



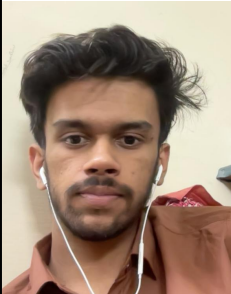


Graphic Era
deemed to be **University**
DEHRADUN

PROJECT AND TEAM INFORMATION

Project Title

Efficient LZW Compression/Decompression Using Trie Dictionary

Student/Team Information

Team ID:	<u>DAA-IV-T251</u>
Team member 1 (Team Lead) Sumit Kumar Student ID: 23021201 Email id: sumitkumarbittuair@gmail.com	
Team member 2 Aviral Maheshwari Student ID: 230213637 Email id: maheshwariaviral05@gmail.com	
Team member 3 Anubhav Vashistha Student ID: 230211649 Email id: anubhavvashistha2005@gmail.com	

PROJECT PROGRESS DESCRIPTION (35 pts)

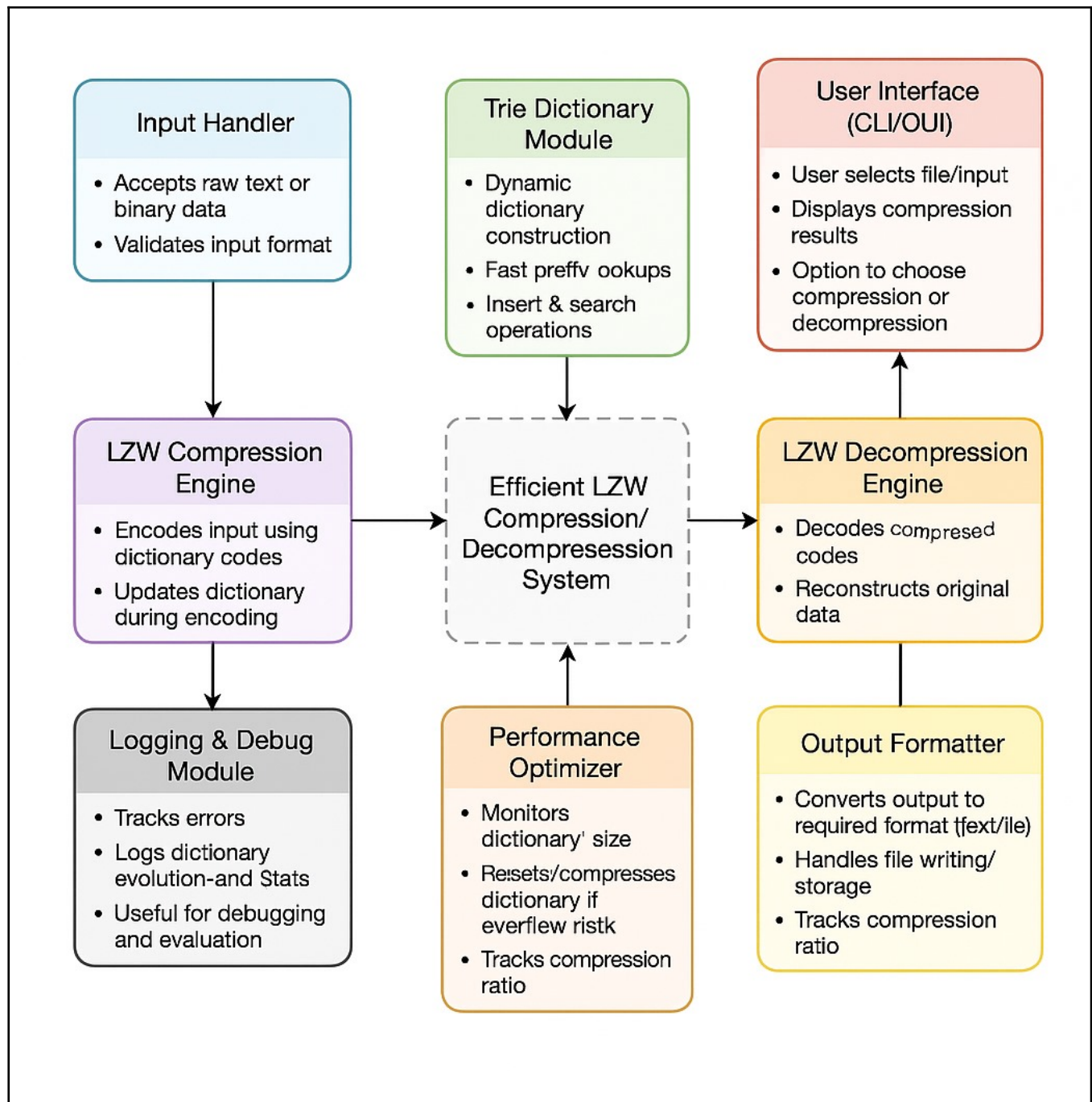
Project Abstract

This project implements the **Lempel-Ziv-Welch** (LZW) compression algorithm using a **trie (prefix tree)** for efficient dictionary storage and string matching. The goal is to achieve lossless data compression by dynamically building a dictionary of repeated substrings, replacing them with shorter codes. The trie optimizes pattern searches during compression, while decompression reconstructs the original data using code-to-string mappings. The implementation includes validation to ensure correctness.

Tasks Completed

Task Completed	Team Member
Implemented trie data structure with dynamic node insertion and lookup.	Sumit
Developed LZW compression logic using the trie for dictionary management.	Sumit
Implemented decompression with dictionary reconstruction.	Sumit
Added validation to ensure compressed data can be perfectly reconstructed.	Aviral
Tested with sample strings (e.g., "TOBEORNOTTOBEORTOBEORNOT").	Anubhav

Project Approach and Architecture



Challenges/Roadblocks

Trie Memory Management:

Issue: Initially used raw pointers, leading to memory leaks.

Solution: Switched to `unique_ptr` for automatic memory cleanup.

Decompression Edge Cases:

Issue: Handling codes not yet in the dictionary (e.g., "KABA" case in LZW).

Solution: Added conditional checks for code validity.

Performance Trade-offs:

Trie lookup is fast but uses more memory than a hash table.

Tasks Pending

Task Pending	Team Member (to complete the task)
Benchmarking: Compare performance against standard LZW implementations.	Aviral
File I/O: Extend to compress/decompress files (binary/text).	Sumit
Error Handling: Add robust exception handling for invalid inputs.	Anubhav & Sumit

Project Outcome/Deliverables

- Working LZW Compressor/Decompressor with trie-backed dictionary.
- Validation Tests proving lossless compression.
- Code Repository with documentation.

Progress Overview

Ahead of Schedule: Core compression/decompression logic is fully functional.
Behind Schedule: File I/O and benchmarking not yet started.
70% Complete: Pending tasks are non-critical enhancements.

Codebase Information

Repository: https://github.com/sumitkumarbittu/Compress
Branch: main
Key Commits: a1b2c3d: Added trie-based dictionary. d4e5f6g: Fixed decompression edge cases.

Testing and Validation Status

Test Type	Status (Pass/Fail)	Notes
Unit Tests: Verified compression/decompression cycles for sample strings.	Pass	
Validation: Confirmed original == decompressed(compress(original)) for all test cases.	Pass	

Deliverables Progress

Deliverable	Status
Trie-Based LZW Compressor	Completed
Decompressor	Completed
File I/O Support	Pending
Performance Benchmarks	Pending