penelitian memfokuskan pada penjadwalan pengerjaan penelitian. Pada penelitian ini terdapat beberapa proses yang harus dilakukan dari tahap *communication, implementation* sampai dengan *testing* maka dari itu diperlukan penjadwalan yang tepat agar penelitian ini dapat selesai sesuai dengan waktunya. Berikut penjadwalan penelitian berdasarkan aktifitas yang dilakukan dengan skala waktu yang digambarkan pada tabel 3.4.

TABEL : 3.4. Tabel Penjadwalan Penelitian

Tahap	No	Aktivitas	Juli				Agustus				September				Oktober			
			1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
Scedulling	1	Penjadwalan																
Comunication	1	Pengumpulan Data																
	2	Analisis Masalah dan Kebutuhan																
	3	Pendefinisian Fungsi																
Scedulling, Estimating, and Tracking	2	Analisis Kebutuhan perangkat keras dan perangkat lunak																
Analysis and Design	1	Perancangan UML																
	2	Perancangan *Interface*																
Construction	1	Pengkodean																
	2	*Testing*																

3.3. *Modelling*

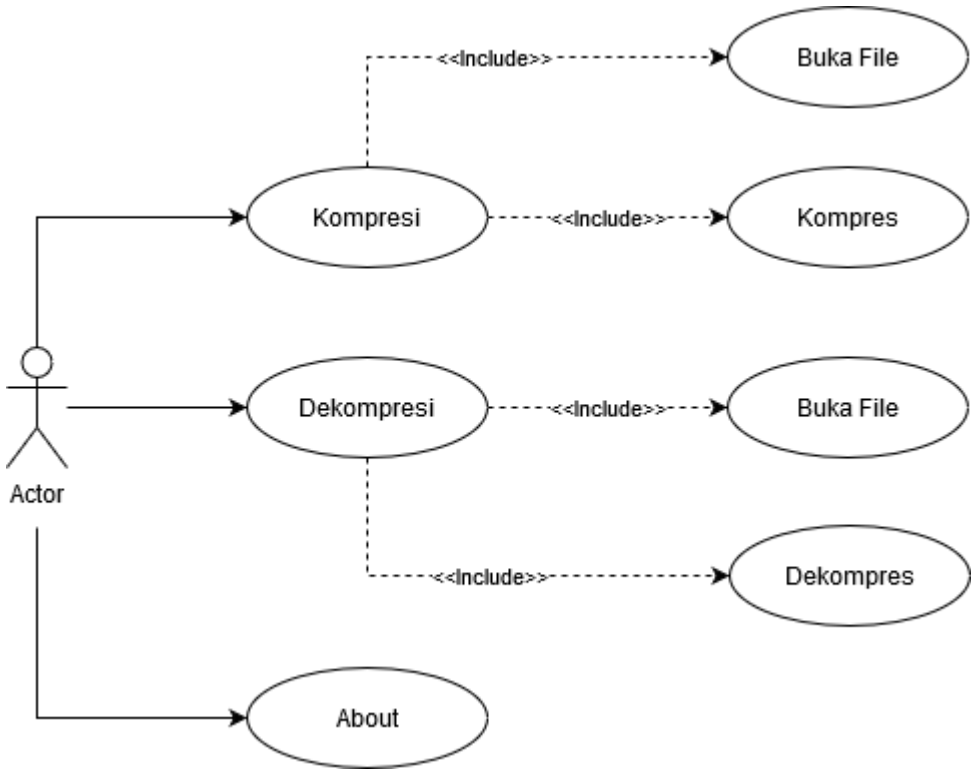
Tahapan ini merupakan proses perancangan aplikasi yang dituangkan kedalam bentuk *use case*, *activity*, *sequence*, *flowchart* dan rancangan antar muka. Penulis dalam tahap ini membuat *use case*, *activity*, *sequence* dan *flowchart* sesuai dengan tahap yang dibutuhkan.

3.3.1. Perancangan Sistem

Pada tahap ini dilakukan perancangan sistem yang dibutuhkan. Perancangan sistem perlu dilakukan agar memberikan gambaran yang jelas dan lengkap rancangan bangun dan implementasi bagaimana sistem dibuat.

3.3.1.1 *Use Case Diagram*

Use-case diagram adalah diagram yang mendeskripsikan interaksi antara *user* dengan sistem. Sebuah *use-case diagram* dapat menggambarkan semua kegiatan didalam satu sistem yang berjalan.



GAMBAR : 3.10. *Use Case Diagram*

Pada gambar 3.10 merupakan interaksi yang terjadi antara *user* dan sistem beserta fungsi-fungsi yang dapat dilakukan didalam sistem. Untuk lebih detailnya akan diuraikan pada spesifikasi *Use Case* yang digambarkan pada tabel 3.5, sampai dengan tabel 3.10.

TABEL : 3.5. Spesifikasi *Use Case* Kompresi

Nama	Kompresi
Aktor	Pengguna
Deskripsi	Pengguna memilih tab kompresi
Pre Kondisi	-
Post kondisi	Sistem menampilkan *Form* untuk proses kompresi
Scenario	Pengguna mengeksekusi tab Kompresi
Alternatif	-

TABEL : 3.6. Spesifikasi *Use Case* Dekompresi

Nama	Dekompresi
Aktor	Pengguna
Deskripsi	Pengguna memilih tab dekompresi
Pre Kondisi	-
Post kondisi	Sistem menampilkan *Form* untuk proses dekompresi
Scenario	Pengguna mengeksekusi tab Dekompresi
Alternatif	-

TABEL : 3.7. Spesifikasi *Use Case About*

Nama	*About*
Aktor	Pengguna
Deskripsi	Pengguna Memilih tab *About*
Pre Kondisi	-
Post kondisi	Sistem menampilkan *Form About*
Scenario	Pengguna mengeksekusi tab *About*
Alternatif	-

TABEL : 3.8. Spesifikasi *Use case* Kompres

Nama	Kompres
Aktor	Pengguna
Deskripsi	Pengguna mengklik tombol kompres
Pre Kondisi	1. Pengguna memilih tab kompresi 2. Pengguna mengklik tombol buka *file*
Post kondisi	Sistem melakukan proses kompresi *audio* WAV
Scenario	1. Pengguna mengklik tombol kompres 2. Sistem melakukan proses kompresi 3. Sistem Menampilkan informasi hasil kompresi 4. Sistem menghasilkan *file output* kompresi
Alternatif	-

TABEL : 3.9. Spesifikasi *Use case* Buka *File* Kompresi / Dekompresi

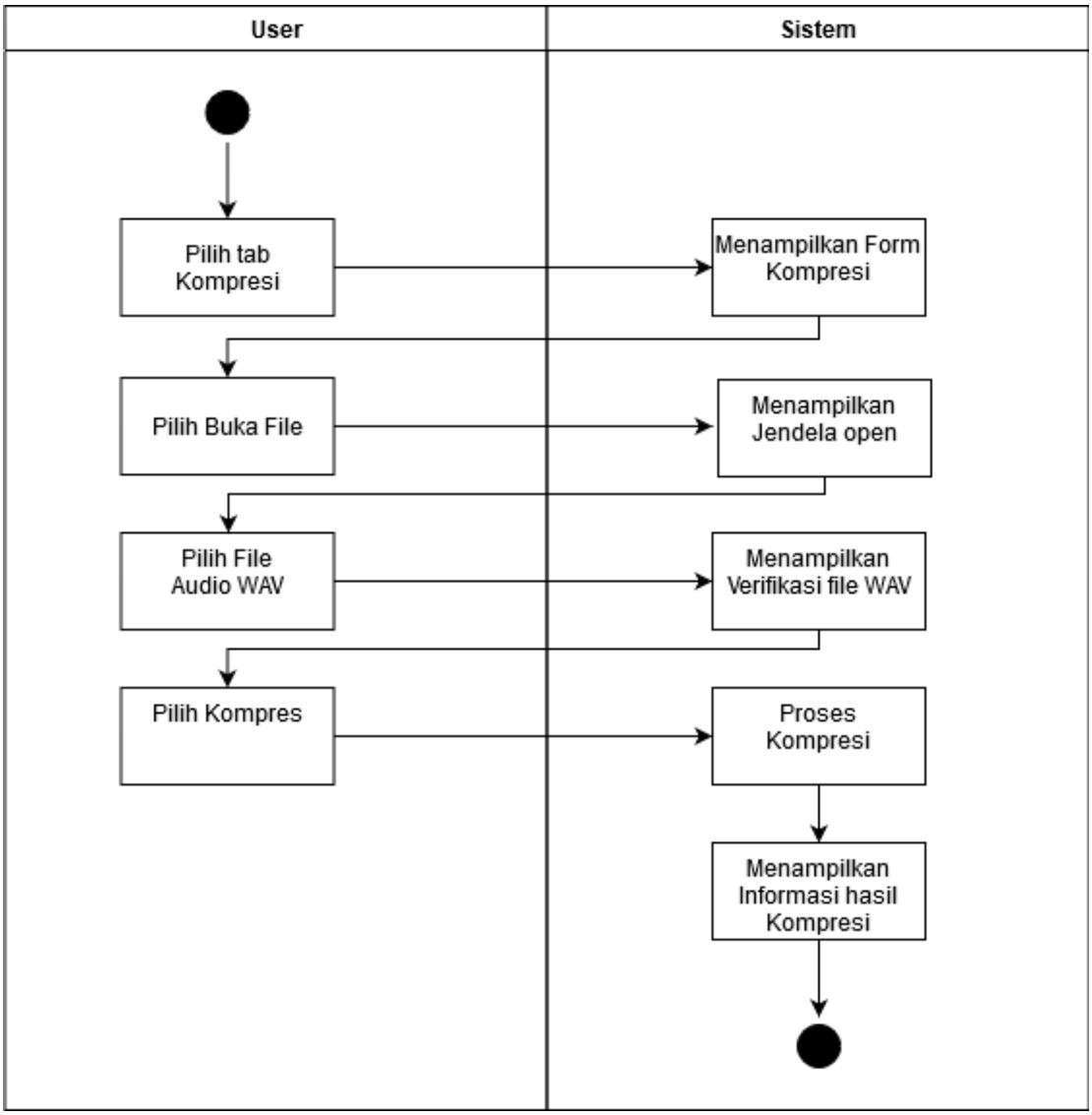
Nama	Buka *file*
Aktor	Pengguna
Deskripsi	Pengguna mengklik tombol *Buka file* pada *form* kompresi / dekompresi
Pre Kondisi	Pengguna memilih tab kompresi / dekompresi
Post kondisi	Sistem menampilkan *file* pada *text box.*
Scenario	1. Pengguna mengklik tombol buka *file* 2. Sistem menampilkan jendela *open file* 3. Pengguna memilih *file*. 4. Pengguna mengklik tombol *open*
Alternatif	-

