PENERAPAN ALGORITMA STOUT CODE UNTUK MENGKOMPRESI FILE VIDEO BERFORMAT MP4

Piter Saputra Hondro[1], Imam Saputra[2], Putri Ramadhani[3]
[1,2,3]Universitas Budi Darma, Medan
[1]efendirahim0301@gmail.com, [2]saputraimam69@gmail.com,
[3]pramadhani@gmail.com

ABSTRAK

Saat ini, banyak file video dengan ekstensi MP4 beredar luas dan sering kali memiliki ukuran yang cukup besar. Untuk mengatasi masalah penyimpanan, kompresi diperlukan untuk mengurangi ukuran file. Tujuan dari penelitian ini adalah untuk mengimplementasikan algoritma Stout Codes dalam kompresi file video MP4, menganalisis tingkat kompresi yang dihasilkan oleh algoritma tersebut, serta mengevaluasi kinerjanya dalam hal rasio kompresi dan kualitas video setelah kompresi. Penelitian ini bertujuan untuk mengeksplorasi efektivitas Stout Codes dalam mengurangi ukuran file MP4 sambil mempertahankan kualitas video yang optimal, sehingga dapat memberikan wawasan mengenai sejauh mana algoritma ini dapat diterapkan dalam pengelolaan dan penyimpanan video. Kompresi data adalah proses yang mengubah data menjadi kode yang lebih efisien, menghemat ruang penyimpanan. Berbagai algoritma digunakan dalam proses kompresi, salah satunya adalah algoritma Stout Codes, yang relatif jarang digunakan. Algoritma Stout Codes bekerja dengan mengkodekan blok-blok video menjadi bentuk biner yang lebih kecil tanpa kehilangan informasi asli. Penerapan algoritma ini pada file video MP4 menghasilkan Compression Ratio sebesar 5,62%, yang berarti ukuran file sebelum kompresi, yaitu 128 bit, dapat dikurangi menjadi 72 bit setelah kompresi. Dengan demikian, algoritma Stout Codes menawarkan solusi efektif untuk mengurangi ukuran file video sambil mempertahankan integritas data, meskipun penggunaannya tidak sebanyak algoritma kompresi lainnya.

Kata Kunci : kompresi, MP4, stout code

ABSTRACT

Currently, numerous MP4 video files are widely distributed and often have relatively large sizes. To address storage issues, compression is needed to reduce file sizes. The aim of this research is to implement the Stout Codes algorithm for compressing MP4 video files, analyze the compression level achieved by this algorithm, and evaluate its performance in terms of compression ratio and video quality after compression. This study seeks to explore the effectiveness of Stout Codes in reducing MP4 file sizes while maintaining optimal video quality, providing insights into how this algorithm can be applied in video management and storage. Data compression is the process of transforming data into a more efficient code to save storage space. Various algorithms are used in the compression process, one of which is the Stout Codes algorithm, which is relatively uncommon. Stout Codes work by encoding video blocks into smaller binary forms without losing original information. Applying this algorithm to MP4 video files results in a Compression Ratio of 5.62%, meaning the file size before compression, which is 128 bits, can be reduced to 72 bits after compression. Thus, the Stout Codes algorithm offers an effective solution for reducing video file sizes while preserving data integrity, despite its less frequent use compared to other compression algorithms.

Keywords: Compression, MP4, Stout Code

A. Pendahuluan

Kompresi data merupakan teknik penting dalam pengolahan komputer yang bertujuan untuk mengurangi ukuran data agar lebih kecil dari versi aslinya. Proses ini menghemat ruang penyimpanan dengan mengurangi jumlah bit yang diperlukan untuk merepresentasikan data, yang pada gilirannya mengoptimalkan kapasitas penyimpanan sistem (Tarigan, 2020);(Tanjung, 2021). Selain manfaat ruang penyimpanan, kompresi data juga berperan dalam mempercepat proses pertukaran data, karena data yang lebih kecil memerlukan waktu yang lebih singkat untuk dikirim dan diterima (Haryanto, 2021). Dalam konteks teknologi modern, kompresi data menjadi semakin krusial seiring dengan pertumbuhan eksponensial volume data yang dihasilkan dan disimpan (Iskandar, 2023);(Andriyani, 2024). Seiring dengan meningkatnya kebutuhan akan penyimpanan yang efisien dan cepat, teknik kompresi data yang efektif memungkinkan pengelolaan dan pemrosesan data yang lebih baik, mendukung kemajuan teknologi dan inovasi di berbagai bidang, dari aplikasi berbasis web hingga sistem penyimpanan besar (Iskandar, 2024).

