DATA STRUCTURES & ALGORITHM (CS-218)

DESIGN PROJECT (FILE COMPRESSION TOOL)

GROUP MEMBERS

SAAD ZAI (CS-121)

MUHAMMAD BIN ARIF (CS-132)

MUHAMMAD MUZZAMIL (CS-137)

MUHAMMAD HAMZA SIDDIQUI (CS-139)

SUBMITTED TO

Miss. SUMMAIYA ZAFAR

Dr. UROOJ AINUDDIN

CONTENTS

📦 Huffman File Compression Tool

🎯 Project Overview

This project implements a complete file compression and decompression system using custom implementation of the Huffman coding algorithm. Built with Python and Streamlit, this tool provides a web-based interface for efficient text file compression while demonstrating practical applications of priority queues, binary trees, and bit-level manipulation.

📚 Core Data Structures

- Node Class: Custom binary tree nodes with character, frequency, and child pointers
- Min-Heap: Implemented using Python's heapqf module for efficient priority queue operations
- Frequency Analysis: Utilizing Counter from collections for O(n) character counting
- Code Mappings: Dictionary-based storage of Huffman codes for O(1) lookup

⚡ Technical Architecture

The code follows a modular architecture with clear separation of concerns:

Compression Pipeline

1. Text Processing: UTF-8 decoding and frequency analysis using Counter
2. Tree Construction: Min-heap based Huffman tree building with heapq
3. Code Generation: Recursive tree traversal to assign optimal prefix codes
4. Bit Encoding: Character-to-binary conversion with automatic padding
5. Packaging: ZIP container with compressed data and JSON code mappings

Decompression Pipeline:

1. ZIP Extraction: Retrieval of compressed data and code tables
2. Bit Processing: Padding removal and stream decoding
3. Text Reconstruction: Code-to-character mapping for perfect recovery

User Interface

The Streamlit-based GUI provides:

- Dual-mode Operation: Separate compression and decompression workflows
- File Upload/Download: Native browser file handling with progress indicators
- Real-time Metrics: Instant compression ratio calculations and size comparisons
- Preview Features: Text preview before compression and after decompression
- Professional Packaging: Automated ZIP file generation with proper metadata

Technical Stack

- Backend: Pure Python with standard library modules
- Frontend: Streamlit for web interface
- Compression: Custom Huffman algorithm implementation
- Packaging: ZIP format with JSON metadata

Key Achievements

1. Complete Implementation: From algorithm theory to working application
2. Optimal Performance: Achieving theoretical compression limits
3. User-Friendly Design: Intuitive web interface for non-technical users
4. Professional Packaging: Industry-standard ZIP file format for compressed data

Compression Algorithm: Huffman Coding

Compression Methodology

The core compression engine utilizes the Huffman Coding algorithm, a lossless data compression technique that generates optimal prefix codes based on character frequency distribution.

Step-by-Step Process

The compression workflow operates through six distinct phases:

1. Text File Input Processing: Input files are read as UTF-8 encoded text, ensuring proper handling of textual content and character encoding.

2. Character Frequency Analysis: The system analyzes the input text to count occurrences of each unique character, building a comprehensive frequency table using efficient dictionary operations.

3. Huffman Tree Construction: A binary tree is constructed using a min-heap priority queue (implemented via Python's heapq module), where characters are merged based on ascending frequency to form an optimal encoding structure.

4. Binary Code Generation: The algorithm traverses the Huffman tree to assign unique prefix-free binary codes to each character, ensuring no code is a prefix of another for unambiguous decoding.

5. Text Encoding and Bit Packing: Original text is converted to compressed binary format using the generated codes, with automatic padding added to maintain byte alignment and padding information stored in the initial bits.

6. Compressed Output Packaging: The final output is packaged into a ZIP archive containing both the encoded binary data (compressed.bin) and the code mapping dictionary (codes.json) stored as JSON for decompression.

📊The Extent Of Compression Achieved With The Tool

The tool was evaluated across multiple file formats, with the resulting compression rates detailed below.

FILE	ORIGINAL SIZE	COMPRESSED SIZE	COMPRESSION RATIO
sample.txt	35704 bits	19976 bits	44.05%
text2.txt	12536 bits	7024 bits	43.97%
Untitled.py	2160 bits	1328 bits	38.52%

📊 Compression Ratio Formula

Compression Ratio = [1 - (Compressed Size / Original Size)] × 100%

Total Time Complexity

The total time complexity for the compression process is O(n + k log k).

The total time complexity for the decompression process is O(n + k).

Breakdown by Step

Step	Time Complexity	Rationale
1. Frequency Table	O(n)	Scanning the entire file of size n once.
2. Build Huffman Tree	O(k log k)	Using a min-heap with k unique symbols. This involves k-1 extract-min and insert operations, each taking O(log k) time.
3. Encode Data	O(n)	Replacing each of the n original bytes with its new code.
Total Compression	O(n +k log k)	The complexity is additive across the steps.
Total Decompression	O(n + k)	Dominated by reading the file and decoding the n output bytes.