TABEL : 3.10. Spesifikasi *Use case* Dekompres

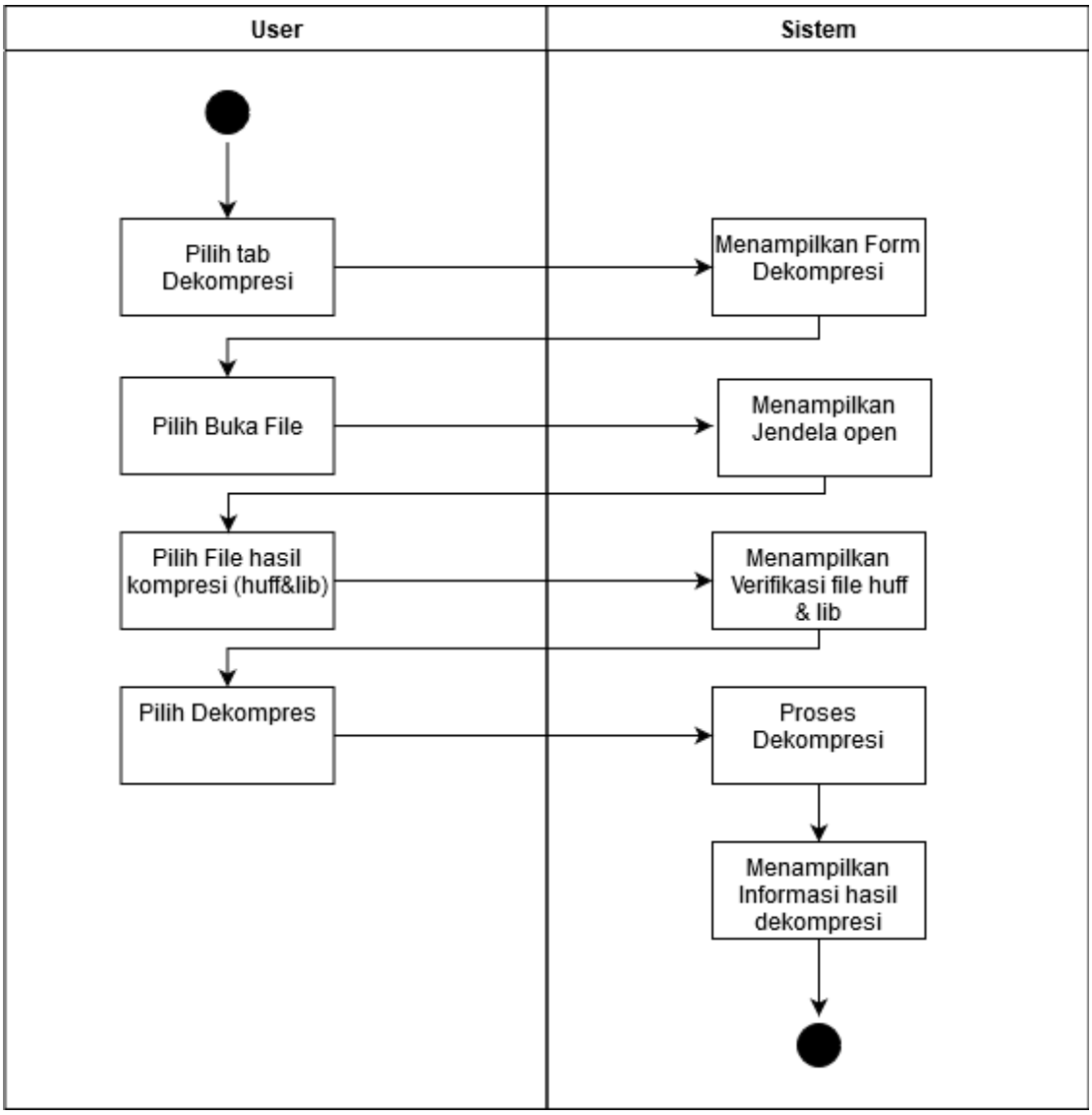
Nama	Dekompres
Aktor	Pengguna
Deskripsi	Pengguna mengklik tombol dekompres pada *Form* dekompresi
Pre Kondisi	1. Pengguna memilih tab dekompresi 2. Pengguna mengklik tombol buka *file*
Post kondisi	Sistem menampilkan hasil dekompresi pada *text box*
	1. Pengguna mengklik tombol dekompres 2. Sistem melakukan proses dekompresi 3. Sistem menampilkan hasil dekompresi
Alternatif	-

3.3.1.2. *Activity Diagram*

Activity diagram adalah suatu diagram yang digunakan untuk menggambarkan aliran proses usaha secara grafis, langkah-langkah dari suatu *use-case*, atau logika dari suatu tingkah laku objek (*method*).



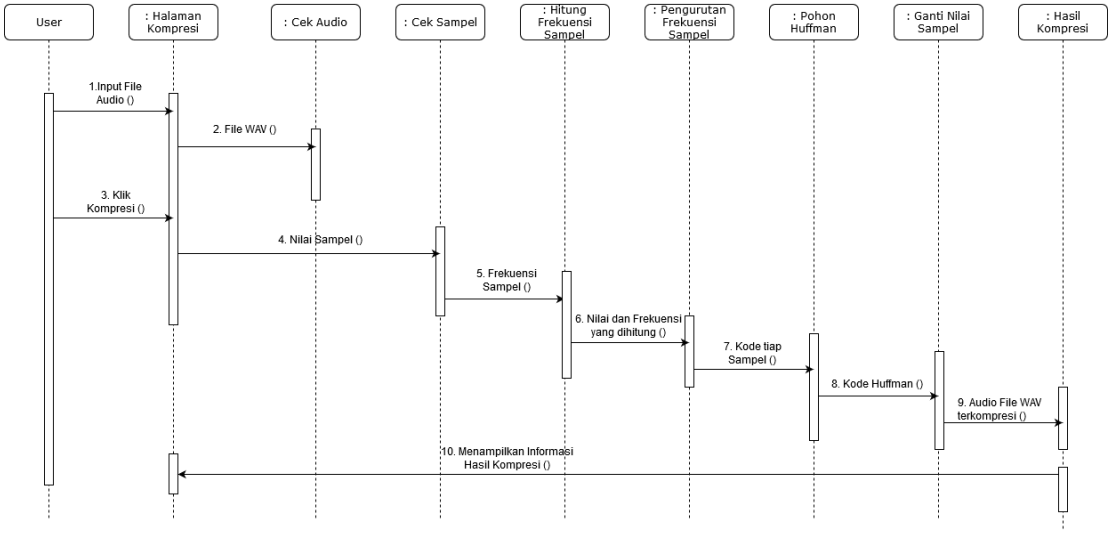
GAMBAR : 3.11. *Activity Diagram* Kompresi



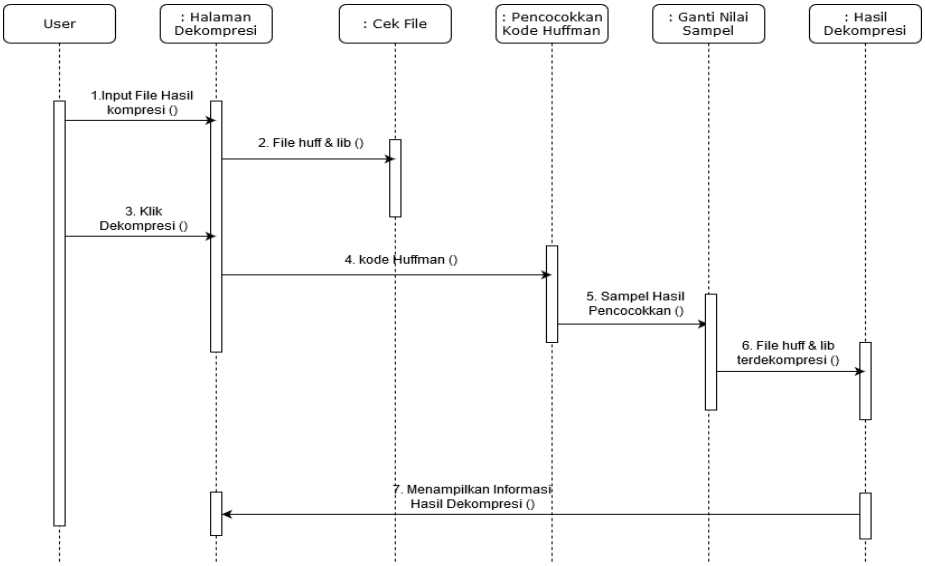
GAMBAR : 3.12. *Activity Diagram* Dekompresi

3.3.1.3. *Sequence Diagram*

Sequence diagram adalah diagram yang memodelkan logika sebuah *use-case* dengan cara menggambarkan interaksi pesan di antara objek – objek dalam rangkaian waktu.



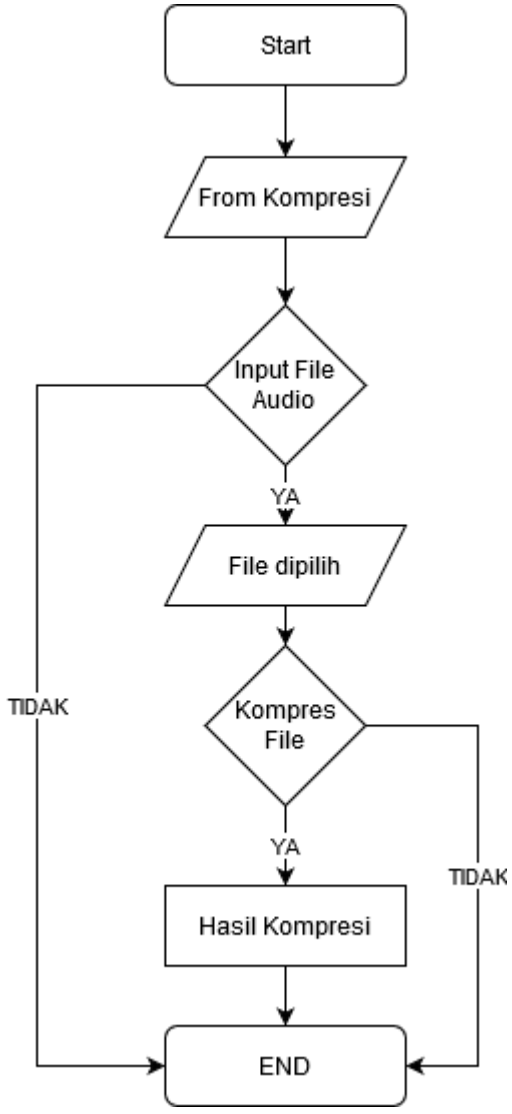
GAMBAR : 3.13. *Sequence Diagram* Kompresi



GAMBAR : 3.14. *Sequence Diagram* Dekompresi

3.3.1.4. *Flowchart* Kompresi

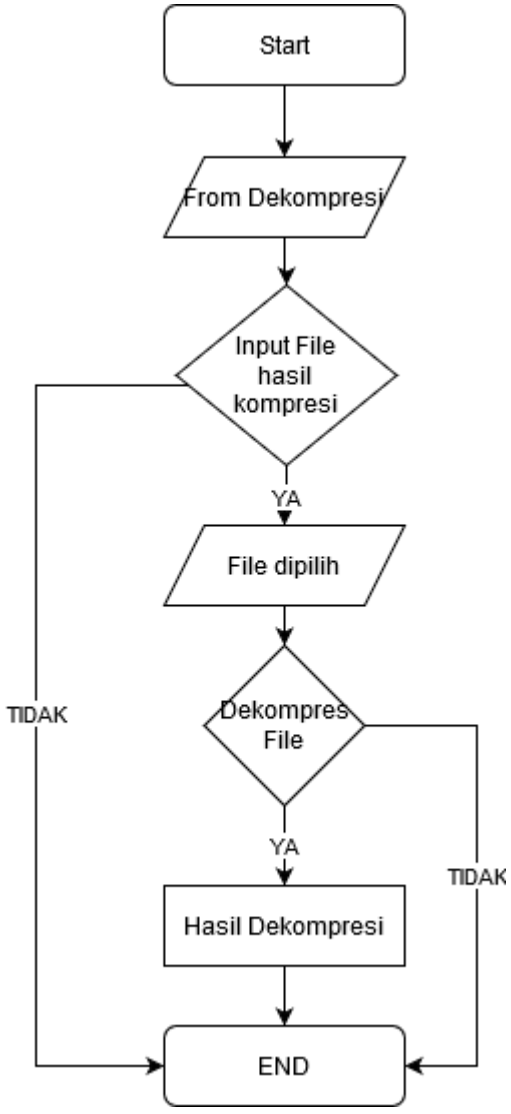
Gambar dibawah ini merupakan *flowchart* proses kompresi *file* dari aplikasi yang di rancang.



