Report ----Naomi

Staging and Commit Logic

My part of the project was to carry out the system that stages file and commits them — basically how files are stored and tracked in different versions. I was responsible for the add and commit commands, analogous to what Git does.

I had no idea how version control systems internally worked when I started. But doing this assignment helped me understand how hashing, tracking, and storage all come together.

What I Did:

I implemented the add command such that when a file is staged, its contents are hashed and saved in a special directory called .minigit/objects. This avoids duplicated storage.

I implemented a simple system for maintaining what files are staged by utilizing an index file.

For commit, I put in place a system that bundles the current state into a commit object with message, timestamp, and pointer to old commit (identical to Git's history).

All commits and file revisions are saved, and the latest one is kept track of using a HEAD file.

I Learned how:

Hashing helps to uniquely identify the contents of a file.

Snapshots help us store file versions without copying unchanged data.

Commits are like checkpoints with full context of the project at that moment in time.

I also made progress in using C++ utilities like fstream, unordered\_map, and filesystem. The hardest part was making sure everything saved and loaded properly, but debugging it helped me understand file I/O a lot more.