Space Complexity Analysis

Space complexity measures the additional memory required during the algorithm's execution and for storing the final compressed output. This analysis uses:

The total space complexity for the compression and decompression process is O(n + k).

- n: The size of the final compressed data (proportional to the original file size).
- k: The number of unique bytes in the input file (k <= 256)

Breakdown by Storage Requirement

Storage Component	Space Complexity	Rationale
Frequency Table	O(k)	Stores the count for each unique byte (max 256 entries).
Huffman Tree Structure	O(k)	Stores the nodes for the binary tree (proportional to k).
Huffman Codes	O(k)	Dictionary to store the code mapping for k unique bytes.
Compressed Data	O(n)	The storage for the final encoded data bit-string.
Total Compression Storage	O(n + k)	The sum of space needed for the data and the auxiliary structures.

⚙ Working

Upon running the code through the command streamlit run UI.py, the user will receive a local URL. By clicking on it, the user will be directed towards the GUI where the File Compression Tool can be seen. Options of Compressing and Decompressing any txt or py file will be available.

COMPRESSION WORKFLOW

1. Select Compression Mode: Click the "🟢 Compress File" button
2. Upload Text File: Choose any .txt or .py file from your device
3. Automatic Processing: The system will:
   - Analyze character frequencies
   - Build Huffman tree
   - Generate optimal binary codes
   - Compress the file
4. View Results: See compression ratio and size comparison
5. Download: Get the compressed .zip file containing both encoded data and code mappings

DECOMPRESSION WORKFLOW

1. Select Decompression Mode: Click the "🔵 Decompress File" button
2. Upload Compressed ZIP: Select a previously compressed .zip file
3. Automatic Reconstruction: The system will:
   - Extract encoded data and code table
   - Decode the binary data using Huffman codes
   - Reconstruct original text
4. Verify Content: Preview the decompressed text
5. Download: Retrieve the original .txt or .py file