GAMBAR : 3.15. *Flowchart* Proses Kompresi

3.3.1.5. *Flowchart* Dekompresi

Gambar dibawah ini merupakan *flowchart* proses dekompresi *file* dari aplikasi yang di rancang.



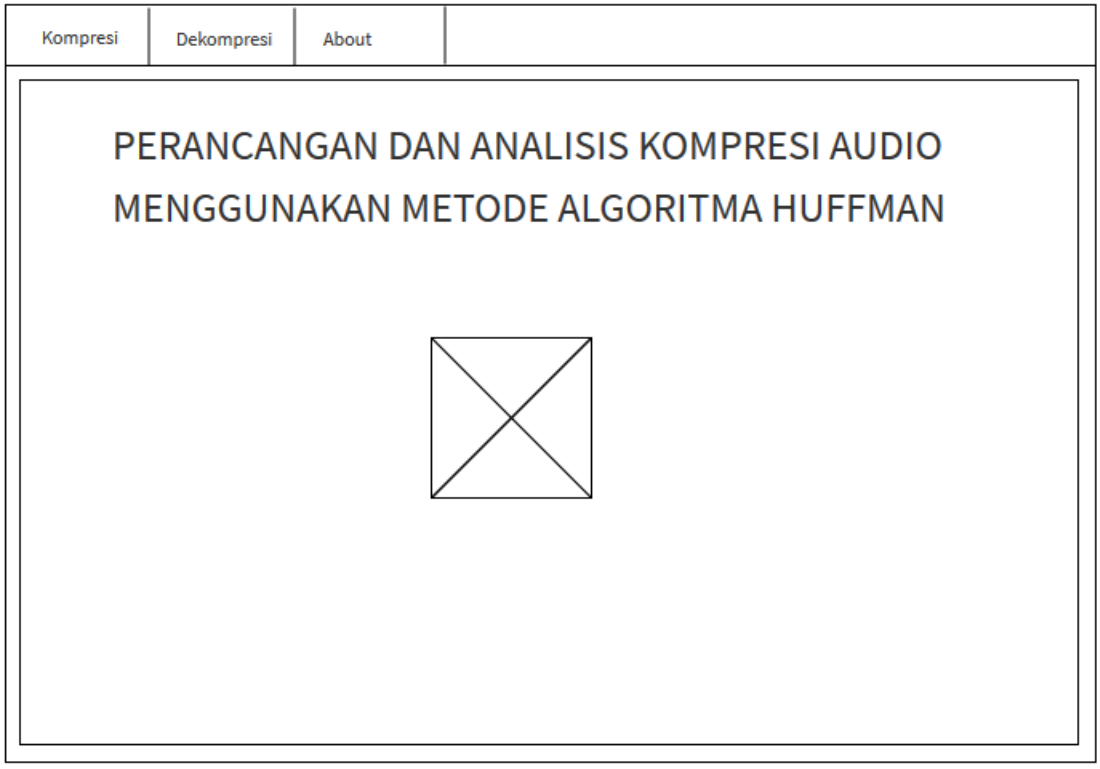
GAMBAR : 3.16. *Flowchart* Proses Dekompresi

3.3.2. Desain Atar muka

Dalam sistem ini terdapat 4 halaman, yaitu halaman utama, halaman Kompresi, halaman Dekompresi, dan halaman *About*.

3.3.2.1. Rancangan Antarmuka Halaman Utama

Halaman utama merupakan tampilan awal ketika aplikasi pertama kali dijalankan. Pada halaman ini akan di tampilkan judul skripsi, keterangan mahasiswa (Nama dan Nim), menu *strip* (Kompresi, Dekompresi dan *About*) serta Logo STMIK-IM. Rancangan halaman utama dapat dilihat pada Gambar 3.17.



GAMBAR : 3.17. Rancangan Halaman Utama

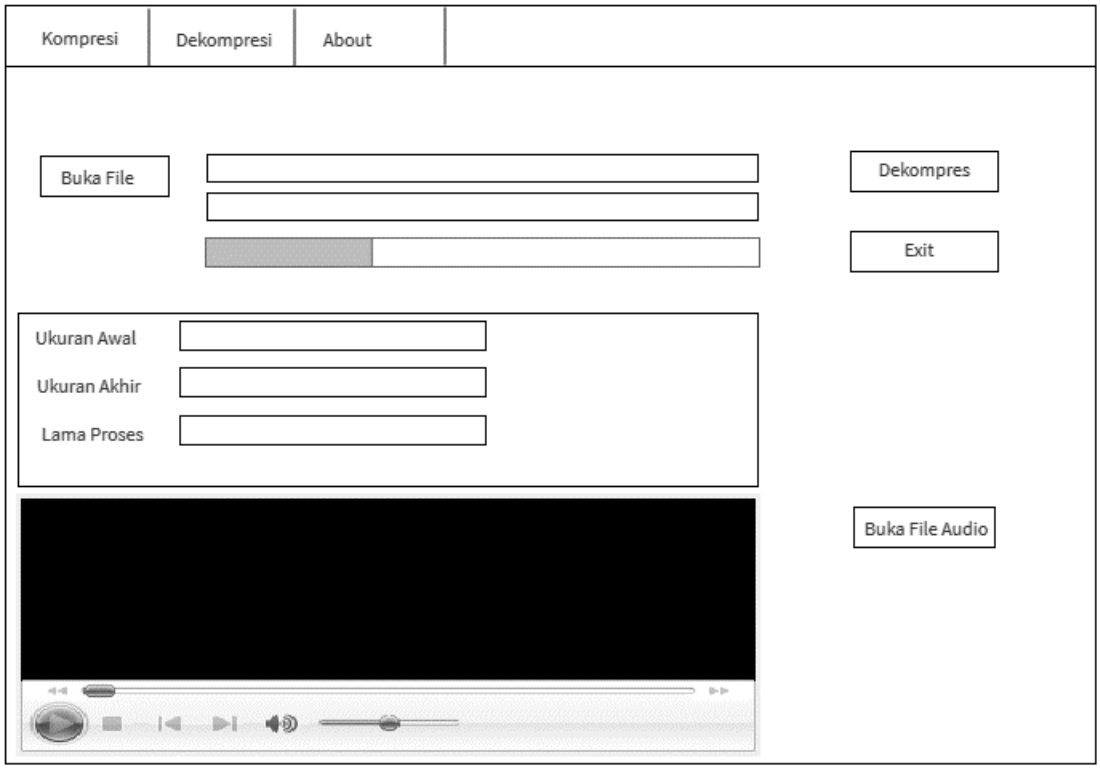
3.3.2.2. Rancangan Antarmuka Halaman Kompresi

Halaman Kompresi berfungsi untuk melakukan proses Kompresi pada *file audio*. Tampilan rancangan halaman kompresi dapat dilihat pada gambar 3.18.

GAMBAR : 3.18. Halaman Kompresi

3.3.2.3. Rancangan Antarmuka Halaman Dekompresi

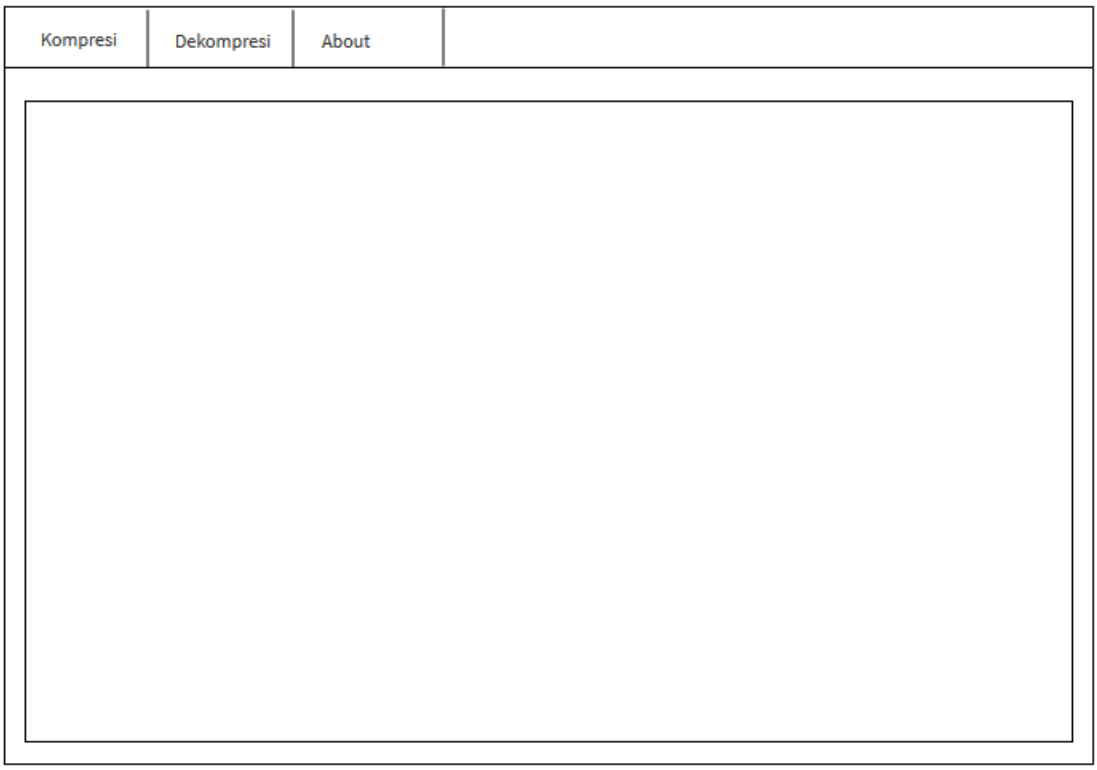
Halaman dekompresi berfungsi untuk melakukan proses dekompresi pada *file audio* dan juga berfungsu untuk melakukan tes *file* hasil dekompresi. Tampilan rancangan halaman dekompresi dapat dilihat pada gambar 3.19.



