Graphic Era deemed to be University DEHRADUN

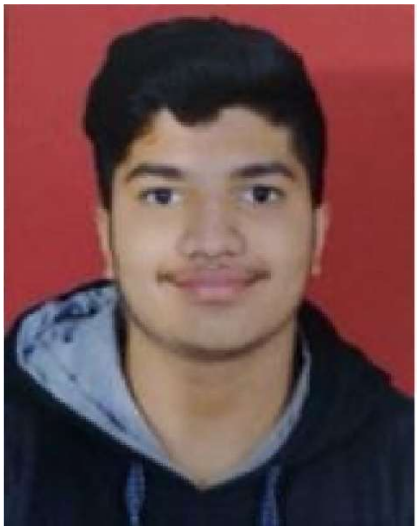
PROJECT AND TEAM INFORMATION

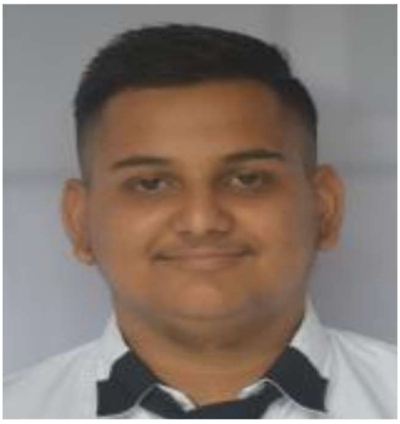
PROJECT AND TEAM INFORMATION

Project Title

Efficient File Compression Using Huffman Encoding

Student / Team Information

Team Name: *Team #*	*Bytesquashers*
Team member 1 (Team Lead)	*Ashish Gupta* *9411742244* *theashish2004@gmail.com* 

Team member 2	***Aditya Kimothi*** ***7505726811*** *adityakimothi31@gmail.com* 
Team member 3	*Sameer Lohani* *7983956049* *sameerlohani200510@gmail.com* 

PROJECT PROGRESS DESCRIPTION (35 pts)

Project Abstract

- *The aim of this project is to create a practical and user-friendly file compression software based on Huffman encoding, one of the most popular lossless compression schemes. Due to the rising demand for data optimization storage and transmission in the digital age, particularly in cloud storage and network delivery, the necessity of practical file compression has never been greater. This project seeks to solve the problem of big file sizes through the application of Huffman encoding within a light, open-source utility tool that is easy to use and learn.*

- *The tool will have a C++ backend for compression and decompression via Huffman encoding and a web interface with simple file uploads and downloads. This project aims to make Huffman coding easier to use by providing a customizable open-source solution unlike widely used commercial software like WinRAR or 7-Zip. The system will combine frontend technologies such as HTML, CSS, and JavaScript with backend assistance from Node.js and Express.js for file operations to provide a smooth user experience.*

- *With this project, we seek to show how Huffman coding can be applied in real-world applications, compress files, and possibly reduce costs in data storage and transmission. The ultimate outputs are an implementable compression and decompression engine, an interactive web interface, and comprehensive project documentation.*

Updated Project Approach and Architecture

- *The project uses a file compression program in C++ for the back-end, utilizing the Huffman encoding algorithm for lossless data compression. HTML, CSS, and JavaScript are utilized to create the front-end with a friendly web interface to handle file uploads and downloads.*

- *In order to integrate the frontend and backend, Node.js and Express.js form an API for facilitating free exchange. HTTP protocols and AJAX calls are applied for effective file exchange between the system and the user.*

- *The C++ compression algorithm manages file reading and writing via File I/O operations, allowing maximum file compression. This modular approach gives high priority to effectiveness and usability, keeping the system light and user-friendly.*

Tasks Completed

Task Completed	Team Member
Design and algorithm implementation ***1. For Compression*** • ***Character Frequency Analysis:*** *The program reads the input file in binary mode and builds a frequency table of characters.* • ***Huffman Tree Construction:*** *With the help of a min-heap priority queue, the program constructs the Huffman tree by merging nodes with the lowest frequencies in an iterative manner.* • ***Code Assignment:*** *Every character is assigned a binary code depending on its location in the Huffman tree. This is achieved recursively.* • ***File Compression:*** *The input file is read again, and characters are substituted with their respective Huffman codes to produce a compressed bitstream.*	*Ashish Gupta*
2. For Decompression • ***Command-Line Argument Handling:*** *The program takes one argument (the input compressed file) and checks for its existence.* • ***File Handling:*** *It opens the input file in binary mode and forms the name of the output file by replacing "-compressed" with "-decompressed.txt".* • ***Extension Handling:*** *It reads one character that signifies the length of the file extension and then reads and adds the extension to the output file name. This is done to ensure the original file*	*Sameer Lohani*