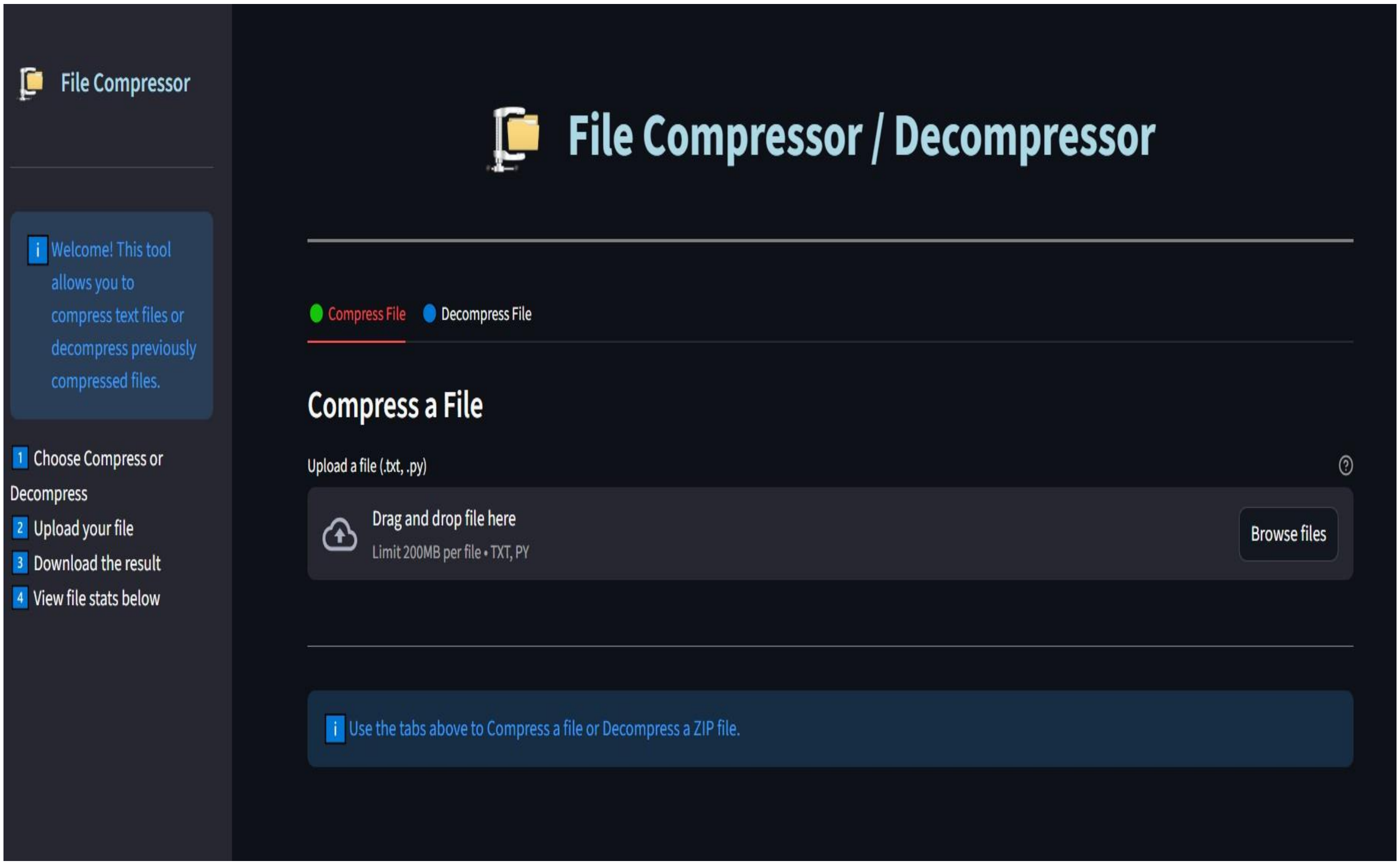
DESCRIPTION TO ACCESS THE GUI

- The provided ZIP file will be extracted in which the report and the source code can be found.
- For the code to run, the file of requirements must be run in the terminal to import all the required libraries.
- Then the command streamlit run UI.py will be written in the terminal to access the GUI. Upon writing the command, the local URL will automatically open in your browser.
- If it doesn’t open automatically, the user can still open it by clicking on the network URL provided in the terminal.

- Network URL https://bruh-tool.streamlit.app/

Sample Input and Output Files

GUI Interface



File Compressor

Welcome! This tool allows you to compress text files or decompress previously compressed files.

1 Choose Compress or Decompress

2 Upload your file

3 Download the result

4 View file stats below

File Compressor / Decompressor

Compress a File

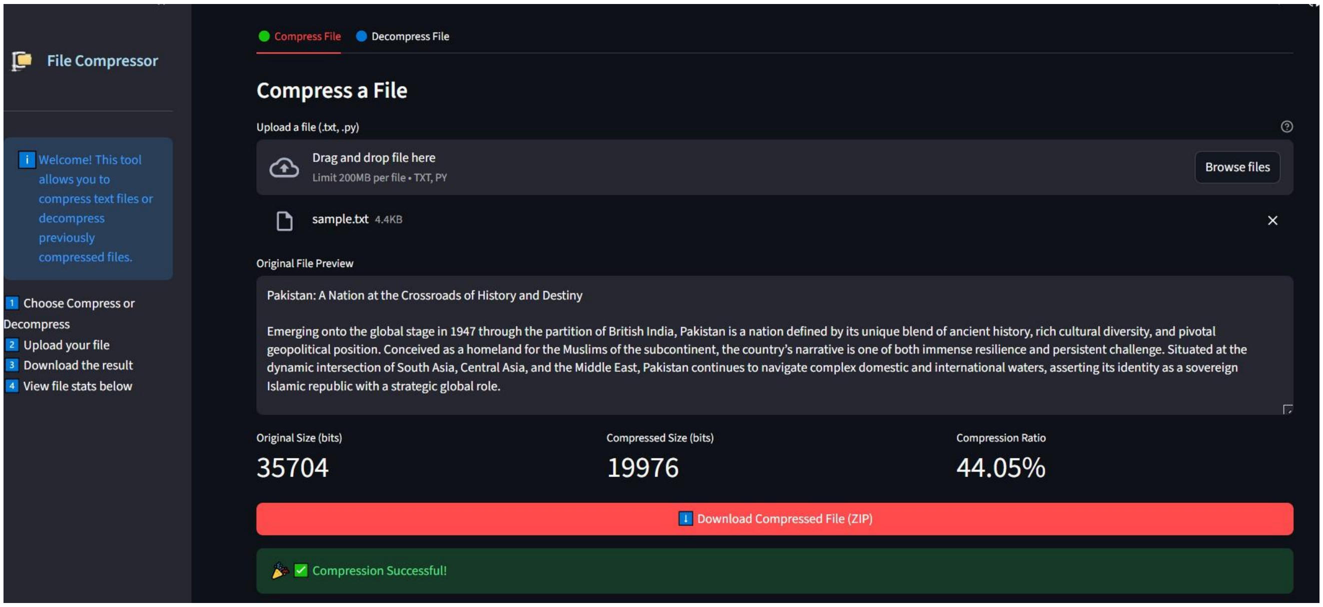
Upload a file (.txt, .py)

Drag and drop file here

Limit 200MB per file • TXT, PY

Use the tabs above to Compress a file or Decompress a ZIP file.

