

Tribhuvan University



Faculty of Humanities and Social Sciences

File Compression and Decompression Using

Huffman Algorithm

A PROJECT REPORT

Submitted to

Department of Computer Application

Asian College of Higher Studies

In partial fulfilment of the requirements for the Bachelor in Computer Application

Submitted by

Bishal Rauniyar [210615]

2024/09/21

Under the supervision of

Basant Karki



Tribhuvan University

Faculty of Humanities and Social Sciences

Asian College of Higher Studies (ACHS)

Supervisor's Recommendation

I hereby recommend that this project prepared under my supervision by Bishal Rauniyar entitled “**File Compression and Decompression**” in partial fulfilment of the requirements for the degree of Bachelor of Computer Application be recommended for the Final-evaluation.

SIGNATURE

Basant Karki

SUPERVISOR

Asian College of Higher Studies (ACHS)

Ekantakuna, Jawalakhel



Tribhuvan University

Faculty of Humanities and Social Sciences

Asian College of Higher Studies (ACHS)

LETTER OF APPROVAL

This is to certify that this project prepared by **Bishal Rauniyar** entitled “**File Compression and Decompression**” in partial fulfilment of the requirements for the degree of Bachelor in Computer Application (BCA) has been evaluated. In our opinion, it is good and satisfactory in the scope and quality of a project for the required degree.

SIGNATURE OF SUPERVISOR

Basanta Karki

SIGNATURE of Internal Examiner

SIGNATURE of Coordinator

Pranaya Nakarmi

Acknowledgement

This project would not have been possible without the help and guidance of many individuals despite the huge efforts from our side.

We would like to extend our sincere thanks to all. We are indebted to the Asian College of Higher Studies (ACHS) and its members for their support and supervision and for providing the necessary information regarding this project. Also, we would like to thank our project guide Basanta Karki for his kind co-operation and guidance which has helped us in the completion of the project.

We give our special thanks and kind appreciation to our seniors, colleagues, and a few individuals for their help during our project.

Abstract

File compression is the process of reducing the size of a file or collection of files by encoding the data in a more compact representation while preserving its essential content. This can be achieved through lossless compression, which removes redundancies in the data without any loss of information, or lossy compression, which selectively discards nonessential details to achieve higher compression ratios. Decompression, the reverse process, restores the compressed file back to its original form, allowing it to be accessed or used. File compression and decompression are widely used for efficient storage, fast transmission, and improved system performance in various applications. File compression and decompression are fundamental techniques in the world of digital data management. They serve a multitude of critical purposes that go beyond just saving storage space.

Keywords: *Huffman Coding, Frequency Table, Priority Queue, Huffman Tree, Encoding, Decoding, Binary Encoding, Compression Ratio, Tree Traversal, Symbol, Frequency Analysis, Code Table*

List of Abbreviation

Acronym	Full form
CSS	Cascaded Style Sheet
CASE	Computer-Aided Software Engineering
DBMS	Database Management System
ER	Entity Relationship
HTML	Hypertext Markup Language
IT	Information Technology
JS	JavaScript
UML	Unified Modelling Language
UX	User Experience
DFD	Data Flow Diagram

Table of Contents

Acknowledgement	iii
Abstract	iv
Table of Contents	vi
Chapter 1: Introduction	1
1.1 Introduction	1
1.2 Problem Statement	3
1.3 Objectives	3
1.4 Scope and Limitations	3
1.5 Development Methodology:	4
1.6 Report Organization:	5
Chapter 2: Literature Review	6
2.1 Background Study:	6
2.2 Literature Review:	6
Chapter 3: System Analysis and Design	9
3.1 System Analysis	9
3.1.1 Requirement Analysis	9
3.1.2 Feasibility Study	10
3.1.3 Data Modeling (ER – Diagram)	12
3.1.4 Process Modeling (DFD)	13
3.2 System Design	14
3.2.1 Architectural Design	14

3.2.2 Database Schema Design.....	15
3.2.3 Interface Design.....	16
3.2.4 Physical DFD	18
3.3 Algorithm Used.....	19
Chapter 4: Implementation and Testing	23
4.1 Implementation	23
4.1.1 Tools used	23
4.1.2 Implementation Details of Modules	24
4.2 Testing.....	25
4.2.1 Test Case for Unit Testing	25
4.2.2 Test Case for System Testing	26
Chapter 5: Conclusion and Future Recommendations.....	27
5.1 Conclusion:	27
5.3 Future Recommendations	28
References.....	30

List of Figures

Figure 1.5: Waterfall Model	4
Figure 3.1.1: Use Case Diagram	9
Figure 3.1.3: ER-Diagram.....	12
Figure 3.1.4.1: Level 0 DFD	13
Figure 3.1.4.2: Level 1 DFD	13
Figure 3.2.1: Architectural Design	14
Figure 3.2.2: Database Schema.....	15
Figure 3.2.3.2: Login Form	16
Figure 3.2.3.1: Register Form	16
Figure 3.2.3.3: Txt Compression	17
Figure 3.2.3.4: Pdf Compression	17
Figure 3.2.3.5: History of Compression	18
Figure 3.2.4: Physical DFD	18
Figure 3.3: Huffman Tree	19

List of Tables

Table 4.2.1: System Testing	25
Table 4.2.2: System Testing	26

List of Abbreviation

Acronym	Full form
CSS	Cascaded Style Sheet
CASE	Computer-Aided Software Engineering
DBMS	Database Management System
ER	Entity Relationship
HTML	Hypertext Markup Language
IT	Information Technology
ROI	Return on Investment
UML	Unified Modelling Language
UX	User Experience

Chapter 1: Introduction

1.1 Introduction

Data compression is an essential aspect of today's digital world, helping us manage the vast amounts of information we generate and use daily. With the explosion of digital content, from documents and emails to images and videos, efficiently storing and transmitting data has become more important than ever. Compression techniques are crucial for reducing file sizes, which in turn saves storage space and speeds up data transfer. This is especially beneficial in environments with limited bandwidth or storage capacity.

One of the most well-known and widely used methods for lossless data compression is the Huffman algorithm. Invented by David A. Huffman in the early 1950s, this algorithm cleverly reduces the size of text files without losing any information. It works by analyzing the frequency of each character in a file and then assigning shorter codes to more common characters and longer codes to less common ones. This approach ensures that the overall file size is minimized, making it a practical solution for a variety of applications where data integrity is crucial. [1]

File compression and decompression are important techniques used in information technology to efficiently manage digital files. Compression reduces file sizes by using special algorithms to make the data more compact, taking advantage of patterns and repetitions in the data. This helps save space and makes it easier to store and send files. Decompression is the opposite process, where compressed files are brought back to their original form so that they can be used or shared. Compression allows organizations and individuals to make the most of their storage capacity, send files faster, and improve their overall computer performance. It is used for tasks like storing large amounts of data, sending files over the internet, or sharing multimedia files.