GAMBAR : 3.19. Halaman Dekompresi

3.3.2.4. Rancangan Antarmuka Halaman *About*

Halaman *About* terdapat *grup box* yang berisi keterangan tentang algoritma *Huffman* dan aplikasi. Halaman *about* dapat dilihat pada Gambar 3.20.



GAMBAR : 3.20. Halaman *About*

BAB IV

IMPLEMENTASI DAN UJI COBA

4.1. *Construction (Code & Test)*

Pada tahap ini hasil rancangan akan diimplemtasikan menjadi sebuah aplikasi berbasis *desktop* yang dibangun menggunakan bahasa pemograman *VB.Net* 2010, sistem ini memiki dua proses yaitu proses kompresi dan dekompresi.

4.1.1. Implemetasi *Hardware & Software*

Implementasi *Hardware & Software* mengacu pada hasil analisis perangkat keras & lunak seperti yang terurai pada Bab III. Untuk lebih jelasnya akan dijelaskan dibawah ini :

4.1.1.1. Perangkat Keras (*Hardware*)

Spesifikasi *Hardware* yang digunakan aplikasi ini adalah :

1. *Processor* AMD A9-9400 (2Cpu) 2.4 GHz.
2. *Memory* (RAM) 4GB.
3. *Harddisk* Minimal 6GB.

4.1.1.2. Perangkat Lunak (*Software*)

1. Sistem Operasi *Windows* 10
2. *Microsoft Visual Basic* 2010

4.1.2. Implementasi Algoritma

Pada aplikasi ini terdiri dari 2 proses utama yaitu kompresi dan dekompresi. Pada proses kompresi dilakukan beberapa tahapan seperti pembacaan jumlah simbol, mengurutkan setiap simbol, pembetukkan pohon *Huffman* dan terakhir adalah menggabungkan 2 *node* pohon. Selanjutnya pada proses dekompresi hanya melakukan pengembalian dan pencarian data yang sebelumnya sudah di simpan pada *library*.

4.1.2.1. Implementasi Proses kompresi

1. Proses pembacaan

Proses pembacaan ini dilakukan dengan membaca seluruh simbol yang ada dalam *file* yang di kompres dan menyimpan jumlah tiap-tiap simbol pada posisi *array*. Apabila ada simbol sama maka jumlah simbol dalam *array* sebelumnya di tambah satu. Proses ini dinyatakan dalam algoritma berikut :

```
Using file As New IO.FileStream(TB_BF.Text, IO.FileMode.Open)
          Dim nilai As Integer = file.ReadByte()
          Do Until nilai = -1
              Node(nilai).Jumlah = Node(nilai).Jumlah + 1
              nilai = file.ReadByte()
          Loop
      End Using
```

2. Proses pengurutan

Setelah seluruh simbol dalam *file* dibaca, dan disimpan selanjutnya adalah proses pengurutan simbol-simbol berdasarkan frekuensi kemunculannya. Proses ini dinyatakan dalam algortima berikut :

```
For I = 0 To Indeks - 1
          For j = 0 To Indeks - 2
              If Node(j).Jumlah > Node(j + 1).Jumlah Then
                  Temp = Node(j)
                  Node(j) = Node(j + 1)
                  Node(j + 1) = Temp
                  Temp = New Nodenya(0, 0)
              End If
          Next
      Next
```

3. Proses pembentukkan pohon *Huffman*

Proses pembentukkan pohon *Huffman* ini di tentukan berdasarkan jumlah *level* pohon *Huffman*. Indeks2 adalah proses yang digunakan untuk pencatatan jumlah *level* pada pohon *Huffman*, nilai ini di tentukan dengan penjumlahan Indeks2 = Indeks2 + 1. Proses ini dinyatakan dalam algoritma berikut :

```
Indeks2 = -1
        Do While Node(Indeks2 + 1).Jumlah = 0
            Indeks2 = Indeks2 + 1
        Loop
```

4. Proses pembentukkan dan penggambungan 2 *node* pohon

Proses pembentukkan *node* (cabang) adalah dengan dengan menggabungkan 2 *node* dan membuat *node* baru yang merupakan gabungan dari 2 *node* tersebut, dimana *node* gabungan ini memiliki *node* pada *level* di bawahnya yang berisi 2 *node* yang digabungkan tersebut.

```
Do
     Indeks = Indeks - 1
     Temp.Kiri = Node(Indeks2 + 1)
     Temp.Kanan = Node(Indeks2 + 2)
     Node(Indeks2 + 1) = Temp
     Node(Indeks2 + 2) = Nothing
     Temp = New Nodenya(0, 0)

     For i = Indeks2 + 2 To Indeks - 1
         Node(i) = Node(i + 1)
         Node(i + 1) = Nothing
     Next

     For i = Indeks2 + 1 To Indeks - 1
         For j = Indeks2 + 1 To Indeks - 2
             If Node(j).Jumlah >= Node(j + 1).Jumlah Then
                 Temp = Node(j)
                 Node(j) = Node(j + 1)
                 Node(j + 1) = Temp
                 Temp = New Nodenya(0, 0)
             End If
         Next
     Next
 Loop Until Indeks = Indeks2 + 2
```

4.1.2.2. Implemtasi Proses Dekompresi

Dekompresi adalah proses pengembalian *file* dari proses kompresi ke bentuk awal (asli). Proses pertama adalah memeriksa apakah *file* adalah hasil kompresi dengan menggunakan metode algortima *Huffman*, jika benar maka proses akan di lanjutkan. Proses dekompresi dapat dilihat pada algoritma berikut :

```
Dim KamusKode(256) As String
        Dim KamusData(256) As String
        Dim Indekskamus As Integer
        Indekskamus = 0
        Dim reader = My.Computer.FileSystem.ReadAllText(TB_FA.Text)
        Dim Datanya As String() = reader.Split(New Char() {"."c})
        Dim Datanyanya As String
        Dim FromTime As Date = DateTime.Now
        For Each Datanyanya In Datanya
            If (Indekskamus <> Datanya.Length - 1) Then
                Dim Datanya2 As String() = Datanyanya.Split(New Char()
{";"c})
                KamusData(Indekskamus) = Datanya2(0).ToString
                KamusKode(Indekskamus) = Datanya2(1).ToString
                Indekskamus = Indekskamus + 1
            End If
        Next
        Dim bytes As Byte() = IO.File.ReadAllBytes(TB_BF.Text)
        Dim hex As String() = Array.ConvertAll(bytes, Function(b)
b.ToString)
        Dim akhir As Integer
        Dim l As String
        Dim cari As String
        Dim cari1 As String
        Dim Indekscari As Integer
        Dim Indekscari2 As Integer
        Dim jumlah As Integer
        cari = ""
        cari1 = ""
        akhir = Integer.Parse(hex(hex.Length - 1))
        jumlah = 1
        PB_dekompres.Value = 0
        Using filesave As New IO.FileStream(TB_BF.Text +
"[Dekompresi].wav", IO.FileMode.Create)
            For i = 0 To hex.LongLength - 3
                l = hex(i)
                Indekscari = 0
                Indekscari2 = -1
                cari = ToBinary(Integer.Parse(l))
                Do
                    cari1 = cari1 & cari(Indekscari)
                    Do
                        Indekscari2 = Indekscari2 + 1
                        If String.Equals(KamusKode(Indekscari2), cari1)
Then

filesave.WriteByte(Integer.Parse(KamusData(Indekscari2)))
                            cari1 = ""
                            Indekscari2 = Datanya.Length - 2
                        End If
                    Loop Until Indekscari2 = Datanya.Length - 2
                    Indekscari2 = -1
```

```
   Indekscari = Indekscari + 1
                Loop Until Indekscari = cari.Length
                jumlah = jumlah + 1
                PB_dekompres.Value = (jumlah / FileSizeAwal) * 100
            Next

            Indekscari = 0
            Indekscari2 = -1
            cari = ToBinary(Integer.Parse(hex(hex.LongLength - 2)))
            cari = cari.Substring(0, cari.Length - akhir)
            Do
                cari1 = cari1 & cari(Indekscari)
                Do
                    Indekscari2 = Indekscari2 + 1
                    If String.Equals(KamusKode(Indekscari2), cari1) Then

filesave.WriteByte(Integer.Parse(KamusData(Indekscari2)))
                        cari1 = ""
                        Indekscari2 = Datanya.Length - 2
                    End If
                Loop Until Indekscari2 = Datanya.Length - 2
                Indekscari2 = -1
                Indekscari = Indekscari + 1
            Loop Until Indekscari = cari.Length
        End Using
```

4.1.3. Implemetasi Antar Muka

Implementasi Antar muka merupakan pemaparan mengenai tampilan aplikasi dan kegunaan fungsi dari setiap *Form* yang ada. Untuk memperjelas bentuk dari implementasi antarmuka, berikut pemaparan dan fungsi dari setiap tampilan yang telah dibuat.

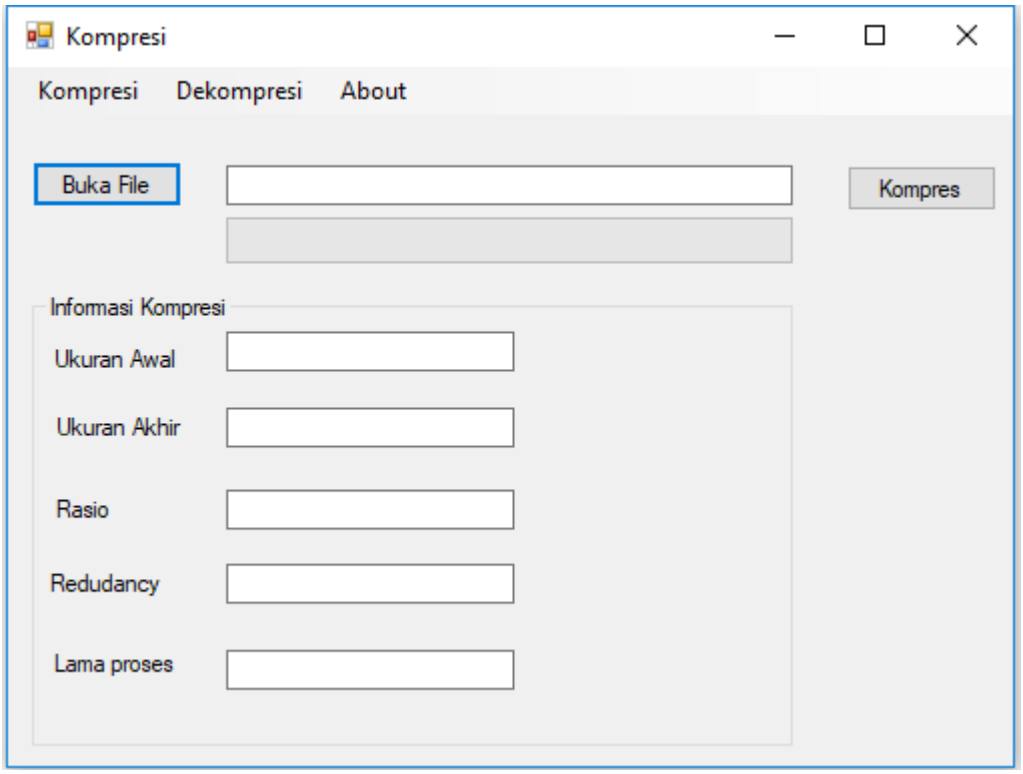
1. Tampilan Utama

Berikut ini merupakan tampilan yang muncul paling awal saat *program* dijalankan. Tampilan ini berisi Judul Skripsi, Logo, Nama dan Menu.

