SCM Lab Report

Your Name

Lalida Krairit Niracha Janavatara Sunidhi Pruthikosit

Throughout the lab exercise, we explored the fundamental Git capabilities using the command line. While some team members were familiar with Git through IDEs or GitHub's GUI, this lab exposed us to the benefits and challenges of using the command line for version control.

Key Learnings:

1. **Git Restore and Reset:**

   The lab initially outlined for us to use git restore, but it did not work as expected. Instead, git reset --hard along with a commit ID from git log was effective in reverting to previous commits.

2. **Conflict Management:**

   We encountered conflicts when merging branches and learned that resolving conflicts in the command line can be more cumbersome compared to IDEs, which streamline the process. After fixing conflicts, it's necessary to commit changes manually using git add . and git commit -m.

3. **Authentication Quirks:**

   A crucial aspect we learned was the importance of the link used in git clone, as it impacts authentication. Using the wrong link resulted in issues when pushing to the repository, which was resolved through careful configuration.

4. **Branching and Parallel Development:**

   We explored branching by creating and switching branches locally with git checkout. Team A worked on a separate branch and later merged changes into the main branch using git merge. We discovered that merging non-conflicting branches was smooth, but merging conflicting branches required manual intervention.

5. **Automatic Merging and Local Commits:**

   We saw that branches allow independent streams of commits, making them ideal for parallel development. However, merging conflicting branches was a different story. Git flagged conflicts within the TempREADME.md file, and we had to manually resolve them by adjusting lines and removing markers.

6. **Best Practices:**

   We developed several best practices for managing our codebase:

   - Always create a new branch when modifying code to avoid conflicts in the main branch.
   - Frequently pull updates from the branch you are working on.
   - Write informative commit messages and ensure that each push is meaningful.
   - Avoid committing unnecessary files or directories that could clutter the repository.

This lab has enhanced our understanding of Git and its command-line capabilities, particularly in managing conflicts and parallel development. We are now more confident in using Git for version control and appreciate the detailed messages and control it offers compared to GUI systems.