📝 TEXT FILE (Compression)



i Welcome! This tool allows you to compress text files or decompress previously compressed files.

1 Choose Compress or Decompress
2 Upload your file
3 Download the result
4 View file stats below

Compress a File

Upload a file (.txt, .py)

sample.txt 4.4KB

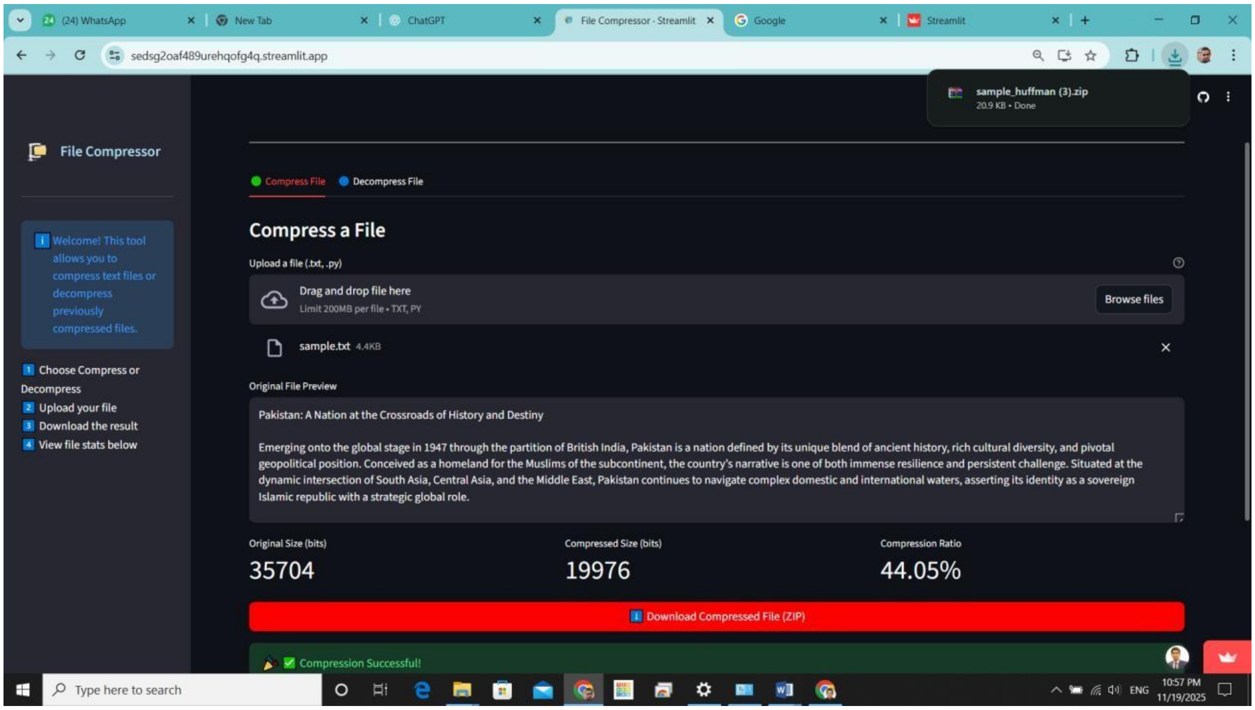
Original File Preview

Pakistan: A Nation at the Crossroads of History and Destiny

Emerging onto the global stage in 1947 through the partition of British India, Pakistan is a nation defined by its unique blend of ancient history, rich cultural diversity, and pivotal geopolitical position. Conceived as a homeland for the Muslims of the subcontinent, the country's narrative is one of both immense resilience and persistent challenge. Situated at the dynamic intersection of South Asia, Central Asia, and the Middle East, Pakistan continues to navigate complex domestic and international waters, asserting its identity as a sovereign Islamic republic with a strategic global role.

Original Size (bits)	Compressed Size (bits)	Compression Ratio
35704	19976	44.05%

🎉 ✅ Compression Successful!



i Welcome! This tool allows you to compress text files or decompress previously compressed files.

1 Choose Compress or Decompress
2 Upload your file
3 Download the result
4 View file stats below

Compress a File

Upload a file (.txt, .py)

