**Placement Empowerment Program**

***Cloud Computing and DevOps Centre***

**Set Up Git Branching**

Create a new branch in your Git repository for testing. Add a new feature and merge it

Name: **Priyadharshini S** Department: **ADS**



**Introduction**

In this task, explore Git branching, a fundamental feature that allows developers to work on new features independently without affecting the main codebase. I will create a new branch named testing to implement a feature, commit the changes, and then merge them back into the main branch. This process involves checking the current branch, making modifications, and handling potential merge conflicts. After merging, we verify the changes through commit history and file existence. This approach ensures a structured and efficient workflow in collaborative development.

**Overview**

1. Navigate to your Git repository using the terminal.
2. Check the current branch using git branch to ensure you are on the correct branch (usually main).
3. Create a new branch named testing and switch to it using git checkout -b testing.
4. Make modifications by adding or updating files (e.g., implementing a new feature).
5. Stage the changes with git add and commit them using git commit.
6. Switch back to the main branch and merge the testing branch using git merge.
7. Verify the merge by checking the commit history with git log and ensuring the changes exist in the main branch.

**Objectives**

By the end of this POC, you will:

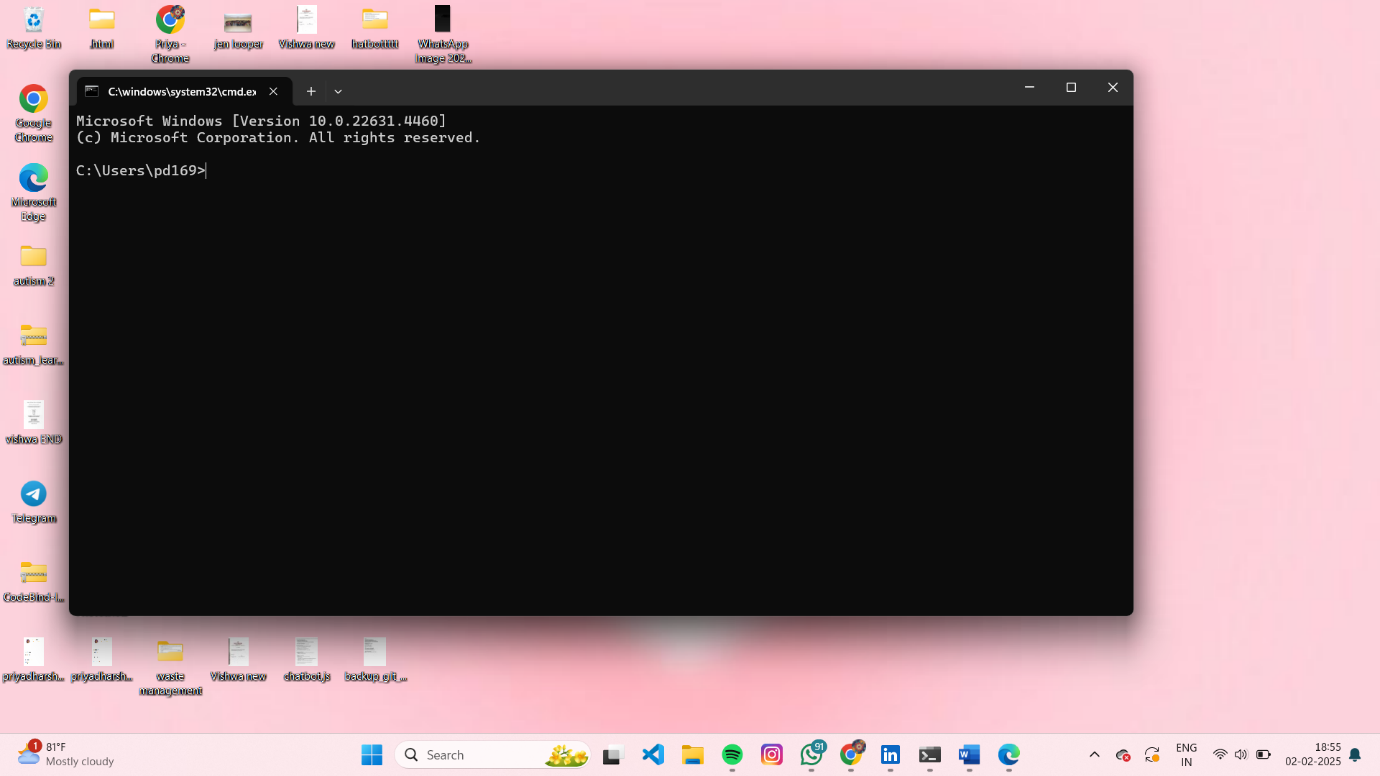
1. Understand and implement **Git branching** to manage feature development separately from the main codebase.
2. Learn how to **create a new branch**, make changes, and commit updates.
3. Merge the new branch back into the **main branch** efficiently.
4. Handle **merge conflicts** if they arise during integration.
5. Verify successful integration by checking **commit history and file existence**.
6. Navigate **Git history** to track changes and maintain proper version control.
7. Improve **collaborative development skills** using Git.