Pada penelitian ini membahas algoritma *Stout Codes* untuk kompresi *file* video berformat MP4. Algoritma *Stout Codes* merupakan jenis teknik kompresi *lossless* (Yus, 2022);(Aftikasyah, 2022). Dengan menerapkan teknik kompresi *lossless*, hasil data video yang dikompres dapat dikembalikan lagi ke data aslinya tanpa kehilangan informasi. Dengan menerapkan algoritma *Stout Codes*, peneliti ingin mengetahui kinerja kompresi file video MP4. Sehingga *file* video MP4 yang berukuran besar dapat dikompresi menjadi berukuran lebih kecil. Dengan begitu, proses transmisi data video dapat berjalan lebih cepat dan hemat ruang penyimpanan (Pradana & Saputra, 2019).

Penelitian oleh Utomo, (2020) menunjukkan bahwa algoritma *Stout Codes* efektif dalam kompresi data teks. Penelitian ini menegaskan bahwa metode *algoritma Stout Codes* cocok untuk kompresi teks, tetapi hasilnya juga dipengaruhi oleh jumlah data yang diproses. Hal ini memberikan gambaran bahwa *Stout Codes* dapat menjadi metode yang

efisien dalam konteks tertentu. Masalah umum terkait file besar, terutama file video, adalah waktu pengiriman yang lama dan kebutuhan ruang penyimpanan yang besar. Kompresi adalah salah satu teknik yang digunakan untuk mengatasi masalah ini (Ramayani, 2021). Penelitian yang menerapkan algoritma *Stout Codes* untuk kompresi file video MP4 membandingkan efektivitasnya dengan algoritma lain seperti *Huffman coding*, *Golomb coding, Elias Delta Code, dan CABAC*. Hasil menunjukkan bahwa *Stout Codes* unggul dalam hal rasio kompresi dan kecepatan kompresi untuk video resolusi tinggi, meskipun penelitian lebih lanjut masih diperlukan untuk menguji performanya pada berbagai dataset dan parameter kompresi.

Stout Codes berbeda dari algoritma kompresi lainnya, seperti *Huffman coding* dan *Golomb coding*, karena menggunakan pendekatan dinamis (Mesran, 2021);(Tamala & Sitanggang, 2022). *Algoritma* ini membagi data video menjadi blok-blok kecil dan mengkodekan setiap blok berdasarkan pola statistiknya. Teknik adaptif ini memungkinkan *Stout Codes* untuk mencapai tingkat kompresi yang optimal untuk berbagai jenis file video dengan resolusi berbeda, memberikan keuntungan dalam fleksibilitas kompresi.