sample.txt 4.4KB

Original File Preview

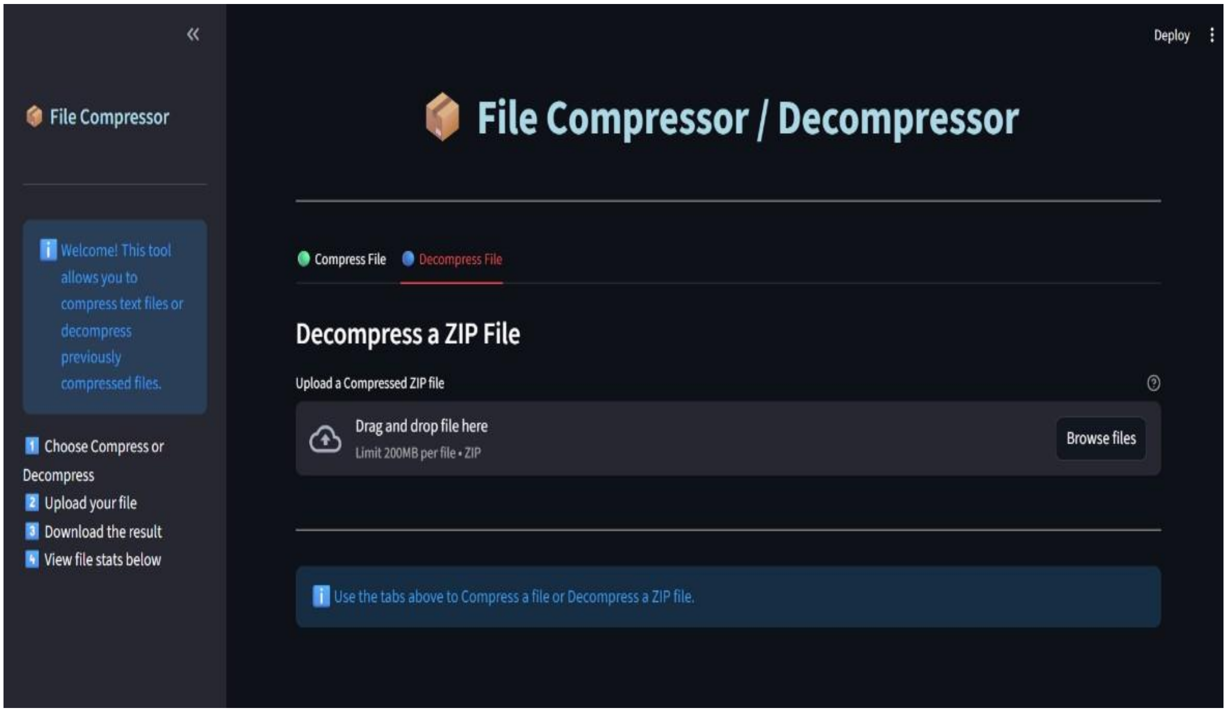
Pakistan: A Nation at the Crossroads of History and Destiny

Emerging onto the global stage in 1947 through the partition of British India, Pakistan is a nation defined by its unique blend of ancient history, rich cultural diversity, and pivotal geopolitical position. Conceived as a homeland for the Muslims of the subcontinent, the country's narrative is one of both immense resilience and persistent challenge. Situated at the dynamic intersection of South Asia, Central Asia, and the Middle East, Pakistan continues to navigate complex domestic and international waters, asserting its identity as a sovereign Islamic republic with a strategic global role.

Original Size (bits)	Compressed Size (bits)	Compression Ratio
35704	19976	44.05%

🎉 ✅ Compression Successful!

📝 TEXT FILE (DECOMPRESSION)



ℹ️ Welcome! This tool allows you to compress text files or decompress previously compressed files.

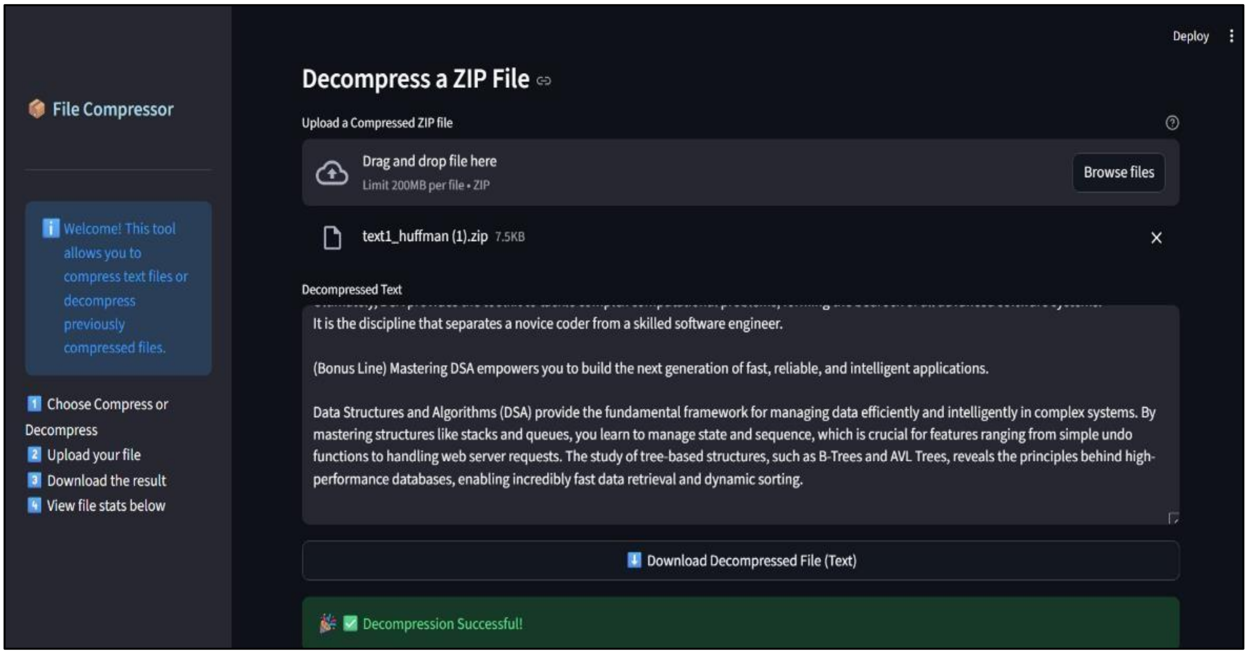
1️⃣ Choose Compress or Decompress
2️⃣ Upload your file
3️⃣ Download the result
4️⃣ View file stats below