File compression is like a digital space-saving technique that works like magic for your files. Imagine you have a big collection of photos, documents, and other digital stuff. Compression helps make all of these files smaller by finding patterns and repetitions in them. It's like folding your clothes neatly and squeezing them into a smaller suitcase, so you can fit more in and carry it around easily. This space-saving ability is incredibly helpful in many ways. First, it lets you

store more files on your computer or devices without running out of space. You can also send files to your friends or colleagues faster because they're smaller and take less time to upload or download. Plus, it can even make your computer run smoother and faster because it doesn't have to work as hard with smaller files. So, whether you're organizing your digital life, sharing files, or just making your computer perform better, file compression is like a helpful magician that simplifies your digital world and makes everything more efficient. File compression serves multiple objectives in the realm of information technology. One of its primary purposes is to enhance speed and performance across various applications and systems. By reducing the size of files, it allows for quicker data access and manipulation, leading to improved overall efficiency. This is particularly crucial in scenarios where large datasets or extensive computational tasks are involved, such as in scientific research or complex data analysis. [6]

Another key objective is to enable real-time compression. In situations where data needs to be processed or transmitted in real-time, such as in video streaming or online gaming, compression algorithms play a vital role. They help reduce the amount of data that needs to be sent or processed instantaneously, ensuring smoother experiences for users and minimizing latency. Additionally, the development of improved lossless compression techniques for multimedia files is a crucial goal. Multimedia files, like images, audio, and video, often require specialized compression methods to reduce their size without compromising quality. Enhancements in lossless compression ensure that multimedia content can be stored, transmitted, and displayed with minimal loss of information, thereby delivering high-quality multimedia experiences. [2]

In summary, file compression serves diverse objectives, including enhancing speed and performance, enabling real-time applications, and continually advancing lossless compression techniques for multimedia content. These objectives collectively contribute to a more efficient and effective digital landscape, benefiting a wide range of industries and applications. This project aims to develop a system that can effectively compress and decompress text files using the Huffman algorithm. By implementing this technique, we aim to demonstrate its practical application in improving data storage and transmission, providing a reliable tool for managing large amounts of text data.

1.2 Problem Statement

With the increasing volume of text-based data, efficient storage and transmission have become critical. Traditional methods of storing text data often result in large file sizes, leading to increased storage costs and longer transmission times. This project seeks to address these issues by developing a solution that uses the Huffman algorithm to compress text files, significantly reducing their size without losing any information, and decompressing them back to their original form when needed. File compression and decompression face challenges that need improvement. One challenge is to optimize compression efficiency, making files smaller without losing data. Another challenge is the speed of compression and decompression, especially for large files. Compatibility and interoperability problems arise due to different file formats. Real-time applications require compression that works quickly without delays. Security is crucial, so encryption methods need integration. Lastly, multimedia compression should preserve quality. Solving these challenges will lead to better data management, faster file transfers, compatibility, security, and improved multimedia compression.

1.3 Objectives

The objectives of this study include:

- To Ensure decompressed files match the original files using Huffman Coding.
- To make files smaller to save space and transfer them faster.
- To uncompress files accurately to retrieve the original content.
- To optimize network usage and save money with compressed files.
- To store files efficiently for long-term archiving and storage.
- To keep data intact during compression and decompression processes.

1.4 Scope and Limitations

A) The scope of File Compression and Decompression includes:

- Designing algorithms to make files smaller while maintaining data integrity.
- Using widely-used compression formats and examining their effectiveness.
- Optimizing compression performance in terms of speed and efficiency.

- Exploring compression techniques for multimedia files, such as images, videos, and audio.
- Integrating compression methods into storage systems to maximize space utilization.

B) The limitation of File Compression and Decompression includes:

- Some data or quality loss with lossy compression.
- More computer work needed for compression and decompression.
- Not effective for already compressed or highly encrypted files.
- Risk of security issues if not properly encrypted or secured.
- Limited file size reduction for already highly compressed files.
- Longer processing times for large or complex files.

1.5 Development Methodology:

Since the project is an individual work and was designed utilizing the waterfall model, task management is made easier by the paradigm's simple, clear, and straightforward steps. The model's accuracy and the fact that each phase has clear deliverables and a review process provide it additional benefits for managing the application's development.

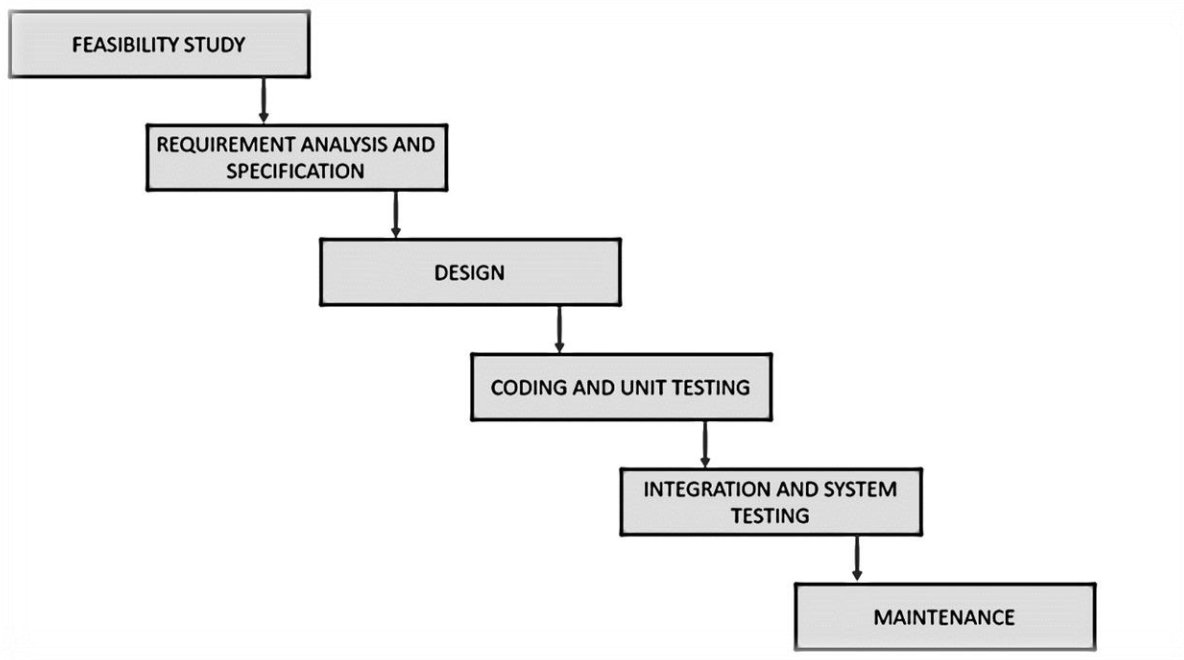


Figure 1.5: Waterfall Model

Therefore, the model was put into use since each phase is processed and finished separately, and most importantly, they do not overlap and it's easier to manage when changes are unlikely to occur during the development process. Additionally, all needs were met at the start of the project in accordance with the model.

1.6 Report Organization:

Chapter 1: Introduction covers the project's major elements: the Overview, Problem Statement, Objectives, Scope, and Limitations and development methodology. The project's operation is briefly clarified in the Overview. The project's problem is stated in the problem statement, which places it in a theoretical and research-focused framework. The chapter then goes on to explain the goals, factors, and limitations of the project.

Chapter 2: Background Study and Literature Review delves into three key areas: Background Study, Literature Review and Current System. The Background section provides essential information that is necessary for understanding the project, with a particular emphasis on the topic of e-commerce. The Literature Review examines related works in the field of e-commerce platforms, justifying the choice of the algorithm used in the project over others. The current system shows the existing system and their problems.

Chapter 3: System Analysis and Design provides detailed information on the algorithms that form the basis of the project. It also includes a system design that serves as a guide during the implementation phase of the project. In this project, there is also a requirement analysis which outlines the project's functional and non-functional requirements, while Feasibility Analysis explores the practicality and viability of the project's implementation.

Chapter 4: Implementation and Testing describes the workflow of the system, detailing how the project was implemented and the tools and platforms utilized in the process.