GAMBAR : 4.1. Tampilan Utama

2. Tampilan Menu kompresi

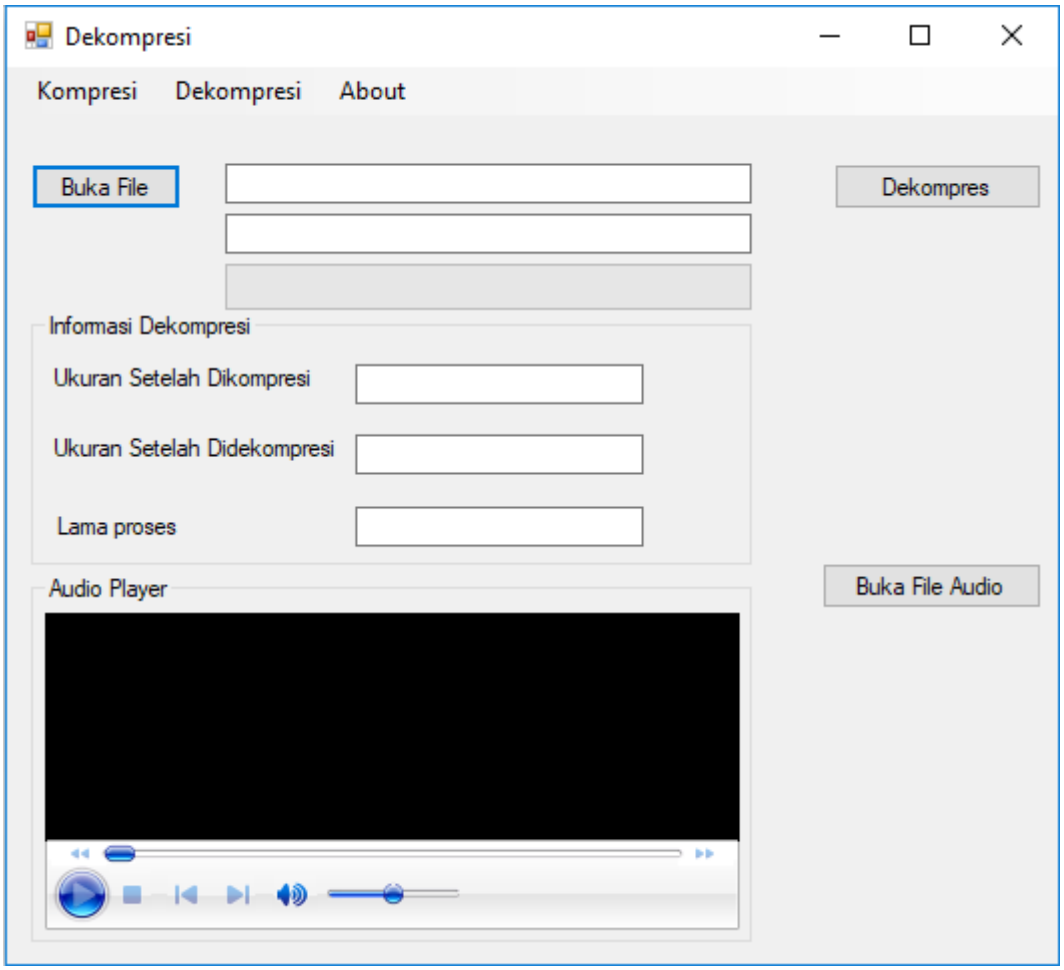
Pada gambar berikut ini merupakan tampilan dari menu kompresi, dimana menu ini berfungsi untuk melakukan kompresi *audio* dengan menggunakan metode *Huffman*.



GAMBAR : 4.2. Tampilan Menu Kompresi

3. Tampilan Menu Dekompresi

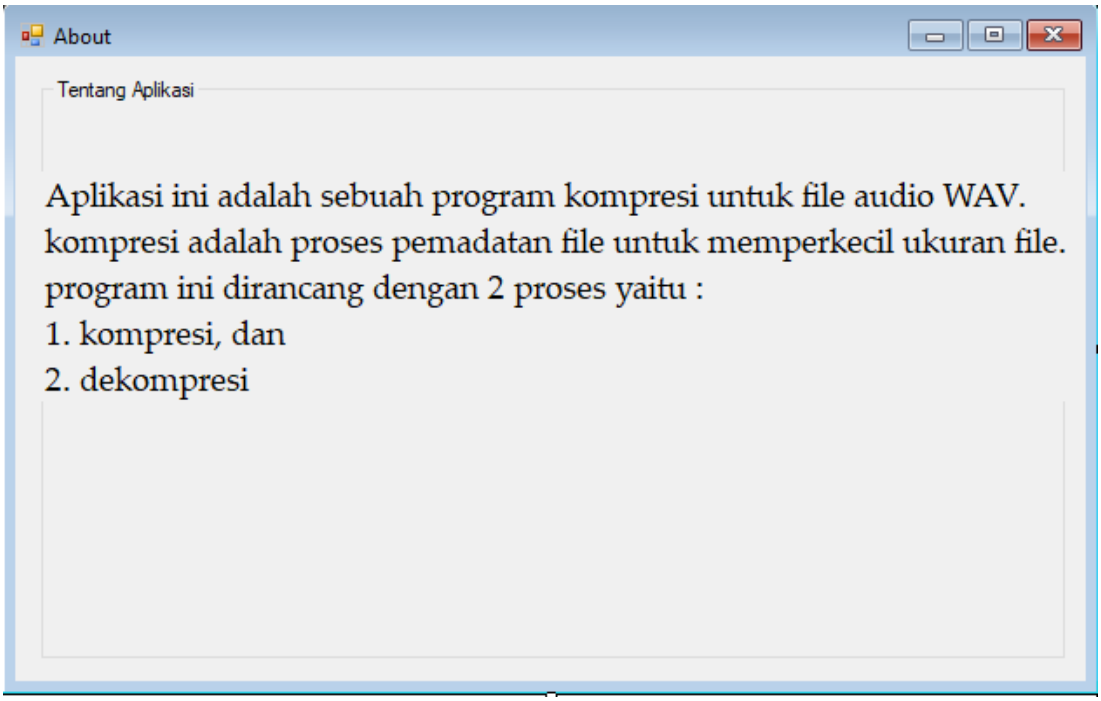
Pada gambar berikut ini merupakan tampilan dari menu dekompresi, dimana menu ini berfungsi untuk melakukan dekompresi *audio*.



Gambar : 4.3. Tampilan Menu Dekompresi

4. Tampilan Menu *About*

Pada gambar berikut ini merupakan tampilan dari menu *About*, dimana menu ini berisi penjelesan mengenai Aplikasi.



GAMBAR : 4.4. Tampilan Menu *About*

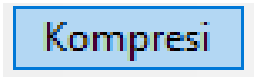
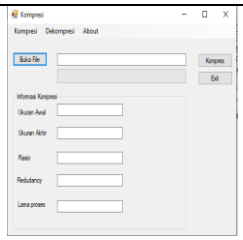
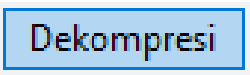
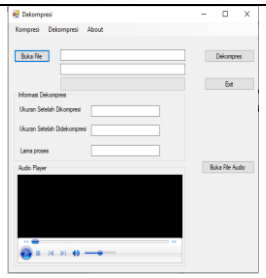
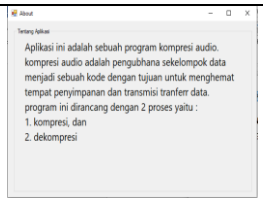
4.1.4. *Testing*

Pada tahap ini, sistem akan di uji apakah sudah sesuai dengan fungsi-fungsi yang telah dirancang pada saat tahap analisis dan perancangan. *Testing* ini dilakukan pada *file audio* berformat WAV.

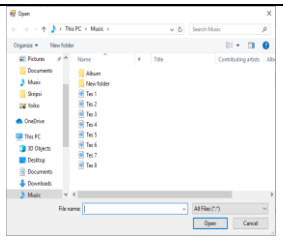
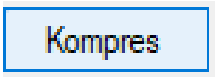
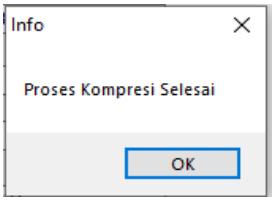
4.1.4.1. *Testing Black Box*

Testing ini dilakukan dengan tujuan untuk menjamin apakah sistem yang dibuat sesuai dengan apa yang diharapkan. Pengujian dibagi tiga yaitu pengujian pada halaman beranda, pengujian pada *form* kompresi dan pengujian pada *form* dekompresi. Berikut hasil pengujian yang digambarkan pada tabel 4.1, sampai dengan tabel 4.3.