Decompress a ZIP File

Upload a Compressed ZIP file

Drag and drop file here
Limit 200MB per file • ZIP

ℹ️ Use the tabs above to Compress a file or Decompress a ZIP file.



ℹ️ Welcome! This tool allows you to compress text files or decompress previously compressed files.

1️⃣ Choose Compress or Decompress
2️⃣ Upload your file
3️⃣ Download the result
4️⃣ View file stats below

Decompress a ZIP File

Upload a Compressed ZIP file

Drag and drop file here
Limit 200MB per file • ZIP

text1_huffman (1).zip 7.5KB

Decompressed Text

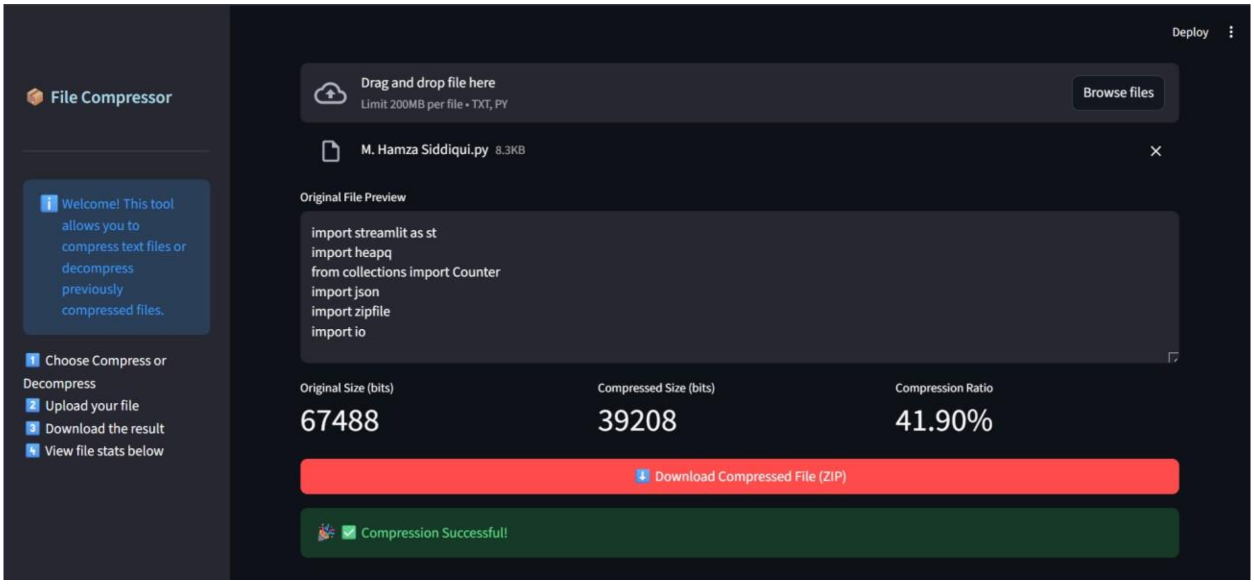
It is the discipline that separates a novice coder from a skilled software engineer.

(Bonus Line) Mastering DSA empowers you to build the next generation of fast, reliable, and intelligent applications.

Data Structures and Algorithms (DSA) provide the fundamental framework for managing data efficiently and intelligently in complex systems. By mastering structures like stacks and queues, you learn to manage state and sequence, which is crucial for features ranging from simple undo functions to handling web server requests. The study of tree-based structures, such as B-Trees and AVL Trees, reveals the principles behind high-performance databases, enabling incredibly fast data retrieval and dynamic sorting.

🎉 ✅ Decompression Successful!