extension is maintained. • *Reading Huffman Code Map:* *The application reads Huffman codes and corresponding characters from the compressed file. It reads a null character ('\\0') as a delimiter to divide code entries, constructing a reverse lookup map (decodeMap) to decode.*	
Web Pages ***1. Home Page*** • ***Page Structure and Layout:*** *The HTML page is organized with a middle container (#wrapper) that centers the content vertically and horizontally, offering a tidy, minimalist-looking user interface.* • ***Visual Styling with CSS:*** *There_are two different fonts used through Google Fonts: 'Oswald' for regular text and 'Kalam' for headings to achieve a contemporary and innovative appearance. Custom styles determine the layout, colors, and interactivity, such as button hover effects that_animate smoothly with an upward shift.* • ***Responsive Design Elements:*** *Bootstrap CSS is added to provide compatibility with different screen sizes and devices. Buttons are styled to look sleek and interactive through transitions and custom padding.* • ***Functional Navigation:*** *Two buttons, "COMPRESS" and "DECOMPRESS", are assigned to their corresponding pages (compress.html and decompress.html) so users can trigger the intended action.* • ***Branding and Thematic Content:*** *The main heading properly conveys the tool's purpose — "File Compression using Huffman Coding" — with proper emphasis and visual hierarchy.* ***2. For Compression*** • ***Page Structure and Styling:*** *The layout aligns the form both horizontally and vertically with CSS. Google Fonts (Oswald and Kalam) improve the*	*Sameer Lohani* *Aditya Kimothi*

look and feel, and added custom styling maintains harmony with the overall application theme. • ***Form Generation for File Upload:*** *A form is declared with the suitable attributes (action=\"/compress\", method=\"POST\", enctype=\"multipart/form-data\") to support file uploads to a backend compression handler.* • ***User Guidelines:*** *Simple labels are used to direct the user in choosing a file and waiting for compression. Font formatting makes it easy to read and properly formatted relative to the rest of the UI.* • ***Input Elements:*** *A file input field is used to choose a file.* *A submit button (Upload) is used to trigger compression.* *A placeholder paragraph (#success) is added for displaying upload status or success messages dynamically.* • ***Download Button:*** *The styled download button is offered, connected to the /download path, where users can obtain the zip file after processing is finished.* ***3. For Decompression*** • ***Page Structure and Styling:*** *The design is done with clean visual hierarchy with Google Fonts (Oswald and Kalam) and Bootstrap for responsiveness. The use of .form-container in the center aligns the content, while uniform font styling provides a unified user experience.* • ***File Upload Form for Decompression:*** *A form is used with action=\"/decompress\", method=\"POST\", and enctype=\"multipart/form-data\" for the user to upload a binary file to decompress.* *An input field allows for file selection, and a themed Upload button initiates the submission.* • ***User Instructions:*** *Instructional labels direct users through the decompression and upload process, giving a clear indication of anticipated file type and post-upload behavior.* • ***Download Option:***	*Ashish Gupta*

A download button is offered to obtain the decompressed file using the /download endpoint to provide an efficient user flow from upload to access. • ***Success and Error Feedback (Alert Boxes):*** *Two collapsible alert boxes (#myalert for success, #myalert1 for failure) are available to display feedback messages dynamically to users.* *Close buttons enable users to close these manually, enhancing interactivity.* • ***Success Message Placeholder:*** *The #success paragraph is available for displaying decompression status updates dynamically.*	

Challenges/Roadblocks

1. ***Algorithm Complexity and Optimization:***
   Implementing the Huffman encoding algorithm efficiently in C++ required careful handling of data structures like priority queues and trees. Initially, our implementation suffered from high memory usage and slow performance on large files. To address this, we are optimizing memory management and exploring the use of bit-level operations for more compact storage.
2. ***User Interface Usability:***
   Creating a simple, intuitive interface that handles uploads/downloads while showing progress or errors is a priority. We are iteratively improving the UI using user feedback and ensuring responsive design through modern frontend practices.
3. ***Team Coordination and Time Management:***
   As students balancing academics and project work, coordinating tasks has been difficult. Through continuous testing and optimization we aim to overcome these challenges and deliver a robust, user-friendly compression tool.