Chapter 5: Conclusion and Recommendation wraps up the document by summarizing the entire project. It also sets the stage for future research by suggesting potential enhancements that could be made to the system

Chapter 2: Literature Review

2.1 Background Study:

File compression and decompression is a field of study focused on making files smaller and then restoring them to their original form. This process involves using special techniques and algorithms to reduce file sizes, making it easier to store and transfer them. Compression is important for efficient data management, faster file transfers, and improved computer performance. However, decompression is equally crucial to restore compressed files back to their original state for use or sharing. By understanding and studying file compression and decompression, we can develop better methods to save space, send files faster, and enhance overall data management. However, it is important to acknowledge the limitations of file compression and decompression. Lossy compression techniques, commonly used for multimedia files, introduce some loss of data or quality. Additionally, the computational overhead associated with compression and decompression processes can impact system performance. Furthermore, compatibility issues with specific file formats or platforms may arise, and security concerns must be addressed to protect sensitive information during compression and decompression operations.[5]

Continued research and development in the field of file compression and decompression aim to overcome these limitations and further optimize the efficiency and effectiveness of these techniques. By refining algorithms, improving compression ratios, and addressing practical challenges, researchers strive to enhance data management capabilities, promote faster file transfers, and facilitate seamless integration into various applications and systems.

2.2 Literature Review:

Text compression is crucial for efficiently managing large volumes of textual data. Among the various methods available, Huffman coding stands out for its effectiveness in reducing file sizes while preserving data accuracy. Developed by David A. Huffman in 1952, this algorithm introduced a significant innovation by using variable-length codes that reflect the frequency of characters in a text. Frequent characters are given shorter codes, while less common ones get longer codes, leading to a noticeable reduction in file size.

Huffman coding works on the principle of entropy coding, constructing a binary tree based on character frequencies. In this tree, characters that appear more often are closer to the root, making their codes shorter. This approach ensures that no code is a prefix of another, which helps avoid confusion during decompression. This method proves particularly efficient for texts where character frequencies vary, making it ideal for scenarios where reducing file size is important without losing data.[2]

While Huffman coding is highly effective, it's worth comparing it with other compression techniques. For example, Run-Length Encoding (RLE) is great for data with lots of repeated characters. It replaces sequences of identical characters with a single character and a count. However, RLE isn't as effective for data with little repetition. The Lempel-Ziv-Welch (LZW) algorithm, used in formats like GIF and TIFF, creates a dictionary of substrings and replaces repeated substrings with references to this dictionary. Though LZW works well for various types of data, its performance can be limited by the dictionary size. Arithmetic coding is another method that encodes entire messages as a fraction, achieving higher compression ratios for data with specific probability distributions, but it's more complex to implement. The Burrows-Wheeler Transform (BWT) rearranges characters to improve compressibility and is often used with other techniques like RLE for better results.

Huffman coding has many practical applications. In data storage, it's integral to formats like ZIP and GZIP, making it essential for efficient file archiving and backup. In multimedia, Huffman coding is crucial for compressing images in JPEG and audio in MP3 formats. For example, JPEG uses Huffman coding to compress the quantized coefficients of the Discrete Cosine Transform (DCT), while MP3 uses it to compress audio data. In telecommunications, Huffman coding helps reduce data transmission sizes, which speeds up communication and lowers costs. This is evident in protocols like MPEG, which use Huffman coding to optimize video compression. Moreover, Huffman coding is useful in embedded systems, where memory and processing power are limited. Its simplicity and efficiency make it ideal for real-time compression in devices like sensors and microcontrollers. [3]

Despite its strengths, Huffman coding does face some challenges. Its static nature means it can't easily adapt to changing data characteristics. To address this, adaptive Huffman coding methods have been developed to adjust the codebook in real-time. Additionally, combining

Huffman coding with other algorithms, such as Arithmetic coding or Lempel-Ziv, is being explored to achieve even better compression ratios. Advances in hardware and parallel processing are also expected to enhance Huffman coding's performance, especially in real-time applications.[4]

In summary, Huffman coding remains a key technique in text compression due to its ability to significantly reduce file sizes while preserving data. Its versatility is evident in its wide range of applications, from data storage and multimedia to telecommunications and embedded systems. Ongoing research and advancements are likely to address its limitations and extend its use to new technologies and data types, ensuring it continues to be a valuable tool in modern data management and communication.

Chapter 3: System Analysis and Design

3.1 System Analysis

3.1.1 Requirement Analysis

Functional Requirement (Illustrated using use case diagram)

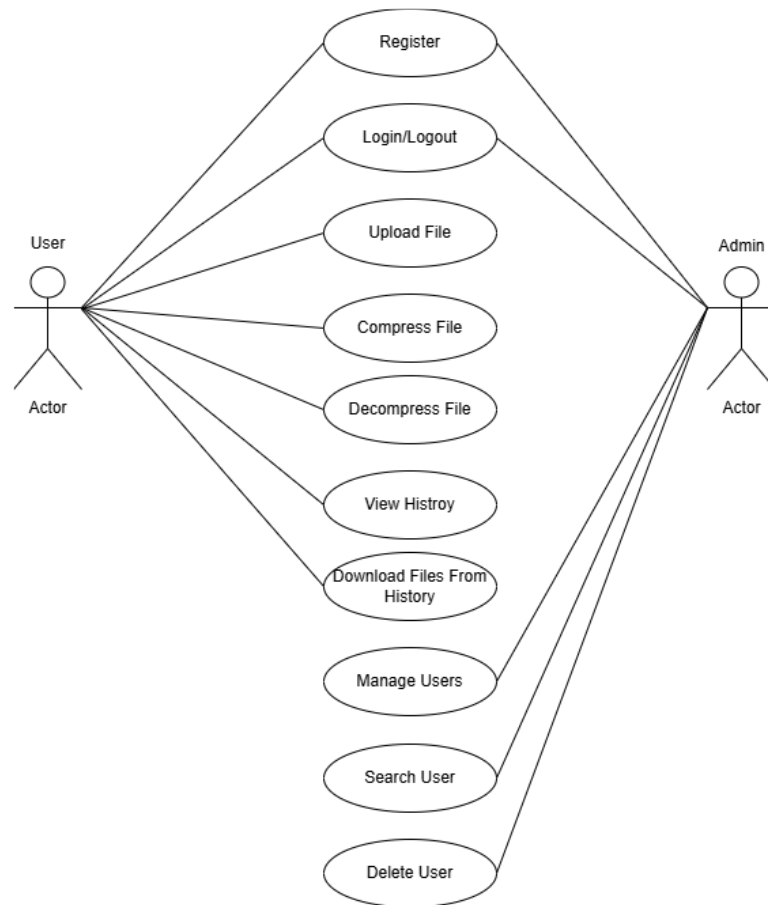


Figure 3.1.1: Use Case Diagram

1. Visit the webpage:

- Users can easily access the webpage

2. Upload File:

- The user can upload the required txt file for compression or decompression

3. Compression / Decompression:

- User can choose between compressing and decompressing the file.

4. Download Compressed or Decompressed file:

- User can easily download the required compressed or decompressed file.

Non-Functional Requirements:

1. Usability:

- The web application should have a user-friendly interface, making it easy to navigate and interact with the system.
- The web application should provide clear instructions and guidance to users for completing tasks and accessing features.

2. Performance:

- The web application should have fast response times, ensuring that users can access information and perform actions without significant delays.
- The web application should be able to handle a large number of concurrent users and maintain performance under peak loads during events or high-traffic periods.

3. Reliability:

- The web application should be reliable and available for use at all times, minimizing downtime and disruptions.

4. Compatibility:

- The web application should be compatible with different devices and platforms, such as desktops, laptops, tablets, and mobile devices.
- It should support popular web browsers and operating systems to ensure widespread usability.