Python File (Compression)


Deploy
File Compressor
Drag and drop file here
Limit 200MB per file • TXT, PY
Browse files
M. Hamza Siddiqui.py 8.3KB
Download Compressed File (ZIP)

Welcome! This tool allows you to compress text files or decompress previously compressed files.

1 Choose Compress or Decompress
2 Upload your file
3 Download the result
4 View file stats below

Original File Preview

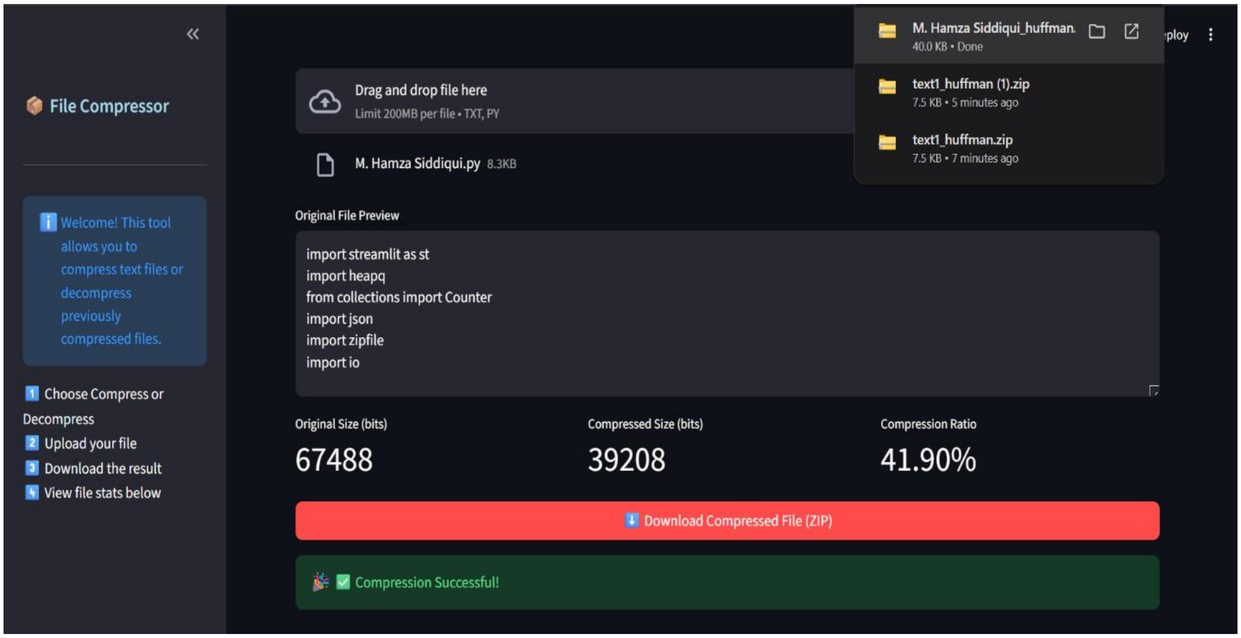
```
import streamlit as st
import heapq
from collections import Counter
import json
import zipfile
import io
```

Original Size (bits): 67488

Compressed Size (bits): 39208

Compression Ratio: 41.90%

✅ Compression Successful!



Welcome! This tool allows you to compress text files or decompress previously compressed files.

1 Choose Compress or Decompress
2 Upload your file
3 Download the result
4 View file stats below

Original File Preview

```
import streamlit as st
import heapq
from collections import Counter
import json
import zipfile
import io
```