Future Scopes

The current implementation of this project lays a solid foundation for practical and educational use of Huffman encoding in file compression. However, there are several potential enhancements and expansions that can be explored in the future to broaden its capabilities, usability, and real-world application.

1. Support for Multiple File Formats

Extend the tool to handle not just text-based files but also other formats such as images (.jpg, .png), PDFs, and multimedia files.

Implement logic to determine optimal strategies for different file types, possibly *combining Huffman with other algorithms.*

2. Real-Time Compression Statistics

- *Provide users with detailed feedback such as compression ratio, space saved, and time taken.*
- *Visual analytics (charts/graphs) can be added to the web interface for better understanding of performance.*

3. Enhanced User Interface

- *Add drag-and-drop file upload, progress bars, and multilingual support to make the interface more interactive and accessible.*
- *Integrate dark mode or customizable themes to improve user experience.*

4. Cloud Integration

- *Enable direct compression and decompression of files from cloud storage services like Google Drive, Dropbox, or OneDrive.*
- *Facilitate secure cloud-based compression for low-storage devices.*

5. Mobile Application Version

- *Develop a lightweight Android/iOS version of the tool for mobile file management and compression needs on the go.*

6. Integration of Other Compression Algorithms

- *Introduce a modular backend that supports alternative or hybrid algorithms (e.g., Run-Length Encoding, LZW, Arithmetic Coding).*
- *Let users choose the most suitable algorithm based on their needs.*

7. Multi-File and Folder Compression

- *Allow batch file compression or folder-based operations to handle multiple files in a single archive.*
- *Add ZIP-style packaging and extraction features.*

8. Security and Encryption

- *Implement basic encryption options during compression to ensure file.*
- *security, especially for sensitive data.*
- *Add user authentication features for secure usage.*

9. Web Hosting and API Service

- *Host the application publicly with a REST API to allow integration with third-party systems or educational platforms.*
- *Offer the service as an open-source SaaS (Software as a Service) model.*

10. Performance Optimization

- *Further optimize the C++ implementation to handle very large files using multithreading or memory mapping.*
- *Use parallelism or GPU acceleration for faster processing.*

Project Outcome/Deliverables

- *C++ Compression Engine: A backend for Huffman encoding for compressing and decompressing files efficiently.*
- *Web Interface: A basic, easy-to-use frontend developed with HTML, CSS, and JavaScript for uploading and downloading files.*
- *API Integration: An Express.js and Node.js API integrating the frontend and backend for file processing.*
- *Tested System: A reliable system tested on different txt file for strong performance.*

Documentation: Document project in extensiveness from a user-manual to technical implementation of Huffman encoding.

Progress Overview

As of now, a significant portion of the project has been completed. The core C++ compression and decompression algorithms using Huffman encoding have been successfully implemented and tested

- *with sample files. Additionally, the web interface — including Home, Compress, and Decompress pages — has been designed and developed, meeting our planned milestones for frontend design.*

- *Currently, the project is slightly behind schedule in terms of integration. The connection between the frontend interface and the backend C++ engine is still in progress, as handling file I/O between Node.js and native C++ requires careful execution and testing. We are prioritizing this task now to stay on track for final testing.*

The focus for the coming days will be on completing frontend-backend integration, performing extensive testing on various file types, and finalizing documentation. With continued progress, we expect to complete the project within the remaining timeline.

Codebase Information

*Repository Link: **https://github.com/Sameer200510/PBL***
Main Branch: main

Deliverables Progress

C++ Program for Huffman Decompression and Compression
Done – Main algorithm programmed and tested on different types of files.

Web Interface for File Operations (Home, Compress, Decompress Pages) Done –
Responsive frontend created through HTML, CSS, and JavaScript.

Frontend with Backend Integration (C++ Engine through Node.js)
In Progress – File uploading functionality implemented; integration of C++ backend is in progress.

Testing on Different Types of Files
In Progress – Passed small to medium file testing; pending for large file and edge case testing.

Fully Working Demo Hosted Locally
Pending – Will be completed once backend integration and testing are done.

Project Documentation and User Manual
In Progress – Basic framework done; final touches will be done once the system is completed.