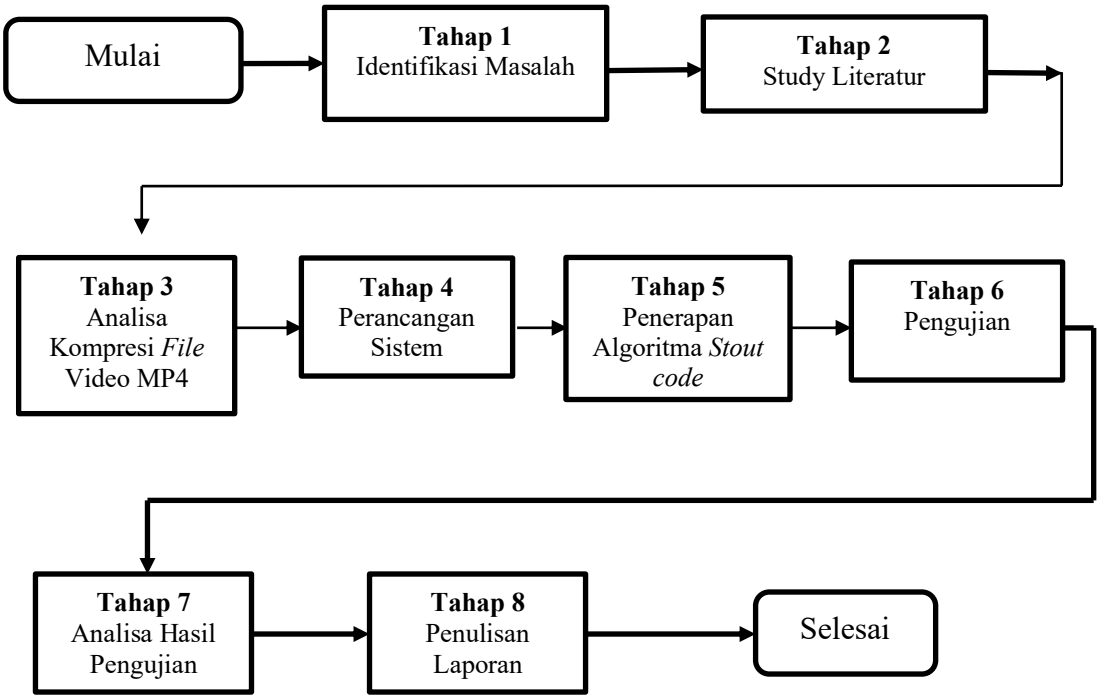
Dalam penelitian oleh Yus, (2022) yang berjudul "Perbandingan A*lgoritma Stout Code dan Algoritma Elias Delta Code* Dalam Kompresi File Video", ditemukan *bahwa algoritma Stout Codes* memiliki rasio kompresi yang lebih baik dibandingkan dengan *Elias Delta Code*. Hasil percobaan menunjukkan rasio kompresi 2,17 untuk *Stout Codes*, sementara Elias Delta Code menghasilkan rasio 1,85, menegaskan keunggulan *Stout Codes* dalam hal efektivitas kompresi. Penelitian lainnya yang dilakukan oleh Rizky Syahputra dengan judul penelitian "Kompresi File Video MP4 dengan Menggunakan Metode *Discrete Cosine Transform*" mengatakan bahwa kompresi video diperlukan untuk memperkecil ukuran data sehingga akan lebih efisien dalam penyimpanan data (Syahputra, 2016).

Berdasarkan permasalahan di atas, maka penulis ingin memaparkan langkah-langkah dan usulan solusi yang dipaparkan dalam

penelitian ini melalui judul. "Penerapan *Algoritma Stout Codes* Untuk Mengkompresi File Vidio Berformat MP4.

B. Metode Penelitian

Metodologi penelitian mencakup tahapan-tahapan yang dilakukan secara sistematis untuk memudahkan proses penelitian. Sebelum menyusun kerangka penelitian, penulis terlebih dahulu menganalisis topik yang akan diteliti, memastikan bahwa setiap langkah dalam metodologi tersebut terkait dengan tujuan penelitian dan mendukung kelancaran proses penelitian secara keseluruhan.



Gambar 1. *Flowchart* Tahapan Penelitian

Waktu Pelaksanaan Penelitian .

Adapun waktu pelaksaan penelitian akan disajikan dalam bentuk tabel di bawah ini

Tabel 1. Waktu Pelaksanaan Penelitian

No	Tahapan	Februari				Maret				April				Mei				Juni			
		1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
1	Identifikasi Masalah			✓	✓																

2	Study Literatur				✓	✓														
3	Ananlisa Kompresi file Video MP4					✓	✓	✓												
4	Analisa Dekompresi file Video MP4						✓	✓												
5	Perancangan Sistem								✓	✓	✓									
6	Penerapan Algoritma									✓	✓	✓								
7	Pengujian											✓	✓	✓						
8	Analisa Hasil Pengujian													✓	✓	✓	✓			
9	Penulisan Laporan																✓	✓	✓	✓	✓

C. Hasil Penelitian dan Pembahasan

Penerapan Algoritma Stout Code

Analisis menunjukkan bahwa file MP4 sering kali memiliki ukuran yang sangat besar, sehingga memerlukan proses kompresi untuk mengurangi ukuran file dan menghemat ruang penyimpanan. Proses kompresi ini bertujuan untuk mengecilkan ukuran file MP4 tanpa kehilangan kualitas yang signifikan, dengan algoritma Stout Codes sebagai metode utama yang dipilih untuk mencapai hal tersebut. Dalam penelitian ini, akan dibahas dua proses utama: kompresi dan dekompresi file MP4 menggunakan algoritma Stout Codes.

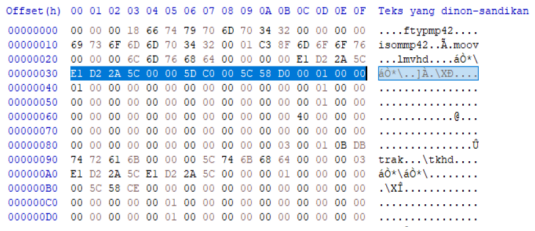
Sebagai langkah awal sebelum kompresi, file MP4 dibaca untuk mendapatkan nilai heksadesimal menggunakan aplikasi HxD. Proses ini penting untuk memulai kompresi dan dekompresi, di mana contoh file MP4 akan digunakan untuk menerapkan teknik tersebut. Dengan cara ini, algoritma Stout Codes dapat diterapkan untuk mengoptimalkan ukuran file sambil mempertahankan kualitas video.