3.1.2 Feasibility Study

A feasibility Study was conducted taking into consideration the particular field, to check whether this project can be implemented in the business field or not. The major analyses, which is called feasibility analysis, are as follows:

- i. Technical Analysis
- ii. Operational Analysis
- iii. Economic Analysis
- iv. Schedule Feasibility

i. Technical Analysis

A device with internet access is a necessary piece of hardware for this application. This specific gadget must support HTML5. It could be a desktop, laptop, or even a mobile device. The system can operate on any operating system, thus that is not a need. The project will be practical because the application will be built using HTML5, CSS 3.0, PHP, and JavaScript and will work on nearly all modern devices, as well as a few older ones.

ii. Operational Analysis

Users must be able to use a computer and have a basic understanding of the English language to operate the system. The system will include basic, understandable components that the intended users may use. The project will have a GUI so that anyone may utilize it with ease. Therefore, all of the operational requirements are realistic, and the project's goals are realistic as well.

iii. Economic Analysis

For the designed system to function, server space and a domain must be purchased. Additionally, the development team will need to be compensated for their services for the project to be created. The system was developed using open-source software, and as the project is entirely software-related, it didn't require a big budget. The project is therefore financially viable.

iv. Schedule Feasibility

Project Timeline: Assess the feasibility of implementing the web application compression and decompression within the desired timeframe. Consider factors such as system development, testing, data migration, user training, and potential customization requirements.

Resource Availability: Evaluate the availability of resources, both human and technical, required for system development and implementation. Ensure that the necessary resources can be allocated within the planned schedule.

3.1.3 Data Modeling (ER – Diagram)

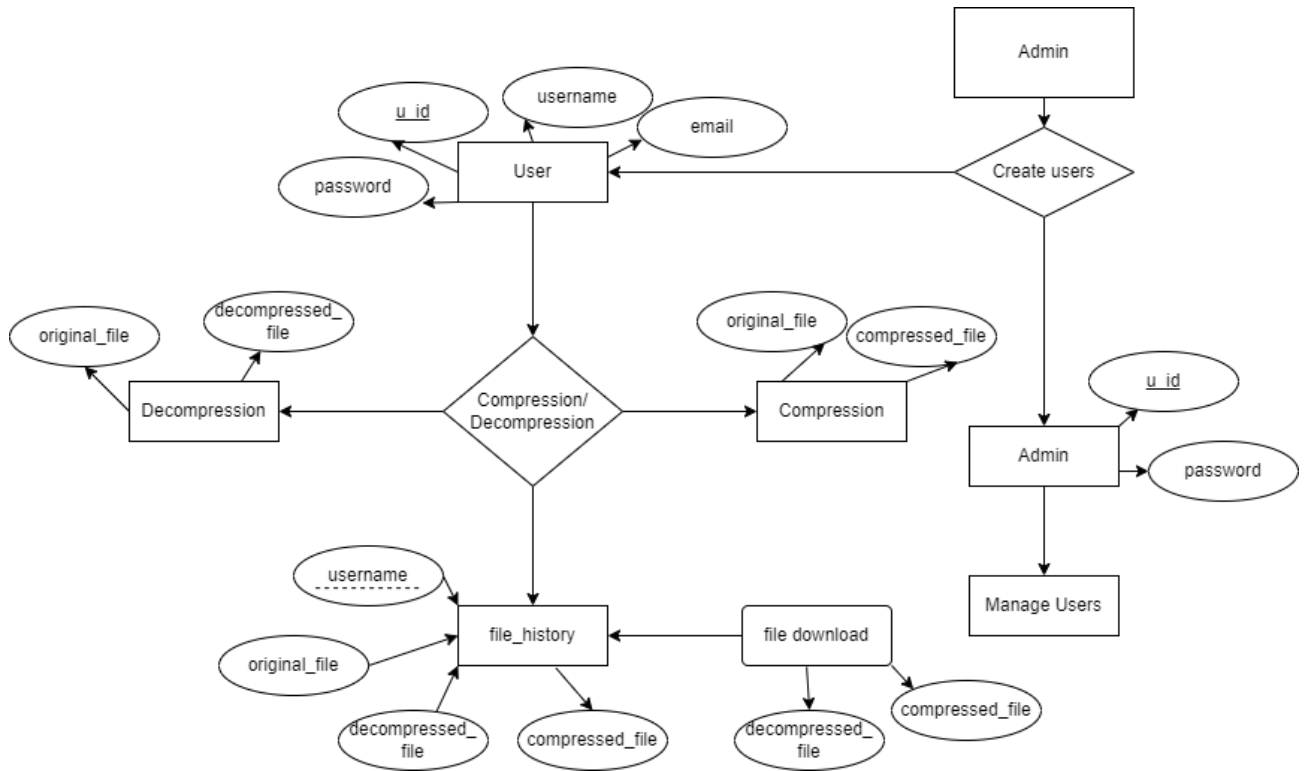


Figure Error! Bookmark not defined..1.3: ER-Diagram

This ER diagram represents a file compression and decompression system with two main roles: User and Admin. Users, identified by attributes like u_id, username, email, and password, can perform compression and decompression actions on files, which are recorded in file_history (logging username, original_file, compressed_file, and decompressed_file). Compression and Decompression entities handle files, with attributes for original, compressed, and decompressed files. Admins, who have their own u_id and password, can create and manage users, overseeing system operations but not directly performing file actions. The file download entity manages access to stored compressed and decompressed files, allowing users to download their processed files.

