# A black and red text Description automatically generated

**PROJECT AND TEAM INFORMATION**

## Project Title

|  |
| --- |
| File Compression and Decompression using Huffman Coding |

## Student/Team Information

|  |  |
| --- | --- |
| Team Name:  Team # (Mentor needs to assign) | BitBusters  [DAA-IV-T145](https://lms.geu.ac.in/course/view.php?id=337) |
| Team member 1 (Team Lead)  (Last Name, name: student ID: email, picture): | Chaudhary, Manvi – 23022908  [nainachaudharydehradun@gmail.com](mailto:nainachaudharydehradun@gmail.com) |
| Team member 2  (Last Name, name: student ID: email, picture): | Nivedan, Harsh – 230211585  [harshnivedan@gmail.com](mailto:harshnivedan@gmail.com) |
| Team member 3  (Last Name, name: student ID: email, picture): | Agarwal, Vidit Saran – 23021864  [viditagarwal60@gmail.com](mailto:viditagarwal60@gmail.com) |
| Team member 4  (Last Name, name: student ID: email, picture): | Mittal, Rishika – 23022417  [rishikamittal2005@gmail.com](mailto:rishikamittal2005@gmail.com) |

**PROJECT PROGRESS DESCRIPTION (35 pts)**

## Project Abstract (2 pts)

|  |
| --- |
| Our project delivers a high-performance, lossless compression and decompression tool built entirely on Huffman Coding. By parsing UTF-8 text files to calculate character frequencies, constructing an optimal prefix code tree, and executing bit-level encoding, we achieve significant reductions in file size without sacrificing any information.  Written in C++ for speed and resource efficiency, the tool also provides clear, educational insights into the inner workings of Huffman Coding. A lightweight web interface will enable users to compress common formats (e.g., .txt, .csv, .json) and instantly visualize compression ratios, making it ideal for bandwidth- or storage-constrained environments. |

## Updated Project Approach and Architecture (2 pts)

|  |
| --- |
| Our current approach focuses on lossless text compression using Huffman Coding, implemented in C++. The system reads a raw text file, builds a frequency map of characters, generates a binary Huffman Tree, and derives character-specific prefix codes for efficient encoding. The output is a compressed binary file with a prepended header for decompression metadata.  The system is modularized into several components:   * Tree Construction: Using a custom Tree struct and priority\_queue, we construct the Huffman Tree based on character frequencies. * Compression:   + Input is read into a buffer using fread.   + Encoded into a bitstring via map lookups and packed into bytes.   + A header is written with metadata: number of padding bits and code mappings. * Decompression:   + The header is parsed to reconstruct the Huffman code map.   + Bitstream is read and decoded using reverse lookup in map<string, unsigned char>.   Libraries Used:   * <iostream>, <map>, <vector>, <queue>, and <cstdio> for system I/O and data handling. * <cstdlib>, <cstring>, and <algorithm> for buffer operations and transformations.   Architecture:   * Follows a file**-**basedpipeline:   + test.txt → [Compression] → compressed.bin (with header + encoded data)   + compressed.bin → [Decompression] → decoded.txt |

## 

## Tasks Completed (7 pts)

|  |  |
| --- | --- |
| Task Completed | Team Member |
| Designed and implemented file input/output modules for reading/writing raw bytes  Developed Huffman tree construction and character frequency-based encoding logic  Created and validated the custom file header format to store encoding metadata  Implemented bitstring generation, padding, and conversion to/from byte streams  Connected compression and decompression modules into a complete CLI workflow  Performed initial testing and validation on different file types | Manvi Chaudhary  Harsh Nivedan  Vidit Saran Agarwal  Rishika Mittal  Harsh Nivedan  All |

## 

## Challenges/Roadblocks (7 pts)

|  |
| --- |
| * Bitstring Padding Logic**:** Ensuring bitstreams are padded to byte boundaries during compression, and trimmed correctly during decompression, was nontrivial and caused errors in early testing. * Cross-Platform File Handling: Binary file I/O behaved inconsistently across platforms due to newline transformations in text mode. Force binary mode for all file streams and test on both Windows and Linux. * UI integration: Bridging C++ and web front-end. Currently using command-line output log and later upgrade to a basic JS-based dashboard for interaction. * Handling Unsupported or Corrupted Files: If a user tries to decompress a file that was not generated by our compressor, it may lead to undefined behaviour or crashes. We will handle this using the frontend. |

## Tasks Pending (7 pts)

|  |  |
| --- | --- |
| Task Pending | Team Member (to complete the task) |
| Finalize UI for file selection and visualization  Implement compression ratio and size metrics display  Comprehensive testing on .json, .xml, .html, .yaml  Performance optimization (large files)  Documentation and user manual | Manvi Chaudhary  Vidit Saran Agarwal  Harsh Nivedan  Rishika Mittal  All |

## 

## Project Outcome/Deliverables (2 pts)

|  |
| --- |
| By the end of this project, we’ll deliver:   * A fully functional text file compression/decompression tool. * Support for .txt, .csv, .json, .xml, .html, .log, .md, and .yaml formats. * GUI interface for ease of use. * Codebase with documentation and performance analysis (compression ratio, runtime, etc.) |

# Progress Overview (2 pts)

|  |
| --- |
| Our project has made substantial progress, aligning well with the original timeline. We have completed 70% of the project including core compression/decompression logic and utilities, and validated functionality on basic text formats.  Ahead of schedule: algorithm implementation and initial tests.  Behind schedule: UI integration and large-file performance testing.  We anticipate finishing remaining tasks in the next two weeks. |

# Codebase Information (2 pts)

|  |
| --- |
| Repository: https://github.com/CosmicIon/File-compression-and-decompression-using-Huffman-Coding  Branch: main  Key commits:  [adding the proposal and sample test files](https://github.com/CosmicIon/File-compression-and-decompression-using-Huffman-Coding/commit/b071f5fea7b585891411b4d047dbf54d0e909e7a)  [skeletal program](https://github.com/CosmicIon/File-compression-and-decompression-using-Huffman-Coding/commit/37f9fe61e4d4a1beb730ca99221e754078833f4e)  [remove the header file dependecies](https://github.com/CosmicIon/File-compression-and-decompression-using-Huffman-Coding/commit/495565fa65d5ada685c9a160a221d407446a1435)  [fixed bugs, include the header files, run the file.](https://github.com/CosmicIon/File-compression-and-decompression-using-Huffman-Coding/commit/671a38dfb2bfe90a0a5892bd2525cb954e87cb1a)  [updated the running command on readme](https://github.com/CosmicIon/File-compression-and-decompression-using-Huffman-Coding/commit/270f847fa9a10afc56d7a6fc6188004d29b3cd6c) |

## Testing and Validation Status (2 pts)

|  |  |  |
| --- | --- | --- |
| Test Type | Status (Pass/Fail) | Notes |
| Unit tests (utility functions)  Compression on .txt  Compression on .csv  Compression on .html  Compression on .xml  Compression on .json  Compression on .md | Pass  Pass  Pass  Pass  In Progress  In Progress  In Progress | Edge cases for empty buffers  Verified by bytewise comparison  Handled commas and newlines  Handled HTML tags |

# Deliverables Progress (2 pts)

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | Command-line compression tool | Completed | | Command-line decompression tool | Completed | | Support for all specified text file formats | In Progress | | UI for file selection and visualization | Pending | | Documentation and performance analysis | In Progress | |