MiniGit: A Custom Version Control System

Team Members

1. ESMAEL ABDUSEMED - ATE/9586/15

2. IMRAN TARIKU - ATE/9968/15

3. NEBIYU YONAS - ATE/7602/15

4. SURAFEL ALEBACHEW - ATE/3176/15

5. YARED BEKELE - ATE/3699/15

Data Structures Used

- Linked List: Used to manage the list of commits and snapshots of the project.

- Hash Map: For efficient file tracking and mapping filenames to contents.

- Vectors and Arrays: Used for dynamic storage of filenames, branches, and logs.

- Structs/Classes: Represent entities like Commit, Branch, and FileVersion.

Design Decisions

- CLI-Based Interface to simulate Git commands and enhance user learning.

- Branching & HEAD Management to allow switching between branches.

- Three-Way Merge Logic to handle merging with manual conflict management.

- Repository Class ties all modules into one cohesive unit.

Strengths

- Educational Value: Excellent learning tool to understand Git internals and version control systems.

- Clean CLI Interface: Simulates real Git commands and provides hands-on experience.

- Modularity: Separated components for commit handling, branching, and merging.

- Extendable Design: Easy to add features like GUI, remote repos, or better conflict handling.

- Lightweight: Local simulation without dependency on external libraries.

Limitations

- No Real File I/O Tracking: Files are simulated rather than tracked on disk.

- Limited Conflict Resolution: No built-in automated conflict resolver.

- Simplified Hashing: Basic hashing instead of robust cryptographic methods.

- Single-User Local Simulation: No support for multi-user or remote repos.

Future Improvements

- Improve Merge Conflict Handling with inline editing.

- Full File I/O Integration for real file tracking.

- GUI Interface to visualize commits and branches.

- Remote Repo Simulation with push/pull functionality.

- Unit Testing & Robust Error Handling for stability.