3.1.4 Process Modeling (DFD)

1)Level 0 DFD

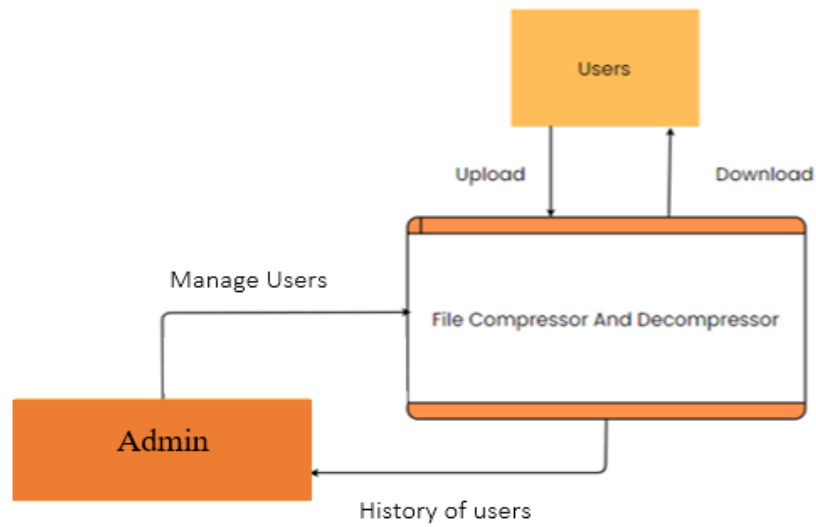


Figure 3.1.3.1: Level 0 DFD

2)Level 1 DFD

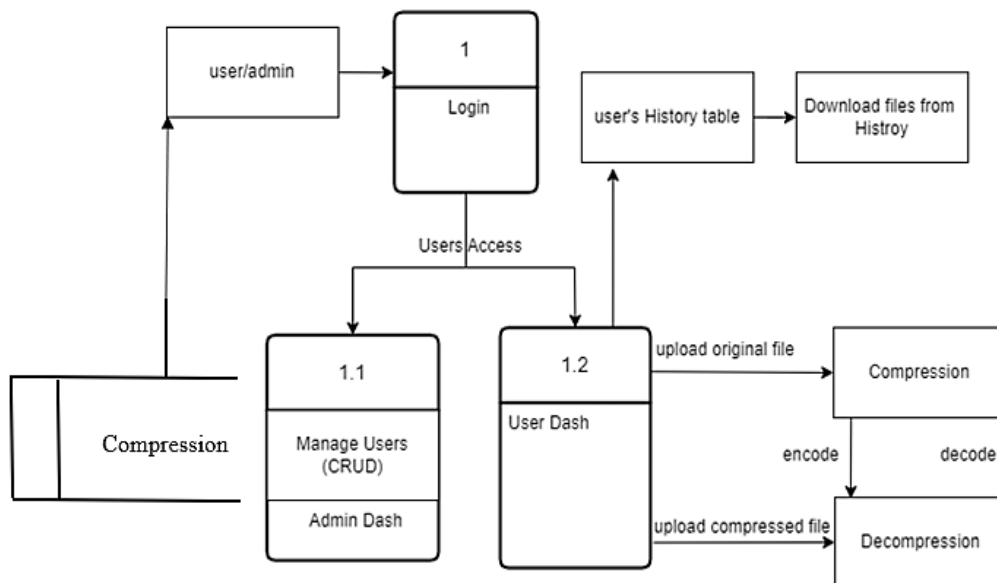


Figure 3.1.4.2: Level 1 DFD

3.2 System Design

3.2.1 Architectural Design

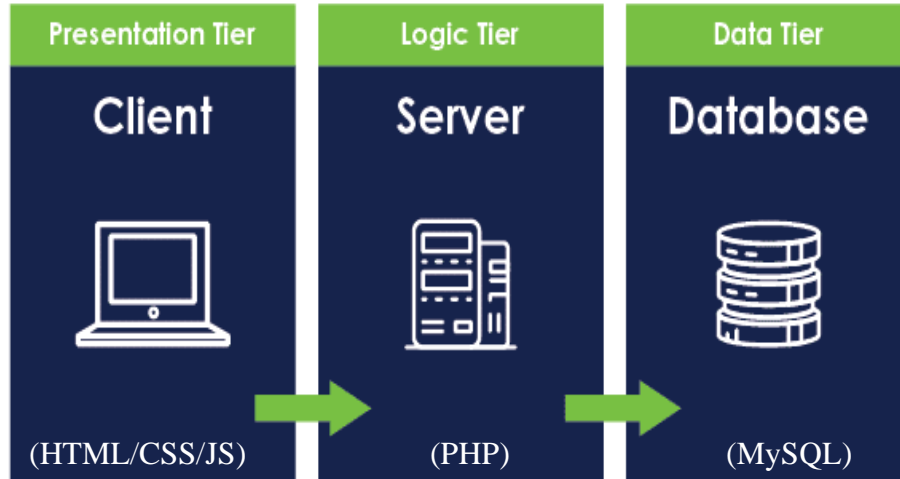


Figure 3.2.1: Architectural Design

This project employs a Client-Server Architecture, facilitating seamless interaction between Customers and Admins through a web interface. The client side allows users to engage with the system intuitively, providing functionalities for both file compression and decompression.

Client Side

Customers access the application through a web interface developed using **HTML**, **CSS**, and **JavaScript**. This setup ensures a dynamic and user-friendly experience, allowing users to:

- Upload original files for compression or decompression.
- View and manage their file history, including original and processed files.
- Download both compressed and decompressed files.
- Navigate the interface easily, thanks to responsive design principles and user-centric features.

Admins have dedicated functionalities that enable them to monitor user activities, manage file uploads, and oversee platform operations, ensuring a smooth experience for all users.

Server SideThe server component, developed using **PHP**, processes client requests, manages user profiles, and handles file processing. A unified login process simplifies authentication for both Customers and Admins, enhancing user experience and security.

Database Management

A **MySQL** database underpins the application, efficiently managing the storage and retrieval of crucial data, including user information and file history records.

Compression Algorithms

The application implements Huffman coding for efficient file compression and decompression. Users can interact with these algorithms through the web interface, allowing them to compress text and PDF files seamlessly.

This architecture optimizes performance and scalability while providing a robust framework for future enhancements, ensuring that the compression and decompression tool effectively meets user needs.

3.2.2 Database Schema Design

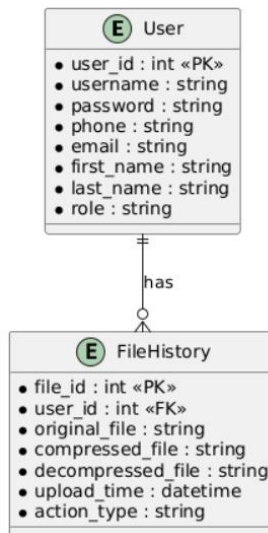
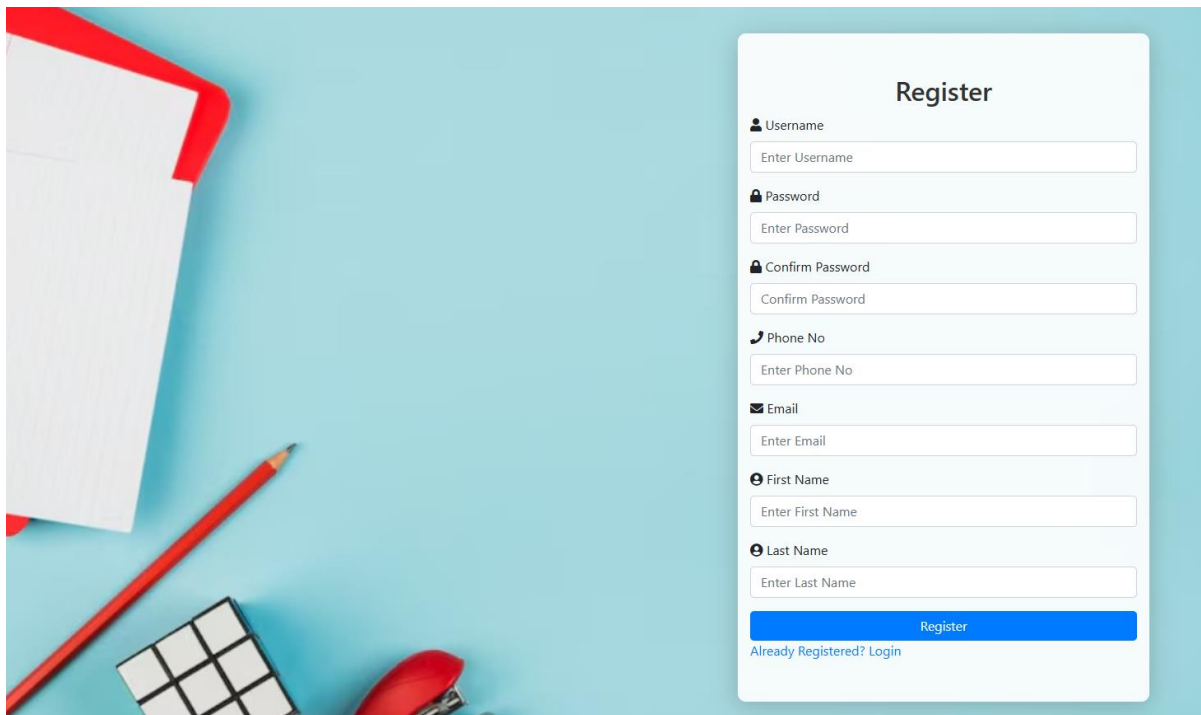