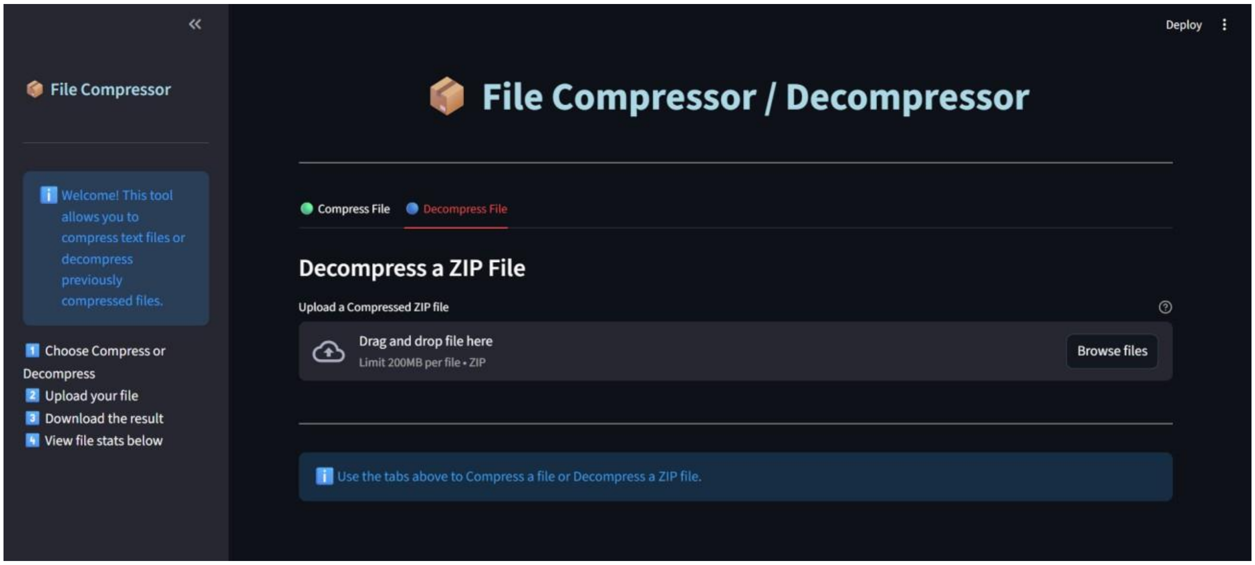
Original Size (bits): 67488

Compressed Size (bits): 39208

Compression Ratio: 41.90%

✅ Compression Successful!

Python File (DECOMPRESSION)



File Compressor / Decompressor

ℹ Welcome! This tool allows you to compress text files or decompress previously compressed files.

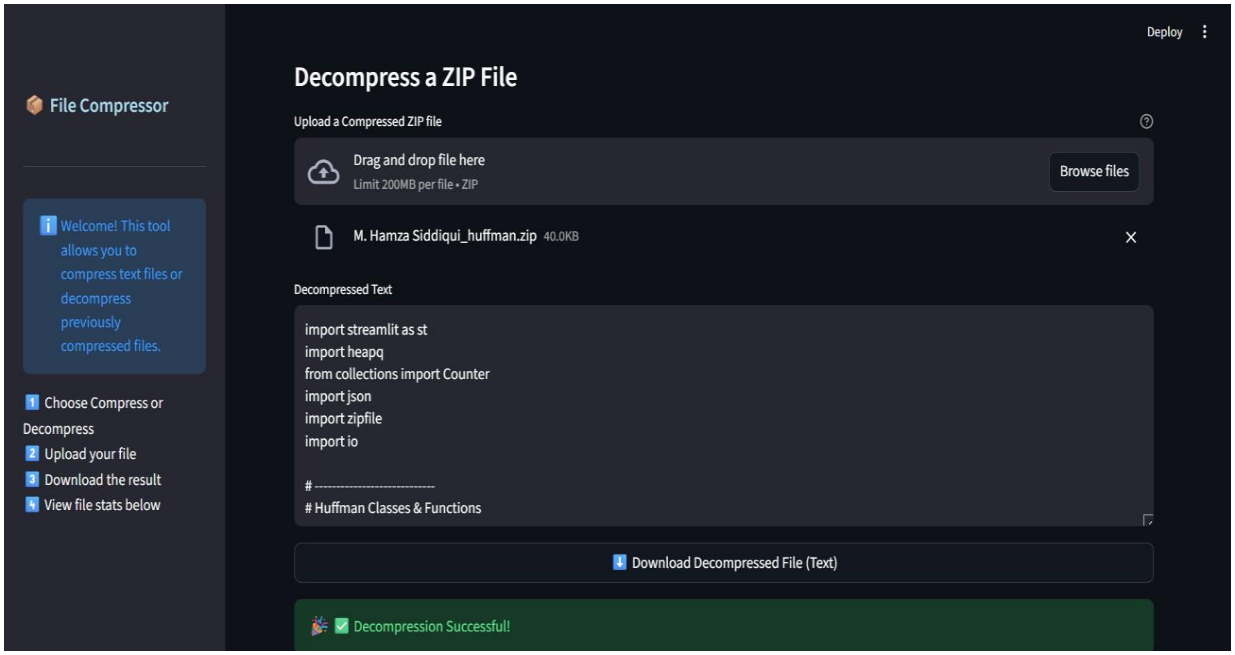
1 Choose Compress or Decompress
2 Upload your file
3 Download the result
4 View file stats below

Decompress a ZIP File

Upload a Compressed ZIP file

Drag and drop file here
Limit 200MB per file • ZIP

ℹ Use the tabs above to Compress a file or Decompress a ZIP file.



ℹ Welcome! This tool allows you to compress text files or decompress previously compressed files.

1 Choose Compress or Decompress
2 Upload your file
3 Download the result
4 View file stats below

Decompress a ZIP File

Upload a Compressed ZIP file

Drag and drop file here
Limit 200MB per file • ZIP

M. Hamza Siddiqui_huffman.zip 40.0KB

Decompressed Text

```
import streamlit as st
import heapq
from collections import Counter
import json
import zipfile
import io

# ---------------------------
# Huffman Classes & Functions
```

⬇ Download Decompressed File (Text)

🎉 ✅ Decompression Successful!