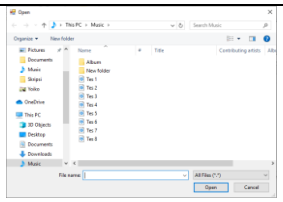
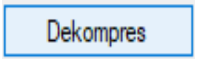
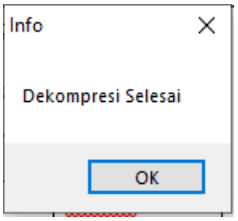
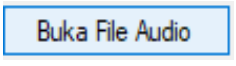
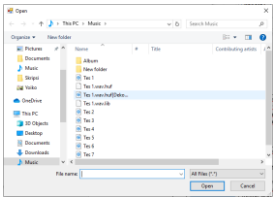
TABEL : 4.1. Tabel pengujian *black box* pada halaman beranda

No	Pengujian sistem aplikasi	Hasil yang diharapkan	Hasil yang ditampilkan Sistem Aplikasi	Kesimpulan
1.	*User* mengklik tab kompresi 	Aplikasi akan menampilkan *form* kompresi.		*Valid*
2.	*User* mengklik tab dekompresi 	Aplikasi akan menampilkan *form* dekompresi.		*Valid*
3.	*User* mengklik tab *about* 	Aplikasi akan menampilkan *form about*.		*Valid*

TABEL : 4.2. Tabel pengujian *black box* pada *form* kompresi

No	Pengujian sistem aplikasi	Hasil yang diharapkan	Hasil yang ditampilkan Sistem Aplikasi	Kesimpulan
1.	*User* mengklik tombol buka *file* 	Aplikasi akan membuka kotak dialog untuk memilih *file audio* dengan format WAV		*Valid*
2.	*User* mengklik tombol kompres 	Aplikasi akan mengkompres *file* WAV dan menampilkan pesan proses kompresi selesai.		*Valid*

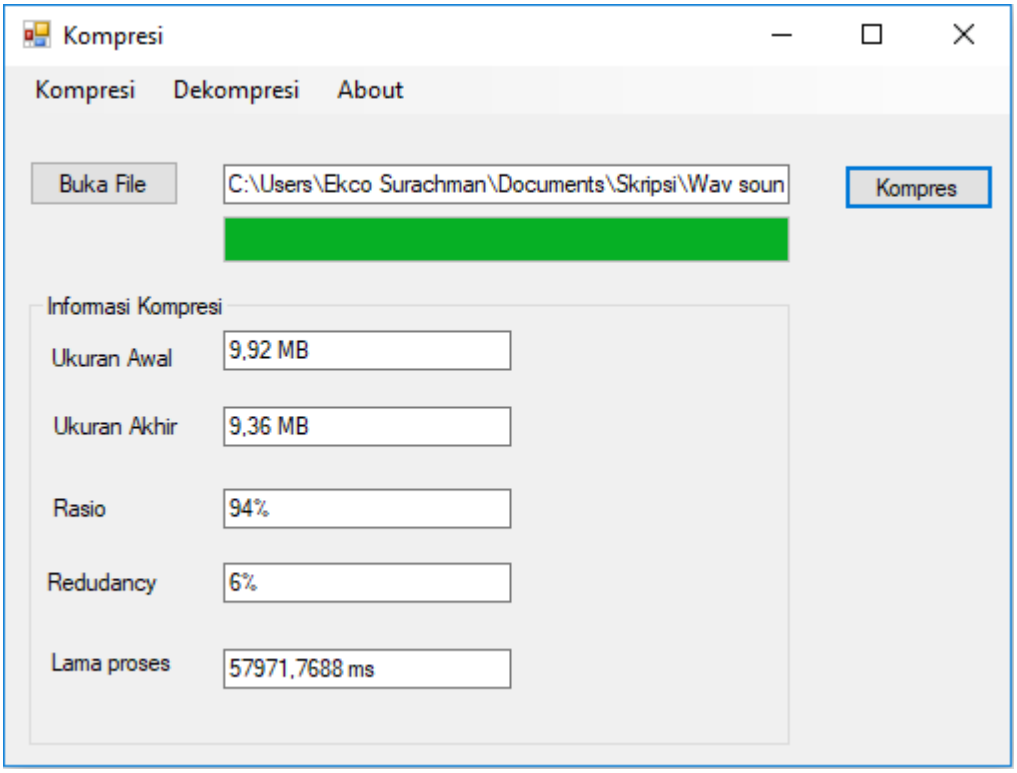
TABEL : 4.3. Tabel pengujian *black box* pada *form* dekompresi

No	Pengujian sistem aplikasi	Hasil yang diharapkan	Hasil yang ditampilkan Sistem Aplikasi	Kesimpulan
1.	*User* mengklik tombol buka *file* 	Aplikasi akan membuka kotak dialog untuk memilih *file* (**.huf*) dan (**.lib*)		*Valid*
2.	*User* mengklik tombol dekompres 	Aplikasi akan mengkompres *file* WAV dan menampilkan pesan dekompresi selesai.		*Valid*
3.	*User* mengklik tombol buka *file audio* 	Aplikasi akan membuka kotak dialog untuk memilih *file audio* dengan format WAV hasil dari dekompresi.		*Valid*

4.1.4.2. *Testing* Proses kompresi

Untuk melakukan proses kompresi tahap awal yang dilakukan adalah memilih tab Kompresi. Setelah tampilan *form* kompresi muncul maka lakukan langkah-langkah berikut untuk melakukan proses kompresi.

1. Menekan tombol pada Buka *File* untuk membuka *File Dialog*, dan pilih *file* audio (*.*Wav*) sebagai *file* masukan.
2. Menekan tombol kompresi untuk melakukan proses kompresi.
3. Setelah proses selesai aplikasi akan menampilkan informasi hasil kompresi seperti pada gambar 4.5 berikut ini :

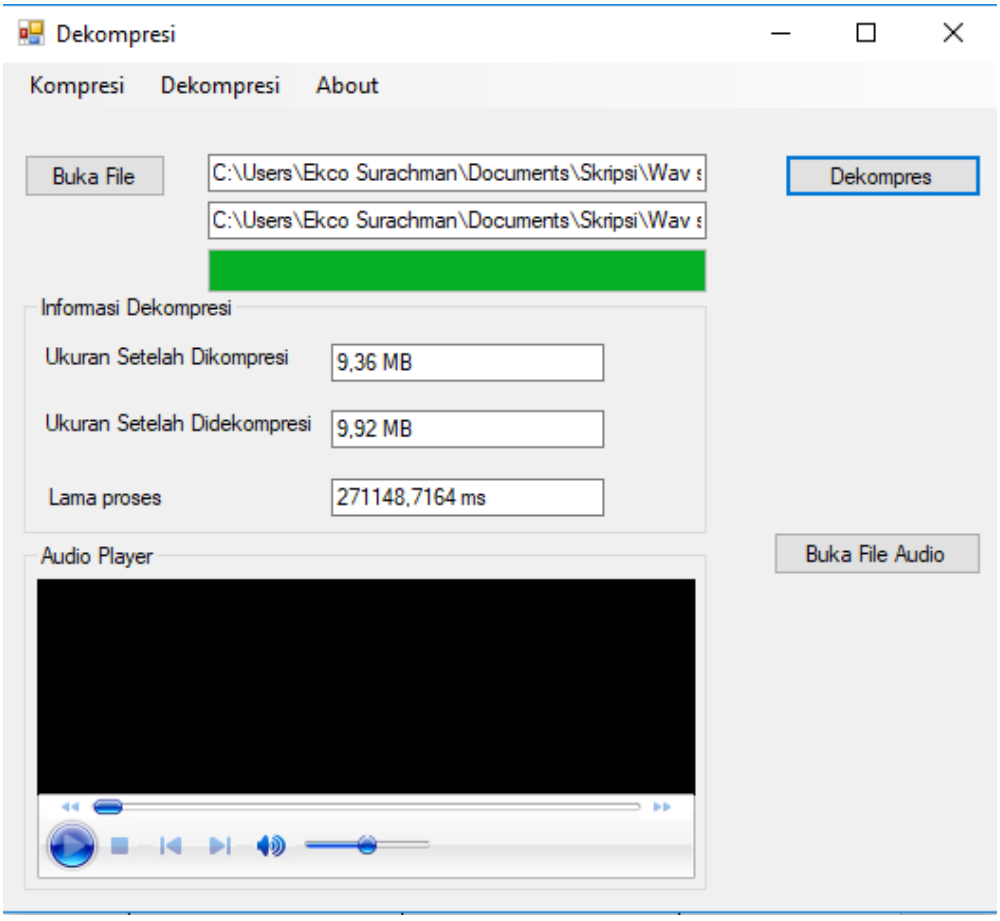


GAMBAR : 4.5. *Testing* proses Kompresi

4.1.4.3. *Testing* Proses Dekompresi

Untuk melakukan proses dekompresi tahap awal yang dilakukan adalah memilih tab Dekompresi. Setelah tampilan *form* dekompresi muncul maka lakukan langkah-langkah berikut untuk melakukan proses dekompresi.

1. Buka *file* yang telah dikompresi sebelumnya dan *library*.
2. Selajutnya klik tombol dekompresi untuk melakukan proses dekompresi.
3. Setelah proses selesai aplikasi akan menampilkan informasi hasil dekompresi seperti pada gambar 4.6 berikut ini :



GAMBAR : 4.6. *Testing* proses Dekompresi

Dari hasil *testing* proses kompresi diatas diperoleh ukuran *file* hasil kompresi, rasio, *redundancy* dan waktu yang dibutuhkan selama proses kompresi. Hasil pengujian tersebut akan digunakan untuk mengetahui berapa rasio dan ukuran ukuran *file* setelah dikompresi, dan berapa lama waktu proses kompresi.

4.1.4.4. Hasil Pengujian Proses Kompresi

Hasil kompresi akan ditentukan dari besar kecilnya persentase rasio yang di dapat dimana semakin kecil rasio maka semakin baik hasil kompresi. Persentase rasio kompresinya dapat dinyatakan sebagai berikut :

1. Rasio 100% - 88% = Kurang baik.
2. Rasio 87% - 50% = Baik.
3. Rasio 49% - 0% = Sangat baik.

TABEL : 4.4. Hasil pengujian proses Kompresi

Nama *File*	Besar Data Sebelum di kompresi (MB)	Besar Data Sesudah di kompresi (MB)	Rasio (%)	*Redudancy* (%)	Waktu kompresi (ms)	Keterangan
Tes 01.wav	9,92	9,36	94	6	57971,7688	Kurang baik
Tes 02.wav	2,13	1,73	81	19	9217,2025	Baik
Tes 03.wav	3,8	2,82	74	26	15058,1261	Baik
Tes 04.wav	2,26	1,84	81	19	10290,2011	Baik
Tes 05.wav	2,47	2,15	87	13	11786,8897	Baik
Tes 06.wav	6,42	5,57	87	13	29436,6094	Baik

Lanjutan TABEL : 4.4. Hasil pengujian proses Kompresi

Tes 07.wav	14,64	12,88	88	12	83714,2745	Kurang Baik
Tes 08.wav	9,38	6,12	65	35	32863,6474	Baik
Tes 09.wav	2,77	2,48	90	10	14811,3344	Kurang baik
Tes 10.wav	2,2	1,81	82	18	11254,5319	Baik
Tes 11.wav	3,35	2,68	80	20	16156,0255	Baik
Tes 12.wav	1,09	0,97	89	11	6593,1894	Kurang Baik
Tes 13.wav	1,67	1,33	79	21	7027,7307	Baik
Tes 14.wav	1,46	1,15	79	21	6901,376	Baik
Tes 15.wav	6,42	5,57	87	13	27502,8788	Baik
Tes 16.wav	3,84	3,4	89	11	16889,2098	Kurang Baik
Tes 17.wav	5,55	4,45	80	20	23366,8159	Baik
Tes 18.wav	2,77	2,29	83	17	11681,2084	Baik
Tes 19.wav	2,49	1,81	72	28	10266,8812	Baik
Tes 20.wav	2,70	2,42	90	10	13356,2078	Kurang Baik

Dari Tabel 4.4 dapat dilihat bahwa hasil kompresi dari *file audio* WAV dengan menggunakan metode *Huffman* memiliki rata-rata rasio sebesar 82,85% dengan nilai rasio terkecil adalah 65 % dan rasio tertinggi adalah 94%. Berdasarkan hasil tersebut dapat diketahui bahwa ukuran *file* yang dihasilkan setelah dilakukan kompresi berkurang sebesar 17,15% dari ukuran aslinya.

