**Final Report – MiniGit: A Custom Version Control System**

Team Members from section 4

1,Hanan Seid............................UGR/5797/163.

2, Hilina Ashenafi.....................UGR/9691/164.

3, Nardos Haile….....................UGR/0410/16

**Project Overview**

We started from scratch using C++ to create MiniGit, a lightweight, basic version of Git.

It allows you to:

-Keep copies of your files (such as snapshots).

-Change between previous iterations

-Make branches

-Combine modifications from several branches.

-Examine the differences between the two versions.

-It operates in the terminal (command line) and teaches us about the inner workings of real Git.

**Main Features We Built**

Init - creates a folder called.minigit/ to hold everything.

Add - include the Stages files in the upcoming commit.

Commit - saves a timestamp and message along with a snapshot of every staged file.

Log - displays a history of previous transgressions.

Branch - makes a new pointer to a commit, which is similar to making a new timeline.

Checkout - loads an older commit or another branch into the working folder..

Merge - combinemerges files from two branches and notifies you when a file is modified.

Diff - displays the differences between two commits, line by line.

**What We learned**

-The inner workings of Git, including commits, branches, diffs, etc.

-How to use C++'s file handling, folders, and metadata.

-How to use data structures such as linked lists, trees, and hashes.

-How to create a command-line interface that is clean.

-How to divide tasks clearly and collaborate as a team.

## **How MiniGit Works Internally**

| **Component** | **Description** | **DSA Concepts Used** |
| --- | --- | --- |
| **.meta file** | Stores commit info (timestamp, message, parent ID) | File I/O |
| **Blob** | Stores the actual content of each file | Hashing (SHA-1 or custom) |
| **Commit** | Points to blobs and keeps metadata | Linked List / DAG |
| **HEAD** | Tracks the current branch or commit | Pointer logic |
| **Branches** | Map branch names to commit IDs | HashMap / File Map |
| **Log History** | Walks backwards from HEAD to show commit history | Linked List traversal |

## **Sample Output**

> .\init\_minigit.exe

Initialized empty MiniGit repository in .minigit/

> .\add\_file.exe

Enter filename to add: notes.txt

> .\commit\_file.exe

Enter commit message: First working commit

> .\log\_history.exe

----- Commit History -----

Commit: 1750249791

Message: first working committ

> .\create\_branch.exe

Enter new branch name:feature

> .\checkout.exe

Enter branch name to checkout:situation5

> .\merge.exe

Enter branch to merge into current: situation3

Merged new file: data.txt

> .\diff.exe

Enter commit1: 1750189532

Enter commit2: 1750193562

Differences:

- Message: First working commit

+ Message: Second commit

**Limitations**

-Does not allow remote pushing like GitHub does.

-Resolution of file conflicts is simple (just displays a message).

-The terminal must be used for all file operations.

**Future Improvements**

-Improved dispute resolution (e.g., interactive merge).

-GUI interface as opposed to just the command line.

-Include options for file renaming, deletion, and reversion.

**Conclusion**

We gained a thorough understanding of version control thanks to MiniGit. In addition to learning how to use Git, we also rebuilt a scaled-down version of it. We gained practical experience with file systems, software engineering, debugging, and teamwork through this project.