Figure 3.2.2: Database Schema

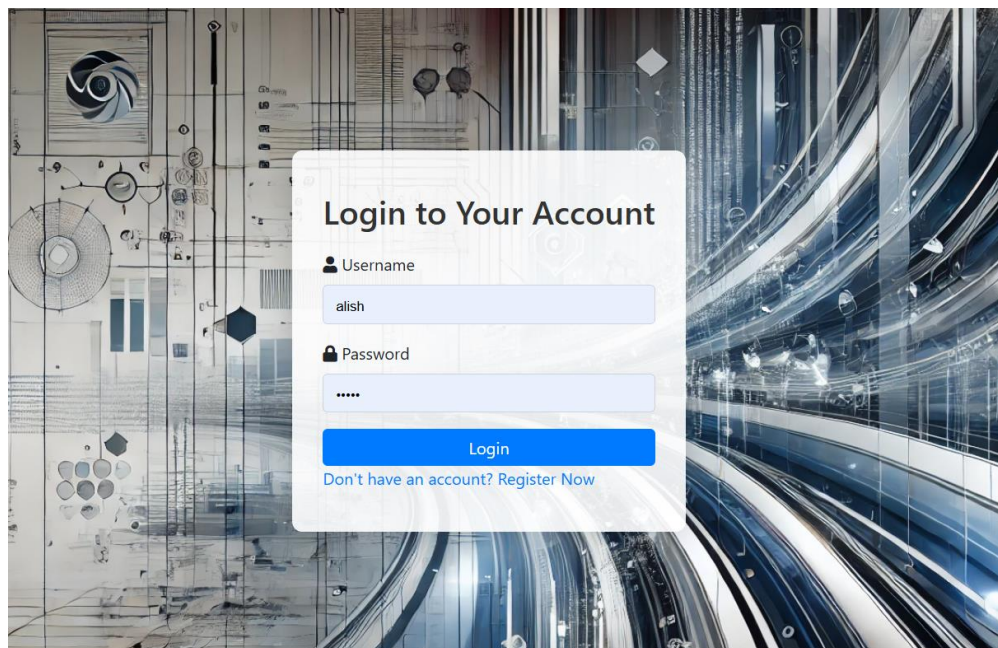
3.2.3 Interface Design



The image shows a 'Register' form overlay on a light blue background with a desk scene (notebook, pencil, eraser). The form is titled 'Register' and contains the following fields and elements:

- Username:** Input field with placeholder 'Enter Username'.
- Password:** Input field with placeholder 'Enter Password'.
- Confirm Password:** Input field with placeholder 'Confirm Password'.
- Phone No:** Input field with placeholder 'Enter Phone No'.
- Email:** Input field with placeholder 'Enter Email'.
- First Name:** Input field with placeholder 'Enter First Name'.
- Last Name:** Input field with placeholder 'Enter Last Name'.
- Register Button:** A blue button labeled 'Register'.
- Link:** A link labeled 'Already Registered? Login'.

Figure 3.2.3.1: Register Form



The image shows a 'Login to Your Account' form overlay on a background of technical drawings and a modern building. The form is titled 'Login to Your Account' and contains the following fields and elements:

- Username:** Input field with the value 'alish'.
- Password:** Input field with masked characters '*****'.
- Login Button:** A blue button labeled 'Login'.
- Link:** A link labeled 'Don't have an account? Register Now'.

Figure 3.2.3.2: Login Form

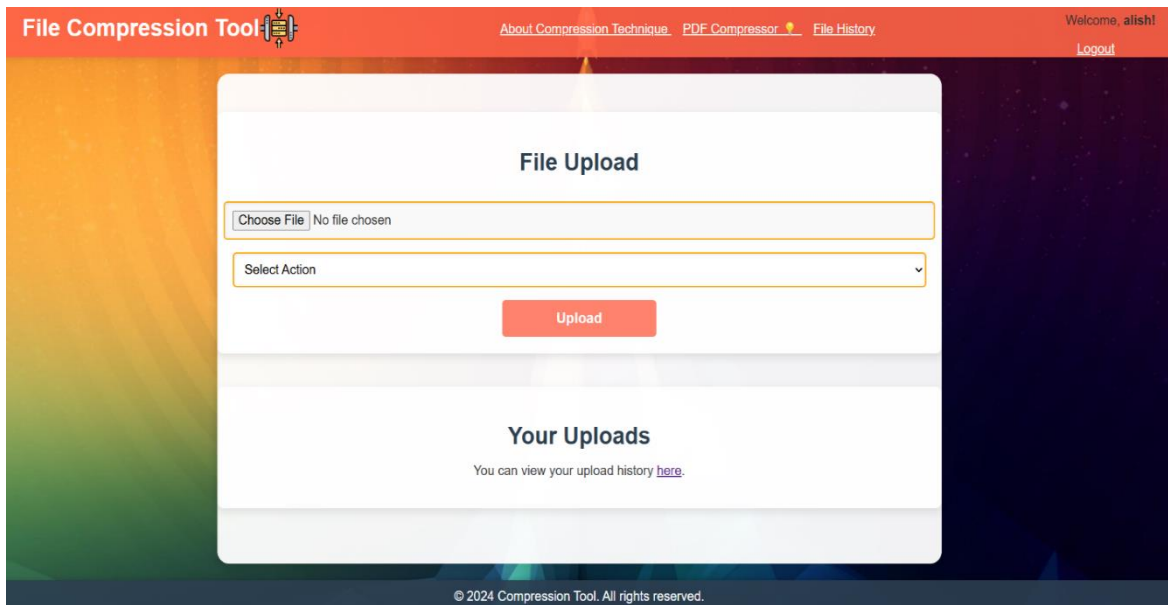


Figure 3.2.3.3: Txt Compression

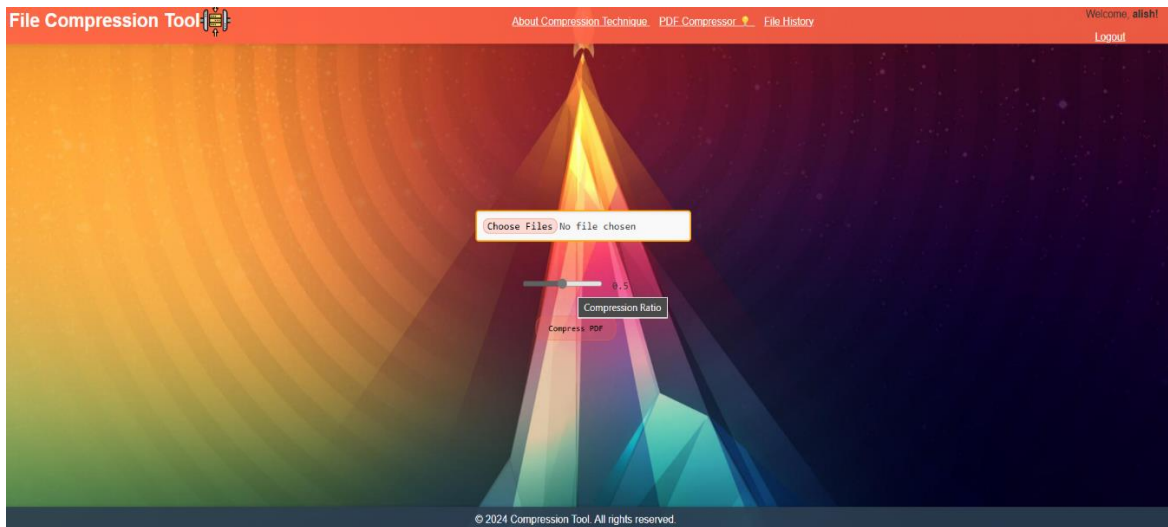


Figure 3.2.3.4: Pdf Compression

Compression Tool

Home

History

Welcome, alish

Logout

Your Upload History

Original File	Compressed File	Decompressed File	Upload Time
demoText.txt	compressed_demoText.txt		2024-10-23 15:17:06
demoText_compressed (28).txt		decompressed_demoText_compressed (28).txt	2024-10-23 15:17:26
demoText.txt	compressed_demoText.txt		2024-10-23 15:19:30
demoText.txt	compressed_demoText.txt		2024-10-23 15:19:48
demoText.txt	compressed_demoText.txt		2024-10-23 15:22:51
demoText.txt	compressed_demoText.txt		2024-10-24 04:33:38
demoText.txt	compressed_demoText.txt		2024-10-24 04:35:25
demoText.txt	compressed_demoText.txt		2024-10-24 04:35:29
bishal.txt	compressed_bishal.txt		2024-10-24 04:36:28