4.1.4.5. Hasil Pengujian Proses Dekompresi

TABEL : 4.5. Hasil pengujian proses Dekompresi

Nama *File*	Besar data Awal (Mb)	Ukuran *File* Setelah Dikompresi (Mb)	Ukuran *File* Setelah Didekompresi (Mb)	Waktu Dekompresi (ms)
Tes 01.wav	9,92	9,36	9,92	271148,7164
Tes 02.wav	2,13	1,73	2,13	38005,5058
Tes 03.wav	3,8	2,82	3,8	76829,4564
Tes 04.wav	2,26	1,84	2,26	47273,8731
Tes 05.wav	2,47	2,15	2,47	61833,4617
Tes 06.wav	6,42	5,57	6,42	157444,7005
Tes 07.wav	14,64	12,88	14,64	374269,0662
Tes 08.wav	9,38	6,12	9,38	134538,9318
Tes 09.wav	2,77	2,48	2,77	53796,4692
Tes 10.wav	2,2	1,81	2,2	45687,9653
Tes 11.wav	3,35	2,68	3,35	58125,8814
Tes 12.wav	1,09	0,97	1,09	18819,1791
Tes 13.wav	1,67	1,33	1,67	24310,8895
Tes 14.wav	1,46	1,15	1,46	22785,4109
Tes 15.wav	6,42	5,57	6,42	124725,3222
Tes 16.wav	3,84	3,4	3,84	69585,4548
Tes 17.wav	5,55	4,45	5,55	113704,1681
Tes 8.wav	2,77	2,29	2,77	43164,3557
Tes 19.wav	2,49	1,81	2,49	43679,5879
Tes 20.wav	2,70	2,42	2,70	52890,2144

Dari Tabel 4.5 dapat dilihat dari hasil dekompresi bahwa *file* kembali ke ukuran semula / asli dengan ini dapat dikatakan bahwa proses dekompresi berhasil.

4.1.5. Hasil Analisis

Dari hasil pengujian proses kompresi didapat bahwa rasionya mempunyai besaran antara 65% untuk nilai terkecil dan 94% untuk nilai tertinggi. Jika dicari hasil rasio kompresi tersebut secara rata-rata adalah sebesar 82,85%. Ini berarti ukuran *file* hasil adalah 0,8285 kali ukuran *file* semula dan pengurangan ukuran *file* sebesar (100% - 82.85%) = 17,15 %.

Dari hasil tersebut juga menunjukkan bahwa persentase kompresi atau dekompresi *file* tidak bergantung pada ukuran *file* melainkan bergantung pada isi data pada *file* tersebut. Semakin banyak perulangan data yang terdapat pada bagian *chunk* data *file* maka rasio kompresi akan semakin rendah.

Dari beberapa pengujian yang dilakukan tingkat kecepatan baik untuk proses kompresi dan dekompresi berbanding lurus dengan ukuran *file* , artinya semakin besar ukuran *file* yang diproses maka semakin lama proses berlangsung. Dari hasil 20 ujicoba yang dilakukan didapat hasil bahwasanya metode *Huffman* memiliki rata-rata rasio sebesar 82,85 % ini membuktikan bahwasanya metode *Huffman* cukup baik dalam mengkompresi *file audio* WAV.

BAB V

PENUTUP

5.1. Kesimpulan

Berdasarkan hasil pembahasan beserta penelitian yang telah dilakukan, maka dapat diambil beberapa kesimpulan, diantaranya sebagai berikut :

1. Aplikasi yang dirancang untuk kompresi *file audio* WAV dengan menggunakan metode *Huffman* telah berhasil direalisasikan dan bekerja dengan baik.
2. Berdasarkan hasil pengujian proses kompresi didapat bahwa rasio mempunyai besaran antara 65 % untuk nilai terkecil dan 94% untuk nilai tertinggi. Dengan rata-rata rasio sebesar 82.85%. Ini berarti ukuran *file* hasil adalah 0,8285 kali ukuran *file* semula dan pengurangan ukuran *file* sebesar 17.15 %. Sedangkan untuk tingkat kecepatan proses kompresi dan dekompresi berbanding lurus dengan ukuran *file* , artinya semakin besar ukuran *file* yang diproses maka semakin lama proses berlangsung.

5.2. Saran

Dengan adanya kesimpulan diatas, ada beberapa saran yang dapat dikemukakan sebagai bahan pertimbangan lebih lanjut guna meningkatkan produktifitas kerja dari aplikasi kompresi ini.

1. Melakukan kompresi dengan *file* lain misalnya seperti *video*, gambar, dan *text*.

2. Penelitian ini dapat di kembangkan lagi dengan memperbandingan metode *Huffman* dengan metode lainnya seperti *Shannon-fano, RLE, Half-Byte, LZW*.
3. Diharapakan untuk penelitian kedepannya hasil kompresi dapat langsung dimainkan tanpa melakukan proses dekompresi lagi.
4. Penelitian ini akan lebih baik lagi apabila menggunakan algoritma *Rice coding* sebagai metode algoritmanya.

DAFTAR PUSTAKA

Aburas, A.A., Rehiel,S.A (2008). *Fingerprint patterns recognition system using Huffman coding. Proceedings of the World Congress on Engineering*, London: WCE,Vol III.

Andysah P.,& Utama.S. 2016. *Implementasi teknik kompresi teks Huffman*. Fakultas Ilmu Komputer Universitas Pembanguan Panca Budi. Jurnal Informatika Vol.10, No.2.

Binanto. 2010. *Multimedia Digital Dasar Teori +Pengembangan*. Penerbit Andi. Yogyakarta.

David Huffman. 1952 . *The Huffman Algorithm*.

Dedi, D., Rizky P., & Nurul.H. 2018. *Kombinasi gifshuffle, enkripsi AES dan kompresi data huffman untuk meningkatkan keamanan data*. Universitas Lampung. Jurnal Teknologi Informasi dan Ilmu Komputer (JTIIK).Vol.05, No.4, Hlm, 389-394.

Deswintiani Sihotang.2017. *Perancangan aplikasi keamanan data text dengan metode idea dan kompresi menggunakan algoritma Huffman*.Teknik Informatika STMIK Budidarma Medan. Majalah Ilmiah INTI Vol. XII, No.1.

Devie R. Suchendra, Sandra W. 2012. *Implementasi kompresi data text menggunakan Huffman coding* .Program Studi Teknik Informatika STMIK LPKIA. JURNAL LPKIA, Vol.1 No.1.

Howe, D. (1993). Free on-line dictionary of computing. Diakses dari http://www.foldoc.org.

Karmela, S.M.W.,Willy, S.R.,& Antonius,R.C. 2010. *Aplikasi Player untuk Menjalankan File Wave yang terkompresi dengan Metode Huffman*. Fakultas Teknik Universitas Kristen Duta Wacana. Jurnal Informatika, Vol.6, No.1.

Kharisma, M., & Karpen.2017. *Rancang bangun aplikasi kompresi dan dekompresi pada citra digital menggunakan metode Huffman*. STMIK Amik Riau. PROCESSOR Vol.12, No.1.

Kurniawan, Erick. 2010. *Cepat Mahir Visual Basic 2010.* Yogyakarta: Andi.

Ramadhani C. 2015. *Dasar Algoritma & Struktur Data dengan Bahasa JAVA, 1e*. Yogyakarta : Andi Offset.

Rolly Yesputra, 2017. *Belajar visual basic. Net Dengan visual studio 2010*. Penerbit Royal Asahan Press. Sumatra Utara.

Salomon, D. (2004). *Data compression complete reference 3rd Edition*. Springer, NewYork.

Salomon. 2007. *The Compression Algorithm* ,Data Compression reference Center.

Sayood, K. 2005. *Introduction to data compression. Morgan Kaufmann,* San Fransisco.

Victor Amrizal.2010. *Implementasi Algoritma Kompresi Data Huffman Untuk Memperkecil Ukuran File MP3 Player*. Universitas Islam Negeri Syarif Hidayatullah Jakarta. Jurnal Sistem Informasi, 3(2), 2010, 1 – 14.