Analisa proses kompresi *file* MP4 dengan menggunakan algoritma *stout codes*

a. Memasukkan *File*

Tabel 2. Sampel *File* MP4 yang akan dikompresi

Nama File	Lagu
Extension File	MP4
Nilai Heksadesimal	E1 D2 2A 5C 00 00 5D C0 00 5C 58 D0 00 01 00 00



Gambar 2 Isi *file* MP4 dari Hasil HxD

b. Melakukan Pembacaan Isi *File*

Nilai heksadesimal dari file MP4 ini adalah E1 D2 2A 5C 00 00 5D C0 00 5C 58 D0 00 01 00 00. Nilai-nilai ini dimasukkan ke dalam tabel untuk menghitung frekuensi kemunculan setiap nilai. Proses pembacaan frekuensi dilakukan dengan menghitung jumlah kemunculan masing-masing nilai yang ada. Hasil pembacaan frekuensi dapat dilihat di bawah ini:

Tabel 3 Nilai *File*

Nilai	Frekuensi	Char
E1	1	á
D2	1	Ò
2A	1	*
5C	2	\
00	6	
5D	1	]
C0	1	À
58	1	X
D0	1	Đ
01	1	

c. Dilakukan pengurutan karakter, urutan nilai dapat dilihat dalam tabel di bawah:

Tabel 4. Nilai *Bit* MP4 Sample

Nilai		Bit	Frek	Bit x Frek
Hexa	Biner			
00	00000000	8	6	48
5C	01011100	8	2	16
E1	11100001	8	1	8
D2	11010010	8	1	8
2A	00101010	8	1	8
5D	01011101	8	1	8
C0	11000000	8	1	8
58	01011000	8	1	8
D0	11010000	8	1	8
01	00000001	8	1	8
Total Bit				128

.

d. Membentuk Tabel *Stout Codes*

Adapun kode *Stout Codes* dapat dilihat pada tabel di bawah ini.

Tabel 5 Kode *Stout Codes*

N	Kode *Stout Codes*
1	01
2	10
3	11
4	00100
5	00101
6	00110
7	00111
8	011000

9	011001
10	011010
11	011011
12	011100

Sumber: David Salomon dan Giovani Motta, 2010[8]

Adapun proses kompresi *file* MP4 sample dapat di lihat pada tabel berikut :

Tabel 6 Kompresi Nilai *File* MP4 Sampel dengan *Stout Codes*

N	Nilai Hexa	Kode *Stout Codes*	Bit	Frek	Bit x Frek
1	00	01	2	6	12
2	5C	10	2	2	4
3	E1	11	2	1	2
4	D2	00100	5	1	5
5	2A	00101	5	1	5
6	5D	00110	5	1	5
7	C0	00111	5	1	5
8	58	011000	6	1	6
9	D0	011001	6	1	6
10	01	011010	6	1	6
Total Bit					56

e. Melakukan hasil *string bit Stout Codes*

Sebelum menghasilkan nilai file dari string bit yang dikompresi dengan Stout Codes, pertama-tama dilakukan pemeriksaan panjang string bit. Nilai bit baru hasil kompresi dihasilkan dari susunan nilai heksadesimal sebelum kompresi, yaitu E1 D2 2A 5C 00 00 5D C0 00 5C 58 D0 00 01 00 00 (tanpa koma dan spasi), yang diubah menjadi bentuk bit biner: “11 00100 00101 10 01 01 00110 00111 01 10 011000 011001 01 011010 01 01”.