Figure 3.2.3.5: History of Compression

3.2.4 Physical DFD

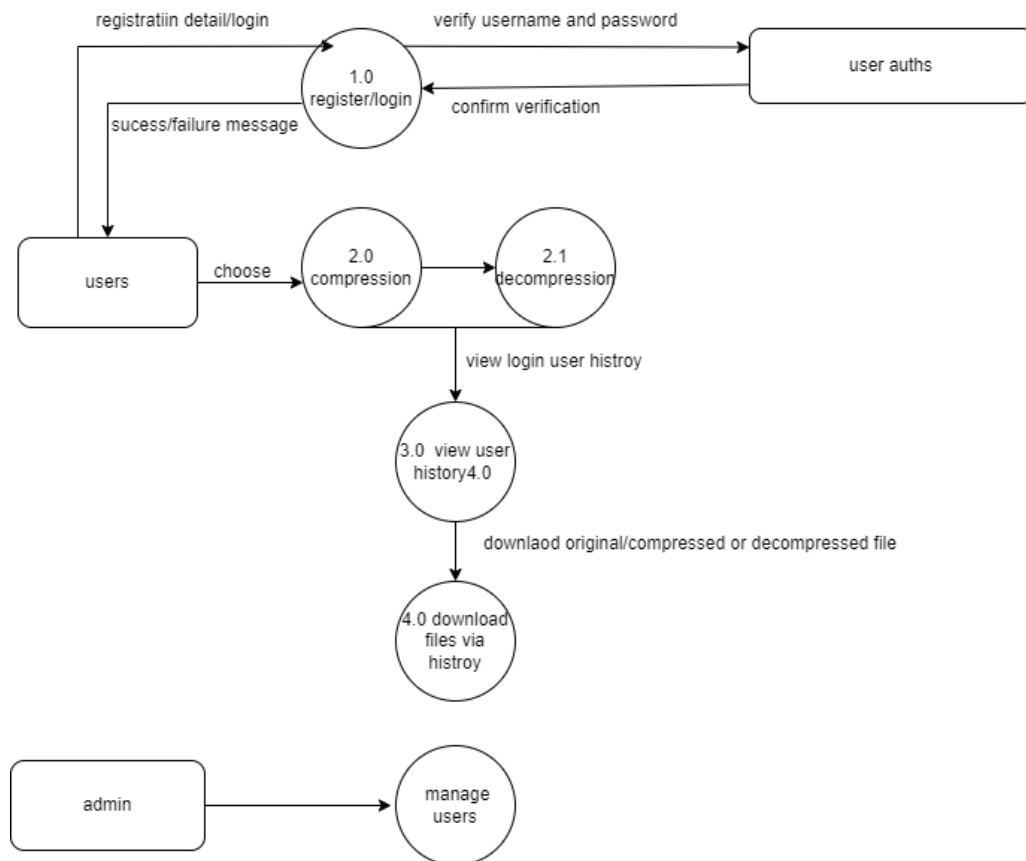


Figure 3.2.4: Physical DFD

3.3 Algorithm Used

The Huffman algorithm is a widely used method for lossless data compression. It reduces the size of data without losing any information by encoding data more efficiently based on the frequency of each character.

Steps in Huffman Algorithm

1. Frequency Analysis

- Calculate the frequency of each character in the input text.
- For the text "BCAADDDCCACACAC", the frequency would be:
 - A: 5
 - B: 1
 - C: 6
 - D: 3 add the conversion size for this case

2. Building the Huffman Tree

- Create a leaf node for each character and build a priority queue (min-heap) based on the frequency of characters.

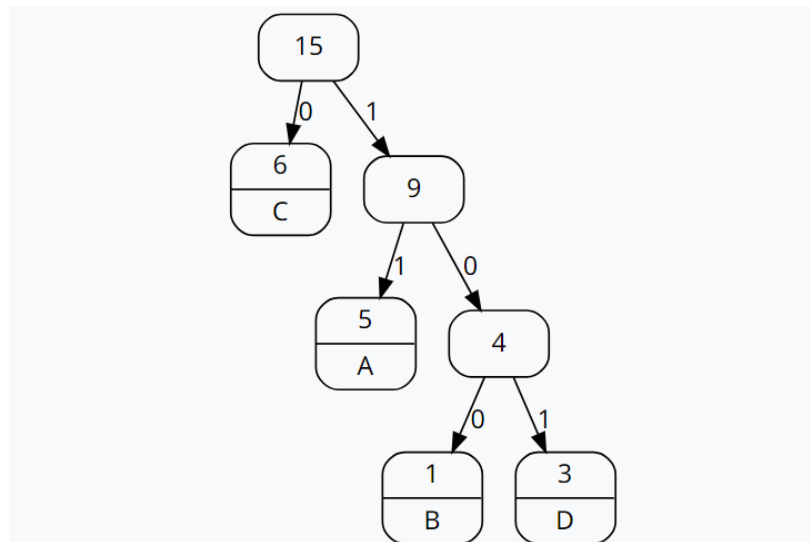


Figure 3.3: Huffman Tree

- While there is more than one node in the queue:
 1. Extract the two nodes with the smallest frequencies.
 2. Create a new internal node with these two nodes as children and the sum of their frequencies as the new frequency.
 3. Insert the new node back into the priority queue.
- The remaining node is the root of the Huffman tree.

3. Generating Huffman Codes

- Traverse the Huffman tree from the root to the leaves, assigning a binary code to each character:
 - Go left, append '0' to the code.
 - Go right, append '1' to the code.
- Each character will have a unique binary code based on its position in the tree.
- Huffman codes:
 - A: 10
 - B: 1100
 - C: 0
 - D: 110

4. Encoding the Data

- Replace each character in the input text with its corresponding Huffman code.
- Example: "BCAAD" would be encoded as "1100010110".

5. Decoding the Data

- Use the Huffman tree to decode the encoded data:
 - Start at the root of the tree and follow the path indicated by the binary code until a leaf node is reached.
 - Append the character of the leaf node to the output.
 - Repeat until all binary codes are decoded.
- Example: "1100010110" would be decoded back to "BCAAD".

The text "BCAADDDCCACACAC":

1. **Frequency Analysis:**

- A: 5, B: 1, C: 6, D: 3

2. **Building the Huffman Tree:**

- Create nodes for B, C, A, and D with their respective frequencies.
- Combine the lowest frequency nodes until only one node (the root) remains.

3. **Generating Huffman Codes:**

- C: 0, A: 10, D: 110, B: 111

4. **Encoding:**

- Original text: "BCAADDDCCACACAC"
- Encoded text: "1110100110110111001010100"

5. **Decoding:**

- Encoded text: "1110100110110111001010100"
- Decoded text: "BCAADDDCCACACAC"

Original Size Before Compression:

- Each character is assumed to use 8 bits in ASCII encoding.
- The original text length: 15 characters
- Original size in bits = 15 characters \times 8 bits/character = **120 bits**

Compressed Size After Encoding:

- Encoded text: "1110100110110111001010100" has 25 bits.
- **Compressed size in bits = 25 bits**

Compression Ratio:

- **Compression Ratio** = (Compressed size) / (Original size)
- Compression Ratio = 25 bits / 120 bits = **0.208**

Advantages of Huffman Algorithm

- **Efficiency:** Produces a prefix-free code which ensures no code is a prefix of another, allowing for efficient and error-free decoding.
- **Optimality:** Minimizes the average code length, achieving optimal compression for a given set of character frequencies.

Applications

- **File Compression:** Used in ZIP files, JPEG images, and other compressed formats.
- **Data Transmission:** Reduces the amount of data to be sent, saving bandwidth and transmission time.

Chapter 4: Implementation and Testing

4.1 Implementation

4.1.1 Tools used

CASE (Computer-Aided Software Engineering) tools are software applications that assist in various stages of the software development lifecycle, including requirements gathering, design, coding, testing, and maintenance.

- **Unified Modeling Language (UML) Tools:**

UML tools can help in creating and visualizing system models, including use case diagrams, class diagrams, sequence diagrams, and state diagrams.