LISTING PROGRAM

A. *Source Code* Kompresi.vb

```
Public Class Kompresi
    Dim FileSizeAwal As Long
    Dim FileSizeAkhir As Long
    Dim RANDOM As System.Random = New System.Random()
    Dim Kamus(256) As String


    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click
        Dim OpenFD As OpenFileDialog = New OpenFileDialog()
        OpenFD.Filter = "All Files (*.*)|*.*"
        If OpenFD.ShowDialog() = DialogResult.OK Then
            TB_BF.Text = OpenFD.FileName
            Dim infoReader As System.IO.FileInfo
            infoReader =
            My.Computer.FileSystem.GetFileInfo(OpenFD.FileName)
            FileSizeAwal = infoReader.Length
        End If
    End Sub

    Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles B_Kompres.Click
        PB_kompresi.Value = 0
        Dim Node(256) As Nodenya
        Dim Temp As Nodenya
        Dim Indeks As Integer
        Dim Indeks2 As Integer
        Dim Isi As String
        Indeks = 256
        Dim FromTime As Date = DateTime.Now
        For i = 0 To 255
            Node(i) = New Nodenya(i, 0)
        Next
        Using file As New IO.FileStream(TB_BF.Text, IO.FileMode.Open)
            Dim nilai As Integer = file.ReadByte()
            Do Until nilai = -1
                Node(nilai).Jumlah = Node(nilai).Jumlah + 1
                nilai = file.ReadByte()
            Loop
        End Using
        For i = 0 To Indeks - 1
            For j = 0 To Indeks - 2
```

```
                If Node(j).Jumlah > Node(j + 1).Jumlah Then
                    Temp = Node(j)
                    Node(j) = Node(j + 1)
                    Node(j + 1) = Temp
                    Temp = New Nodenya(0, 0)
                End If
            Next
        Next

        Indeks2 = -1
        Do While Node(Indeks2 + 1).Jumlah = 0
            Indeks2 = Indeks2 + 1
        Loop

        Do
            Indeks = Indeks - 1
            Temp.Kiri = Node(Indeks2 + 1)
            Temp.Kanan = Node(Indeks2 + 2)
            Node(Indeks2 + 1) = Temp
            Node(Indeks2 + 2) = Nothing
            Temp = New Nodenya(0, 0)

            For i = Indeks2 + 2 To Indeks - 1
                Node(i) = Node(i + 1)
                Node(i + 1) = Nothing
            Next

            For i = Indeks2 + 1 To Indeks - 1
                For j = Indeks2 + 1 To Indeks - 2
                    If Node(j).Jumlah >= Node(j + 1).Jumlah Then
                        Temp = Node(j)
                        Node(j) = Node(j + 1)
                        Node(j + 1) = Temp
                        Temp = New Nodenya(0, 0)
                    End If
                Next
            Next
        Loop Until Indeks = Indeks2 + 2

        GeneratePola(Node(Indeks2 + 1), "")

        Dim jumlah As Integer
        Isi = ""
        Using file As New IO.FileStream(TB_BF.Text, IO.FileMode.Open)
            Dim nilai As Integer = file.ReadByte()
            Using filesave As New IO.FileStream(TB_BF.Text + ".huf",
IO.FileMode.Create)
                Do Until nilai = -1
                    Isi = Isi + Kamus(nilai)
                    Do While (Isi.Length > 8)
                        filesave.WriteByte(Bin_To_Dec(Isi.Substring(0, 8)))
                        Isi = Isi.Substring(8, Isi.Length - 8)
                    Loop
```

```
                    nilai = file.ReadByte()
                    jumlah = jumlah + 1
                    PB_kompresi.Value = (jumlah / FileSizeAwal) * 100
                Loop
                Dim Flag As Integer
                Flag = 8 - Isi.Length
                If Flag <> 8 Then
                    For i = 0 To Flag - 1
                        Isi = Isi + "0"
                    Next
                    filesave.WriteByte(Bin_To_Dec(Isi))
                End If
                filesave.WriteByte(Flag)
            End Using
        End Using
        Dim librarinya As String
        For i = 0 To 255
            librarinya = librarinya + i.ToString + ";" + Kamus(i) + "." +
vbNewLine
        Next
        My.Computer.FileSystem.WriteAllText(TB_BF.Text + ".lib", librarinya,
False)
        My.Computer.FileSystem.WriteAllText(TB_BF.Text + ".lib", librarinya,
False)
        Dim ToTime As Date = DateTime.Now
        Dim ts As TimeSpan = ToTime.Subtract(FromTime)
        Dim DifferenceInMilliseconds As Double = ts.TotalMilliseconds
        TB_LP.Text = DifferenceInMilliseconds.ToString & " ms"
        Dim infoReader As System.IO.FileInfo
        infoReader =
        My.Computer.FileSystem.GetFileInfo(TB_BF.Text + ".huf")
        FileSizeAkhir = infoReader.Length
        Dim redu As Long
        Dim rasio As Long
        redu = 100 - (FileSizeAkhir / FileSizeAwal) * 100
        rasio = (FileSizeAkhir / FileSizeAwal) * 100
        TB_redu.Text = redu & "%"
        TB_rasio.Text = rasio & "%"
        TB_akhir.Text = Math.Round((FileSizeAkhir / 1048576), 2) & " MB"
        TB_awal.Text = Math.Round((FileSizeAwal / 1048576), 2) & " MB"
        MessageBox.Show("Proses Kompresi Selesai", "Info")

    End Sub
    Private Sub GeneratePola(ByVal Akar As nodenya, ByVal Kode As String)
        If Akar.Kiri IsNot Nothing Then GeneratePola(Akar.Kiri, Kode + "0")
        If Akar.Kanan IsNot Nothing Then GeneratePola(Akar.Kanan, Kode + "1")
        If (Akar.Kanan Is Nothing) And (Akar.Kiri Is Nothing) Then
            Kamus(Akar.Data) = Kode
        End If
    End Sub
    Public Function Bin_To_Dec(ByVal Bin As String)
        Dim dec As Double = Nothing
        Dim length As Integer = Len(Bin)
```

```
        Dim temp As Integer = Nothing
        Dim x As Integer = Nothing
        For x = 1 To length
            temp = Val(Mid(Bin, length, 1))
            length = length - 1
            If temp <> "0" Then
                dec += (2 ^ (x - 1))
            End If
        Next
        Return dec
    End Function

    Private Sub Button3_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs)
        Me.Close()
    End Sub

    Private Sub DekompresiToolStripMenuItem_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
DekompresiToolStripMenuItem.Click
        Dekompresi.Show()
    End Sub


    Private Sub AboutToolStripMenuItem_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles AboutToolStripMenuItem.Click
        About.Show()
    End Sub
End Class
```

B. *Source Code* Dekompresi.vb

```
Public Class Dekompresi
    Dim FileSizeAwal As Long
    Dim RANDOM As System.Random = New System.Random()

    Private Sub B_BF_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles B_BF.Click
        Dim OpenFD1 As OpenFileDialog = New OpenFileDialog()
        Dim OpenFD2 As OpenFileDialog = New OpenFileDialog()
        OpenFD1.Filter = "Huffman File (*.huf*)|*.huf*"
        OpenFD2.Filter = "library (*.lib)|*.lib"
        If OpenFD1.ShowDialog() = DialogResult.OK Then
            TB_BF.Text = OpenFD1.FileName
            Dim infoReader As System.IO.FileInfo
            infoReader =
            My.Computer.FileSystem.GetFileInfo(OpenFD1.FileName)
            FileSizeAwal = infoReader.Length
            If OpenFD2.ShowDialog() = DialogResult.OK Then
                TB_FA.Text = OpenFD2.FileName
            End If
```

```
        End If
    End Sub

    Private Sub B_dekompres_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles B_dekompres.Click
        Dim KamusKode(256) As String
        Dim KamusData(256) As String
        Dim Indekskamus As Integer
        Indekskamus = 0
        Dim reader = My.Computer.FileSystem.ReadAllText(TB_FA.Text)
        Dim Datanya As String() = reader.Split(New Char() {"."c})
        Dim Datanyanya As String
        Dim FromTime As Date = DateTime.Now
        For Each Datanyanya In Datanya
            If (Indekskamus <> Datanya.Length - 1) Then
                Dim Datanya2 As String() = Datanyanya.Split(New Char() {";"c})
                KamusData(Indekskamus) = Datanya2(0).ToString
                KamusKode(Indekskamus) = Datanya2(1).ToString
                Indekskamus = Indekskamus + 1
            End If
        Next
        Dim bytes As Byte() = IO.File.ReadAllBytes(TB_BF.Text)
        Dim hex As String() = Array.ConvertAll(bytes, Function(b) b.ToString)
        Dim akhir As Integer
        Dim l As String
        Dim cari As String
        Dim cari1 As String
        Dim Indekscari As Integer
        Dim Indekscari2 As Integer
        Dim jumlah As Integer
        cari = ""
        cari1 = ""
        akhir = Integer.Parse(hex(hex.Length - 1))
        jumlah = 1
        PB_dekompres.Value = 0
        Using filesave As New IO.FileStream(TB_BF.Text + "[Dekompresi].wav",
IO.FileMode.Create)
            For i = 0 To hex.LongLength - 3
                l = hex(i)
                Indekscari = 0
                Indekscari2 = -1
                cari = ToBinary(Integer.Parse(l))
                Do
                    cari1 = cari1 & cari(Indekscari)
                    Do
                        Indekscari2 = Indekscari2 + 1
                        If String.Equals(KamusKode(Indekscari2), cari1) Then

filesave.WriteByte(Integer.Parse(KamusData(Indekscari2)))
                            cari1 = ""
                            Indekscari2 = Datanya.Length - 2
                        End If
                    Loop Until Indekscari2 = Datanya.Length – 2
```

```
                    Indekscari2 = -1
                    Indekscari = Indekscari + 1
                Loop Until Indekscari = cari.Length
                jumlah = jumlah + 1
                PB_dekompres.Value = (jumlah / FileSizeAwal) * 100
            Next
            'hapus padding
            Indekscari = 0
            Indekscari2 = -1
            cari = ToBinary(Integer.Parse(hex(hex.LongLength - 2)))
            cari = cari.Substring(0, cari.Length - akhir)
            Do
                cari1 = cari1 & cari(Indekscari)
                Do
                    Indekscari2 = Indekscari2 + 1
                    If String.Equals(KamusKode(Indekscari2), cari1) Then

filesave.WriteByte(Integer.Parse(KamusData(Indekscari2)))
                        cari1 = ""
                        Indekscari2 = Datanya.Length - 2
                    End If
                Loop Until Indekscari2 = Datanya.Length - 2
                Indekscari2 = -1
                Indekscari = Indekscari + 1
            Loop Until Indekscari = cari.Length
        End Using
        Dim ToTime As Date = DateTime.Now
        Dim ts As TimeSpan = ToTime.Subtract(FromTime)
        Dim DifferenceInMilliseconds As Double = ts.TotalMilliseconds
        Dim FileSizeAkhir As Long
        Dim infoReader As System.IO.FileInfo
        infoReader =
        My.Computer.FileSystem.GetFileInfo(TB_BF.Text + "[Dekompresi].wav")
        FileSizeAkhir = infoReader.Length
        TB_akhir.Text = Math.Round((FileSizeAkhir / 1048576), 2) & " MB"
        TB_awal.Text = Math.Round((FileSizeAwal / 1048576), 2) & " MB"
        TB_LP.Text = DifferenceInMilliseconds.ToString & " ms"
        MessageBox.Show("Dekompresi Selesai", "Info")
    End Sub

    Private Function ToBinary(ByVal dec As Integer) As String
        Dim bin As Integer
        Dim output As String
        While dec <> 0
            If dec Mod 2 = 0 Then
                bin = 0
            Else
                bin = 1
            End If
            dec = dec \ 2
            output = Convert.ToString(bin) & output
        End While
        If output Is Nothing Then
```

```
            Return "00000000"
        Else
            For i = 1 To 8 - output.Length
                output = "0" + output
            Next
            Return output
        End If
    End Function

    Private Sub B_BFA_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles B_BFA.Click
        Dim OpenFD As OpenFileDialog = New OpenFileDialog()
        OpenFD.Filter = "All Files (*.*)|*.*"
        OpenFD.ShowDialog()
        Me.Audio.URL = OpenFD.FileName
    End Sub

    Private Sub KompresiToolStripMenuItem_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles KompresiToolStripMenuItem.Click
        Kompresi.Show()
    End Sub

    Private Sub Button3_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs)
        Me.Close()
    End Sub

    Private Sub AboutToolStripMenuItem_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles AboutToolStripMenuItem.Click
        About.Show()
    End Sub
End Class
```

C. *Source code* Nodenya.vb

```
Imports Microsoft.VisualBasic

Public Class Nodenya
    Private _Kiri As Nodenya = Nothing
    Private _Kanan As Nodenya = Nothing
    Private _Data As Integer = 0
    Private _Jumlah As Integer = 0
    Public Property Kiri As Nodenya
        Get
            Return _Kiri
        End Get
        Set(ByVal value As Nodenya)
            _Kiri = value
        End Set
    End Property
    Public Property Kanan As Nodenya
```

```
        Get
            Return _Kanan
        End Get
        Set(ByVal value As Nodenya)
            _Kanan = value
        End Set
    End Property
    Public Property Data As Integer
        Get
            Return _Data
        End Get
        Set(ByVal value As Integer)
            _Data = value
        End Set
    End Property
    Public Property Jumlah As Integer
        Get
            Dim A As Integer
            A = _Jumlah
            If Not IsNothing(_Kiri) Then
                A = A + _Kiri.Jumlah
            End If
            If Not IsNothing(_Kanan) Then
                A = A + _Kanan.Jumlah
            End If
            Return A
        End Get
        Set(ByVal value As Integer)
            _Jumlah = value
        End Set
    End Property
    Public Sub New(ByVal datanya As Integer, ByVal jumlahnya As Integer)
        _Kiri = Nothing
        _Kanan = Nothing
        _Data = datanya
        _Jumlah = jumlahnya
    End Sub
End Class
```