Padding	Flagbit
7 – 0 + “1”	Bit Akhir 9 – n
7 – 0 + “1” = **00000001**	Bit Akhir = 9 – 0 = 9 = **00001001**

Gambar 3 Perhitungan Penambahan bit

11001000 01011001 01001100
01110110 01100001 10010101
10100101 **00000001** **00001001**

Gambar 3 *String bit* yang telah dilakukan penambahan

11001000 01011001 01001100
01110110 01100001 10010101
10100101 **00000001** **00001001**

Gambar 4. Pembagian *String bit*

Berdasarkan pembagian kelompok nilai biner, diperoleh 9 kelompok nilai biner baru (9 byte) hasil kompresi dari nilai biner yang

telah ditambahkan bit. Setelah pembagian, nilai-nilai biner tersebut diubah menjadi karakter dengan terlebih dahulu mencari nilai heksadesimal dan string bit menggunakan kode ASCII. Proses ini memungkinkan transformasi nilai biner menjadi bentuk yang lebih dapat dipahami dan dianalisis. Nilai yang telah terkompresi hasil dari proses ini dapat dilihat dalam tabel di bawah ini.

Tabel 7. Nilai *Hexadecimal* Terkompresi

Urutan Nilai	Nilai	Char
Terkompresi	*Hexadecimal* Terkompresi	
1	C8	È
2	59	Y
3	4C	.
4	76	v
5	61	C
6	95	•
7	A5	¥
8	1	
9	9	

Analisa proses dekompresi *file* MP4 menggunakan Algoritma *Stout Codes*

Hasil *string bit Stout Codes* yang dijadikan nilai *file* diubah ke dalam bentuk biner. *Bit* keseluruhan hasil kompresi dapat dilihat pada tabel berikut :

Tabel 4.7 Nilai *Hexadecimal* dan Biner Kompresi

Urutan Nilai Terkompresi	Nilai *Hexadecimal*	Char	Nilai Biner
1	C8	È	11001000
2	59	Y	01011001
3	4C	.	01001100
4	76	v	01110110
5	61	C	01100001
6	95	•	10010101
7	A5	¥	10100101
8	1		00000001
9	9		00001001

Berdasarkan tabel di atas maka diambil seluruh nilai biner dan digabungkan menjadi 11001000 01011001 01001100 01110110 01100001 10010101 10100101 00000001 00001001

Untuk mengembalikan binary menjadi string bit awal, langkah pertama adalah menghilangkan bit yang ditebalkan dari hasil kompresi. Proses ini dimulai dengan membaca 8 bit terakhir dari data biner, kemudian mengubahnya menjadi bilangan desimal dan menyebut

hasilnya sebagai \(n \). Bit-bit di bagian akhir yang harus dihilangkan adalah sebanyak \(7 + n \). Setelah perhitungan ini, bit-bit akhir yang terhapus—sebanyak 8 bit jika \(n=1 \)—dihilangkan, dan pembacaan dilakukan pada bit-bit yang tersisa. Hasil akhir dari proses ini mengembalikan string bit ke bentuk semula sebagai 11001000 01011001 01001100 01110110 01100001 10010101 10100101. Dengan menggunakan algoritma Stout Codes, string bit ini, yang berjumlah 64 bit, menunjukkan bahwa hasil akhir sesuai dengan string bit awal sebelum kompresi. Tabel hasil perhitungan yang mendukung proses ini disediakan di bawah:

Tabel 7 Nilai *file* dan *Stout Codes*

N	Nilai *Stout Codes*	Nilai *File*	Char
1	11	E1	á
2	00100	D2	Ò
3	00101	2A	*
4	10	5C	\
5	01	00	
6	01	00	
7	00110	5D	]
8	00111	C0	À
9	01	00	
10	10	5C	\
11	011000	58	X
12	011001	D0	Đ
13	01	00	
14	011010	01	
15	01	00	
16	01	00	

Dari hasil dekompresi di atas, didapat nilai hexadesimal awal sebelum kompresi yaitu E1, D2, 2A, 5C, 00, 00, 5D, C0, 00, 5C, 58, D0, 00, 01, 00, 00.

Tabel 8 Perbedaan Nilai Sebelum Kompresi

Deskripsi	Sebelum Kompresi	Setelah Kompresi
Bit	128 bit	72 bit

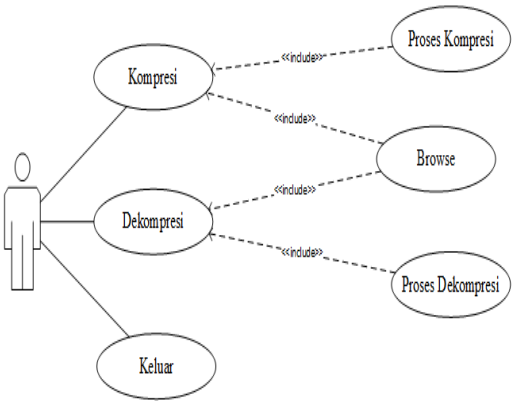
Pemodelan Sistem

Dalam penelitian ini penulis menggunakan beberapa pemodelan sistem untuk mendapatkan hasil yang jelas serta hal-hal yang harus dilakukan oleh sistem, yaitu :