**Step-by-Step Overview**

Step 1:

**Navigate to Your Git Repository**

Open the terminal and navigate to the folder where your Git repository is located using the cd command.

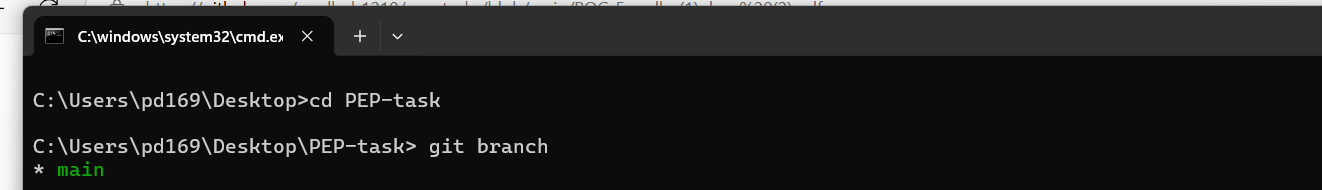


Step 2

**Check the Current Branch**

Before creating a new branch, confirm which branch you are currently on (typically main) by running:

git branch

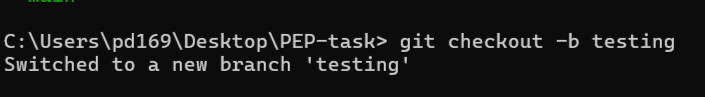


Step 3

**Create a New Branch**

Create a new branch named testing and switch to it using:

git checkout -b testing

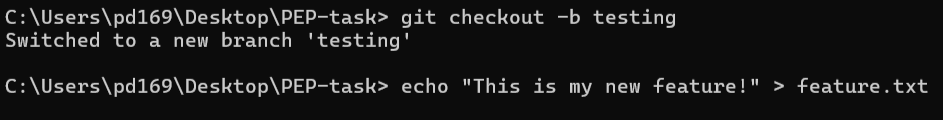


Step 4

**Make Changes**

Once on the testing branch, add a new feature by creating or modifying files. For example, to add a file named feature.txt, run

echo "This is my new feature!" > feature.txt



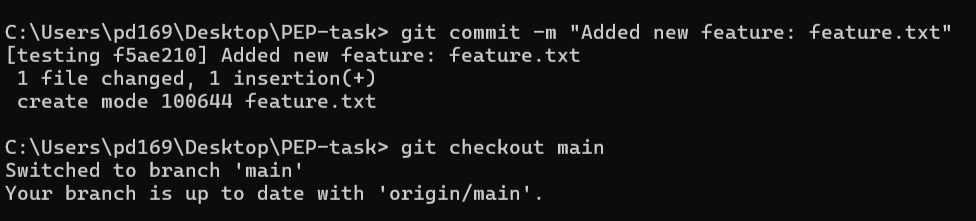
Step 5

**Stage and Commit Changes**

After making changes, stage and commit them using:

git add feature.txt

git commit -m "Added new feature: feature.txt"



Step 6

**Switch Back to the Main Branch**

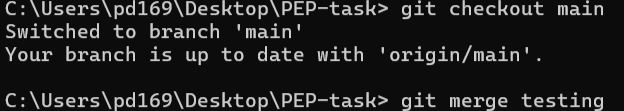
Once the changes are committed, switch back to the main branch using:

git checkout main

Step 7

**Merge the testing Branch into main**

Now that you're back on the main branch, merge the changes from the testing branch using:

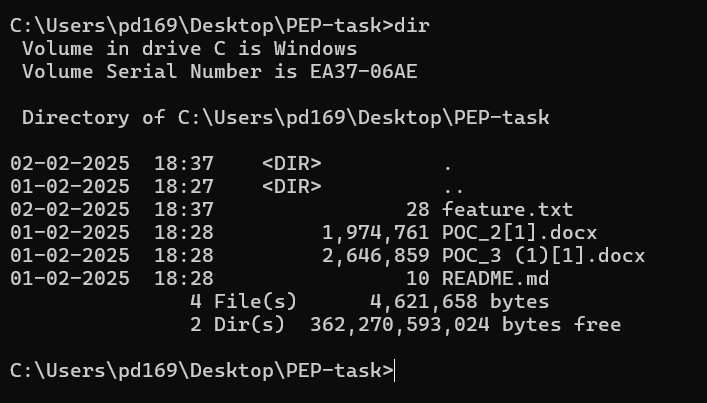


Step 8

**Verify the Merge**

To confirm that the merge was successful, list the files in the current directory and check for the presence of feature.txt:

Dir



**Expected Outcome**

By completing this PoC of setting up a local Git repository, you will:

1. Successfully initialize a Git repository in your local static website folder.

2. Track changes made to your website files (HTML, CSS, etc.) using Git version control.

3. Understand the basic Git commands (git init, git add, git commit) for version control.

4. Commit your changes locally with a descriptive commit message.

5. Gain hands-on experience with Git and how it helps manage and track website file changes.