**LAB** ACTION **PLAN FOR WEEK 2**

**Objective:**

To demonstrate the ability to manage and share a complete multi-folder software project containing image files, Java files, web pages, and a pom.xml file—using Git and GitHub. The task involves initializing a local Git repository, committing all project files, and pushing the project to both a **public** and a **private** remote repository on GitHub. This objective aims to develop student practical skills in **version control, and repository configuration** while reinforcing best practices for maintaining and deploying code in different access-controlled environments.

**Scenario based questions on basic git commands**

1. You made changes to one file in the above project but haven’t staged them yet. You realize they were a mistake. What Git command will you use to discard the local changes?

2. You accidentally ran git add file1.txt (note instead of file.txt consider one file from above project), but you’re not ready to commit yet. How do you remove it from the staging area without losing the changes?

3. You made a commit but typed the wrong commit message. You haven’t pushed it yet. How do you fix it?

4. You want to view the commit history of the current branch in a readable format. What Git command should you use?

5. After cloning a repo, how can you set your name and email globally for all Git repositories?

6. You’ve made some edits to your files but haven’t staged them. How can you view the changes?

7. You’re on the main branch but need to switch to feature/login. What command do you

8. You deleted the feature-ui branch by mistake. You haven’t pushed the deletion. How

9. You’ve made some commits locally and now want to upload them to the remote repository. What do you run?

10. How can you fetch the latest changes from the remote repository without merging them automatically?

11. You’re on the main branch and want to start working on a new feature called search-filter. What command do you use?

12. You committed a file containing an API key and want to completely remove it from the repo's history. What should you do?

13. You want to see all the branches that exist both locally and on the remote. What

14. You’re on main and want to merge the completed feature/signup branch into it. What command do you use?

15. You tried to merge two branches and Git reported a conflict in app.js. What are the general steps to resolve it?

16. You don’t want Git to track changes to .log files or node\_modules/. How do you achieve this?

17. You're investigating a bug and want to know who last changed line 25 in script.py. What command do you use?

18. You’re in the middle of working on a file but need to switch branches quickly. What do you do to save your work?

19. You previously ran git stash and now want to restore those changes. What command do you use?

20. The feature/test branch is no longer needed. How do you delete it locally?

21. You just merged the feature-ui branch into main. You want to clean up your local branches. How do you safely delete feature-ui?

22. You created a branch feature-experiment, made some changes, but now realize the code is no longer needed. You want to delete it, even though it hasn’t been merged. What command will you use?

23. You're currently on the feature-login branch and want to delete feature-ui. What must you ensure before running the delete command?

24. You want to delete bugfix-footer, but you're unsure if it's been merged. What should you do before deleting it?

25. You finished working on feature-a, feature-b, and feature-c. All have been merged into main. How do you delete them in one command?

**Scenario based questions on working with remote repository using Git commands**

### 1. Specify the git command when you're starting work on a project and need to clone a remote repository to your local machine.

### 2. Specify the git command if you want to see the remotes that are connected to your local repository.

### 3. Specify the git command if you want to add a new remote repository to your local repository.

### 4. Specify the git command to remove a remote repository from your local configuration:

### 5. Specify the git command if you want to rename an existing remote:

### 6. Specify the git command to fetch updates from the remote repository but not merge them into your local branch.

### 7. Specify the git command to pull the latest changes from the remote repository and merge them into your local branch.

### 8. Specify the git command to push your local commits to a remote repository.

### 9. Specify the git command to when pushing for the first time and want to set the remote branch you are pushing to.

### 10. Specify the git command to change the URL of a remote (e.g., after changing the remote repository address).

### 11. Specify the git command if you want to list all branches on the remote repository.

### 12. Specify the git command if a branch was deleted on the remote but still shows up locally.

### 13. Specify the git command to fetch a specific branch from a remote.

### 14. Specify the git command if you want to see detailed information about a remote repository.

### 15. Specify the git command if you want to rebase your local branch onto a remote branch (this can be useful to keep your history linear)

### Answers:

1. git checkout -- <filename>
2. git reset HEAD <file1.txt>
3. git commit --amend
4. git log --oneline
5. git config --global user.name "Your Name" and git config --global user.email "you@example.com"
6. git diff
7. git switch feature/login *(or)* git checkout feature/login
8. git checkout -b feature-ui *(if the branch existed locally before deletion and reflog/history still exists)*
9. git push
10. git fetch
11. git switch -c search-filter *(or)* git checkout -b search-filter
12. git filter-repo --path <filename> --invert-paths
13. git branch -a
14. git merge feature/signup
15. Open the file, manually resolve the conflict, run git add app.js, then git commit
16. Add them to .gitignore
17. git blame script.py
18. git stash
19. git stash pop
20. git branch -d feature/test
21. git branch -d feature-ui
22. git branch -D feature-experiment
23. Ensure you're **not on** feature-ui; switch to another branch first.
24. Run git branch --merged to check if it’s merged before deleting.
25. git branch -d feature-a feature-b feature-c

**Scenario-Based Remote Git Commands**

1. git clone <repo-url>
2. git remote -v
3. git remote add <name> <url>
4. git remote remove <name>
5. git remote rename <old-name> <new-name>
6. git fetch
7. git pull
8. git push
9. git push -u origin <branch-name>
10. git remote set-url <name> <new-url>
11. git branch -r
12. git remote prune origin
13. git fetch origin <branch-name>:<branch-name>
14. git remote show <name>
15. git pull --rebase origin <branch-name>

### Conclusion:

Successfully uploading a multi-folder project to both public and private GitHub repositories demonstrates student’s practical understanding of **core Git commands** and **remote repository management**. By handling diverse file types (e.g., images, Java files, web pages, and pom.xml) and pushing them to different remote repositories, students show proficiency in:

* **Initializing and configuring Git repositories**
* **Staging, committing, and managing version history**
* **Connecting to and working with multiple remote repositories**
* **Handling Gitignore and tracking large files appropriately**
* **Applying best practices for public vs. private repository usage**

Completing these tasks equips students with essential version control skills and the ability to collaborate securely and professionally. It also reflects their readiness to contribute effectively to real-world software projects where source code management, team coordination, and secure code sharing are vital.