1. Use Case Diagram

   Use case diagram adalah model fungsional yang menggambarkan interaksi antara pengguna dan sistem, menunjukkan bagaimana sistem digunakan melalui layanan dan fungsi yang disediakannya. Diagram ini menggambarkan peran pengguna, yang disebut aktor, dalam hubungannya dengan sistem. Diagram use case pada aplikasi yang dirancang dapat dilihat pada gambar di bawah ini, memberikan gambaran visual

tentang bagaimana berbagai fungsi sistem diakses dan digunakan oleh pengguna.



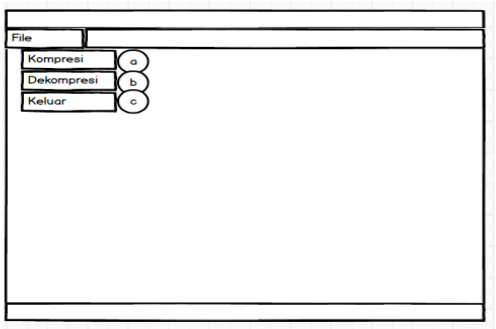
Gambar 5 *Use Case Diagram*

Perancangan Antarmuka

Perancangan antarmuka adalah penghubung antar media perangkat lunak kepada pengguna untuk melakukan proses kompresi dan proses dekompresi *file* MP4. *Form* kompresi dan dekompresi *file* MP4 berfungsi untuk mengkompresi file MP4 yang diinginkan dan mendekompresikan kembali *file* MP4 yang sudah tekompres, gambar form kompresi dan dekompresi dapat dilihat pada gambar di bawah ini :

1. Rancangan *Form* Utama

Form utama adalah halaman yang pertama kali mucul ketika aplikasi dibuka. Di dalam halaman ini terdapat menu strip kompresi dan dekompresi. Adapun gambar rancangan form utama dapat dilihat di bawah ini.



Gambar 6 Tampilan Awal Form Kompresi File MP4

Keterangan :

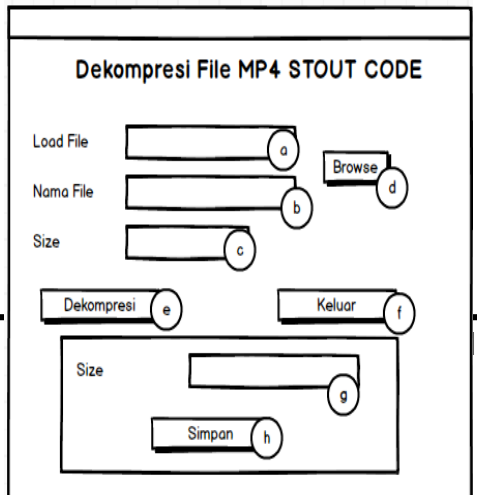
a. Menu strip kompresi berfungsi untuk masuk ke halaman kompresi.

b. Menu strip dekompresi berfungsi untuk masuk ke halaman dekompresi.

c. Menu stip keluar berfungsi untuk keluar dari form menu utama.

2. Rancangan *Form* Dekompresi

Rancangan *form* dekompresi adalah halaman yang digunakan untuk melakukan proses dekompresi *file* yang telah dikompresi. Adapun tampilan rancangan *form* dekompresi dapat dilihat pada tabel di bawah ini.



Gambar 7. Tampilan Form Dekompresi

Implementasi Program

Implementasi program merupakan tahap uji coba dari sistem yang dibangun, pada bagian implementasi program membahas spesifikasi perangkat keras, perangkat lunak dan hasil dari tampilan sistem ketika sedang berjalan.