- **Data Modeling Tools:**

Tools like Erwin, MySQL Workbench, or draw.io can be used to create entity-relationship (ER) diagrams.

- **Code Editors and Integrated Development Environments (IDEs):**

IDE Visual Studio provide advanced coding features, debugging capabilities, and integration with version control systems to streamline the development of the system.

- **Version Control Systems:**

Version control tools such as Git enable collaborative development, versioning, and tracking of source code changes.

- **Documentation Tools:**

Tools like Microsoft Word or Google Docs can be used for creating and maintaining project documentation, including requirements specifications, design documents, and user manuals.

- **Client-Side:**

HTML, CSS, and JavaScript are used for the user interface and interactivity.

- **Server-Side:** PHP is utilized for backend processing, while SQL is used for database interactions to manage compression history.

4.1.2 Implementation Details of Modules

User Modules

- **Responsibilities:** Manages user registration, authentication, and account details, including viewing compression history.
- **Interactions:** Connects with authentication functions for login/logout and interacts with the database to retrieve user-specific compression records.

Compression Modules

- **Responsibilities:** Handles the compression and decompression of files using Huffman coding, ensuring efficient data handling.
- **Interactions:** Interfaces with file upload functionality and communicates with the database to log compression activities.

History Modules

- **Responsibilities:** Manages the storage and retrieval of the user's compression and decompression history.
- **Interactions:** Connects with the database to save user actions and displays the history of compressed files to the user in the interface.

File Management Modules

- **Responsibilities:** Manages file operations, including uploading original files and downloading compressed files.
- **Interactions:** Works with the Compression Modules for processing files and connects with the User Modules to ensure user-specific file handling.

Admin Modules

- **Responsibilities:** Provides administrative functionalities, such as viewing user histories and managing system settings.
- **Interactions:** Interfaces with the History Modules to access user data and communicates with the database for any necessary administrative updates.

Logging and Error Handling Modules

- **Responsibilities:** Logs system activities and errors for monitoring and debugging purposes.
- **Interactions:** Integrates with all other modules to capture and log events related to file processing and user actions.

4.2 Testing

Testing is an essential process to evaluate the functionality of a software application, ensuring that it is free from errors and meets the desired quality standards. It goes beyond fixing bugs and involves different testing techniques to produce a reliable and efficient product.

4.2.1 Test Case for Unit Testing

Table 4.2.1: System Testing

Test Case ID	Test Description	Input	Expected Output	Actual Output	Status
UT-001	Test file upload functionality	Valid file (e.g., .txt)	Success message with file details	File uploaded successfully.	Passed
UT-002	Test file upload with invalid format	Invalid file (e.g., .exe)	Error message indicating invalid format	Invalid file format.	Passed
UT-003	Test compression function	Original file	Compressed file	Compression successful.	Passed
UT-004	Test decompression function	Compressed file	Original file	Decompression successful.	Passed
UT-005	Test input validation for user ID	Invalid user ID	Error message indicating invalid user ID	User ID not found.	Passed

4.2.2 Test Case for System Testing

Table 4.2.1: System Testing

Test Case ID	Test Description	Input	Expected Output	Actual Output	Status
ST-001	Test complete upload and compression workflow	Valid file (e.g., .txt)	Compressed file with success message	File compressed successfully.	Passed
ST-002	Test complete upload and decompression workflow	Compressed file	Original file with success message	File decompressed successfully.	Passed
ST-003	Test error handling for invalid file upload	Invalid file (e.g., .exe)	Error message indicating invalid file upload	Invalid file format.	Passed
ST-004	Test handling of simultaneous uploads	Multiple valid files	Cannot add more than one file	All files not uploaded successfully.	Failed
ST-005	Test database connection	N/A	Connection success	Database connection successful.	Passed

Chapter 5: Conclusion and Future Recommendations

5.1 Conclusion:

The mid-term report for the File Compression and Decompression Using Huffman Algorithm project provides a clear overview of the progress made so far and the key aspects of developing the system. It covers important elements such as the system's goals, requirements, design, and a feasibility analysis.

Throughout this phase, we've gained a deeper understanding of the system's functional and non-functional requirements. Diagrams like use case, entity-relationship (ERD), and data flow (DFD) have helped illustrate how the system works, its data structure, and its processes. The high-level design and architecture have been thoughtfully planned, creating a solid foundation for moving forward with implementation. Additionally, a feasibility study has shown that the project is on the right track, with positive outcomes in terms of technology, cost, efficiency, and scheduling.

As the project continues, the File Compression and Decompression Using Huffman Algorithm system is expected to significantly improve file compression. This will offer users key benefits, such as saving storage space and speeding up file transfers. Ultimately, the system will enhance efficiency in handling files, making it a valuable tool for many users.

5.2 Lesson Learned / Outcome

Through the course of this project, I gained in-depth knowledge of compression algorithms, specifically Huffman coding, and their practical applications in file compression and decompression. This study provided important insights into the strengths and weaknesses of these algorithms, particularly regarding their efficiency in reducing file size while maintaining data integrity.

I learned how crucial it is to design a user-centric system that balances performance and usability, ensuring that users can easily upload and access their files without sacrificing the effectiveness of the compression process. Challenges encountered, such as handling various file formats, optimizing compression rates, and managing user data securely, underscored the need for continuous improvements to enhance the system's accuracy and efficiency.

Final Result:

- The outcome is a personalized and efficient compression tool that significantly improves user experience.

Future Development:

- The project has the potential to incorporate machine learning techniques for better prediction of file compression rates.
- There are opportunities to develop context-aware features that adapt the compression methods based on user preferences and file types.

Overall Impact:

- The compression tool serves as a valuable resource for users needing efficient file management.
- It lays the groundwork for ongoing enhancements and adaptability in technology.

5.3 Future Recommendations:

For future improvements to the File Compression and Decompression Using Huffman Algorithm, here are a few simple recommendations:

1. **Better Handling of Large Files:** While Huffman coding works well, it can be slow with very large files. Future updates could focus on making the system faster by using techniques like parallel processing or combining Huffman coding with other methods to speed things up.
2. **Support for More File Types:** Right now, the system may only work with certain file types. Expanding it to support images, videos, and other formats would make it more useful. Tailoring the algorithm for different kinds of data could also improve the compression results.
3. **Make the System Easier to Use:** Adding a simple, user-friendly interface (such as drag-and-drop features or batch processing) would make the system more accessible, especially for people who aren't familiar with coding.

4. **Cloud Integration:** Allowing users to compress and decompress files through cloud services would make the system more convenient. This would let users access the system from anywhere and share files more easily.
5. **Improve Data Reliability:** Adding error detection and correction features would help ensure that files remain intact during compression and decompression, especially when transferring files over the internet or unreliable networks.
6. **Show Performance Stats:** Giving users real-time information, like how much space they're saving or how fast the compression is happening, could make the system more useful and transparent.
7. **Explore New Compression Methods:** Researching and experimenting with other compression algorithms or combining Huffman coding with newer techniques might result in even better file compression performance. [4]

References

- [1] Huffman, D. A. (1952). A Method for the Construction of Minimum-Redundancy Codes. *Proceedings of the IRE*, 40(9), 1098-1101. doi:10.1109/JRPROC.1952.273898
- [2] Salomon, D. (2007). *Data Compression: The Complete Reference* (4th ed.). Springer.
- [3] Sayood, K. (2017). *Introduction to Data Compression* (5th ed.). Morgan Kaufmann.
- [4] MacKay, D. J. C. (2003). *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press.
- [5] Mark Nelson, *The Data Compression Book, 2nd Edition, M&T Books*, 1995. ISBN: 978-1558514348.
- [6] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein, *Introduction to Algorithms, 3rd Edition, MIT Press*, 2009. ISBN: 978-0262033848.