Kebutuhan Sistem

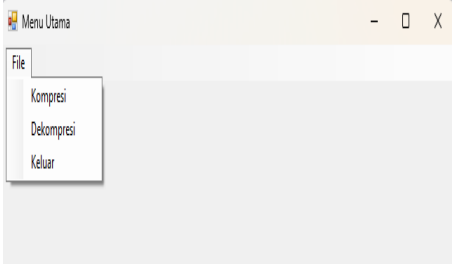
Kebutuhan sistem merinci perangkat lunak dan perangkat keras yang diperlukan untuk aplikasi, dengan fokus pada spesifikasi yang harus dipenuhi untuk memastikan implementasi sistem yang efektif. Implementasi sistem memerlukan perencanaan yang matang dan mengikuti prosedur tertentu untuk mencapai tujuan yang diinginkan. Agar sistem dapat berfungsi dengan baik, spesifikasi perangkat keras dan perangkat lunak yang tepat harus dipenuhi. Berikut adalah spesifikasi perangkat yang dibutuhkan: Perangkat Keras: Perangkat keras yang diperlukan meliputi komputer dengan spesifikasi sebagai berikut: Processor Core i5, RAM 8 GB, layar dengan resolusi 1920 x 1080 piksel, SSD 256 GB, dan mouse. Perangkat Lunak: Untuk perangkat lunak, sistem yang digunakan harus memenuhi spesifikasi minimal berikut: Sistem Operasi Windows 2011 dan perangkat lunak Microsoft Visual Studio 2008.

Tampilan *Input*

Tampilan *input* merupakan tampilan yang akan menjelaskan *form* aplikasi yang digunakan untuk kebutuhan program yang akan dibutuhkan.

1. *Form* Halaman Utama

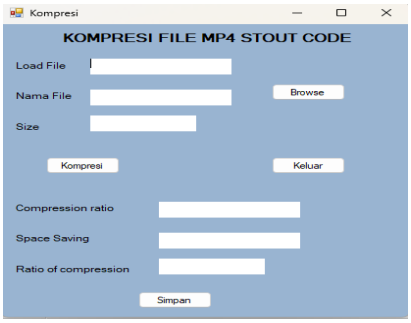
Form halaman utama adalah *form* yang pertama kali muncul ketika aplikasi kompresi file MP4 dibuka. Adapun gambar form halaman utama dapat dilihat pada gambar di bawah ini.



Gambar 8. Tampilan Form Menu Utama

2. Form Kompresi

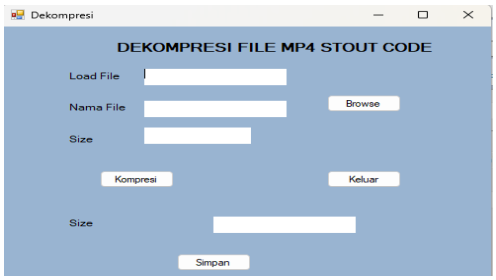
Form kompresi merupakan *form* yang digunakan untuk melakukan proses kompresi. Pada *form* ini disediakan *button* yang digunakan untuk memilih *file* MP4.



Gambar 9 Tampilan Form Kompresi

3. Form Dekompresi

Form dekompresi adalah form yang digunakan untuk melakukan proses dekompresi *file* MP4. Adapun halaman input form dekompresi dapat dilihat pada gambar di bawah ini.



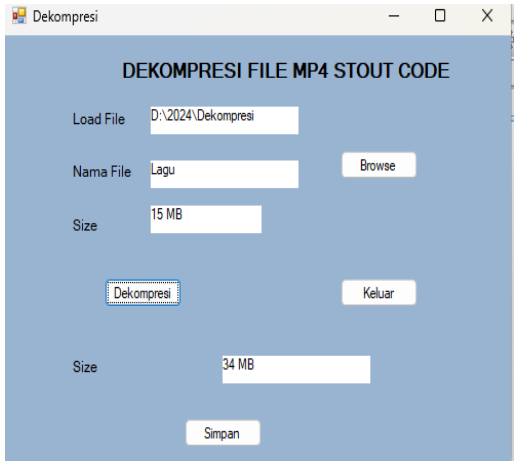
Gambar 9 Tampilan Form Dekompresi

Tampilan *Output*

Tampilan *output* merupakan tampilan proses dari pengembalian *file* MP4 yang telah dikompresi ke bentuk awal sebelum proses kompresi dilakukan. Adapun tampilan output antara lain.

1. Form Kompresi

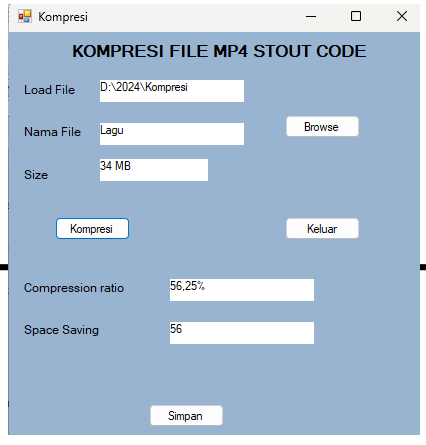
Tampilan output form kompresi menampilkan hasil dari proses kompresi dilakukan yaitu compression ratio, space saving dan compression of ratio. Adapun gambar form kompresi dapat dilihat di bawah ini.



Gambar 10. Tampilan Hasil Kompresi

2. Form Dekompresi

Halaman output form dekompresi halaman yang digunakan untuk melakukan proses dekompresi file yang telah dikompresi dengan algoritma stout code. Adapun gambar tampilan output form dekompresi dapat dilihat di bawah ini



Gambar 11 Hasil Dekompresi

D. Kesimpulan

Berdasarkan penelitian, dapat disimpulkan bahwa kompresi file MP4 dapat dilakukan dengan mengonversi file ke dalam bentuk heksadesimal dan kemudian memprosesnya menggunakan algoritma Stout Code untuk menghasilkan bilangan heksadesimal baru yang mengurangi ukuran file MP4. Algoritma Stout Code terbukti efektif dalam mengompresi file MP4 besar menjadi ukuran yang lebih kecil, dengan mencapai rasio kompresi sebesar 56,25%. Proses kompresi ini dirancang menggunakan Microsoft Visual Studio 2008, dan penerapan algoritma Stout Code memungkinkan aplikasi yang dihasilkan untuk secara efektif mengurangi ukuran file MP4 setelah kompresi.

DAFTAR PUSTAKA

Aftikasyah, N. (2022). Penerapan Algoritma Yamamoto ' s Recursive Code Untuk Mengkompresi File Dokumen. *KOMIK*, *5*(2), 255–263.

Andriyani. (2024). Data Sebagai Fondasi Kecerdasan Buatan. In *TOHAR MEDIA*.

David Salomon, G. M. (2010). *Handbook Of Data Compression*. Spinger.

Haryanto. (2021). Jaringan Komputer. In *Penerbit Andi.*

Iskandar. (2023). Peran Teknologi Dalam Dunia Pendidikan. In *Yayasan Cendekiawan Inovasi Digital Indonesia.*

Iskandar. (2024). Teknologi Big Data: Pengantar Dan Penerapan Teknologi Big Data Di Berbagai Bidang . In *PT. Green Pustaka Indonesia.*

Mesran. (2021). Peningkatan Keamanan Kriptografi Caesar Cipher dengan Menerapkan Algoritma Kompresi "Stout Codes." *JURNAL RESTI Peningkatan Keamanan Kriptografi Caesar Cipher Dengan Menerapkan*, *1*(10), 7–12.

Pradana, A. D., & Saputra, I. (2019). Penerapan Algoritma Interpolative Coding Pada Aplikasi Kompresi File Gambar. *KOMIK (Konferensi Nasional Teknologi Informasi Dan Komputer)*, *3*(1), 444–448. https://doi.org/10.30865/komik.v3i1.1592

Ramayani, K. (2021). Penerapan Algoritma Rice Codes Untuk Mengkompresi File Video. *Komputa : Jurnal Ilmiah Komputer Dan Informatika*, *5*(1), 186–192. https://doi.org/10.30865/komik.v5i1.3670

Syahputra, R. (2016). Kompresi File Video Mp4 Dengan Menggunakan Metode. *Jurnal Riset Komputer*, 52–57.

Tamala, R., & Sitanggang, A. (2022). Penerapan Algoritma Stout Codes Untuk Kompresi Databases Pada Pinjaman Online. *Bulletin of Information Technology (BIT)*, *3*(3), 189–194.

Tanjung, R. Y. (2021). Perancangan Aplikasi Kompresi File Dokumen Menggunakan Algoritma Adiitive Code. *JURIKOM (Jurnal Riset Komputer)*, *8*(4), 108–113.

Tarigan, B. (2020). Bulletin of Information Technology (BIT) Penerapan Algoritma VLBE Pada Aplikasi Kompresi File. *Bulletin of Information Technology (BIT)*, *1*(2), 75–82.

Utomo, D. P. (2020). Penerapan Algoritma Stout Codes Untuk Kompresi Record Pada Databade Di Aplikasi Kumpulan Novel. *KOMIK*, *4*(1), 311–314.

Yus, L. R. M. (2022). Perbandingan Algoritma Stout Code Dan Algoritma Elias Delta Code Dalam Kompresi File Video. *KOMIK (Konferensi Nasional Teknologi Informasi Dan Komputer)*, *6*(1), 175–181.