**Name : Reena Qureshi**

**Sem : V-B**

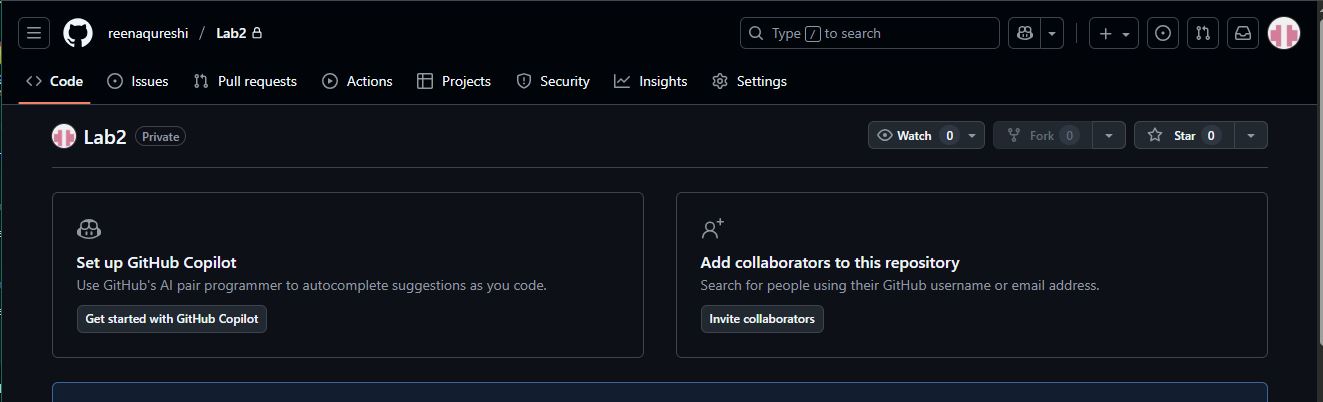
**Reg No : 2033-BSE-052**

**Lab 2**

**Task 1: Create Private GitHub Repository**

**Create a new private repository named**Lab2**on GitHub.**

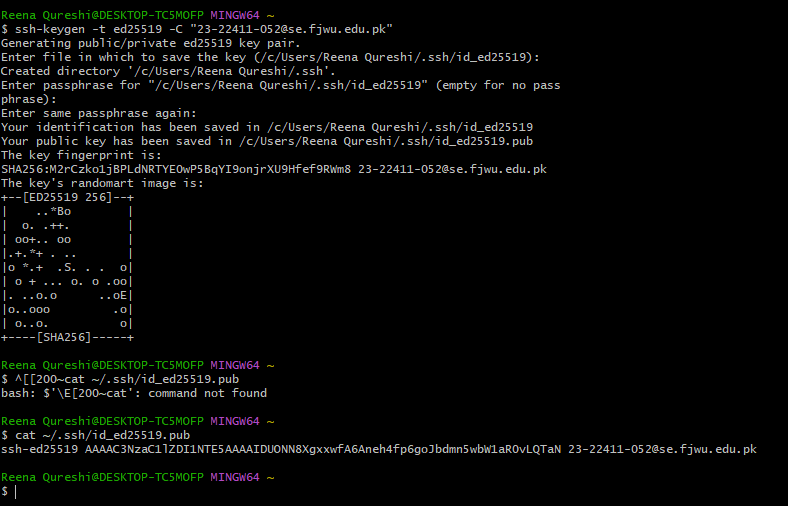
Take a screenshot of your repo settings showing it's private.

****

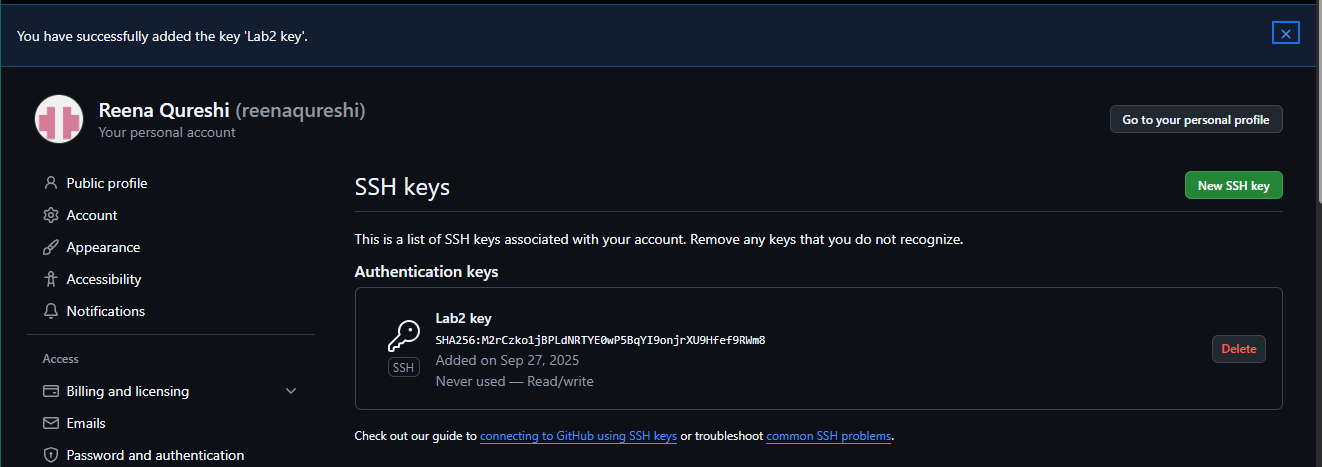
**Task 2: Connect Repository via SSH**

**Generate a new SSH key using PowerShell:**

ssh-keygen -t ed25519 -C "your\_email@example.com"

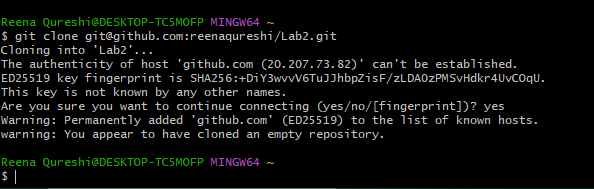
****

**2.Add your SSH public key to GitHub (Settings > SSH and GPG keys).**

****

Clone your Lab2 repo using SSH.

git clone git@github.com:<yourusername>/Lab2.git

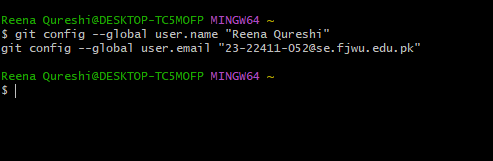
****

**Task 3: Configure Git Username and Email**

**Set up your Git identity (this ensures all commits are linked to you):**

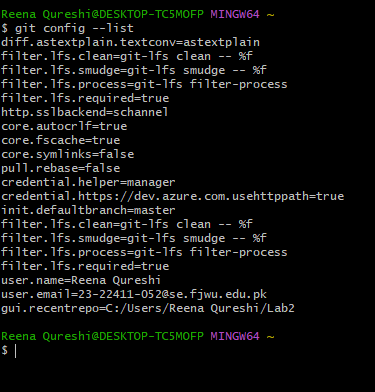
git config --global user.name "Your Name"

git config --global user.email [your\_email@example.com](mailto:your_email@example.com)

****

**Verify your configuration:**

git config --list



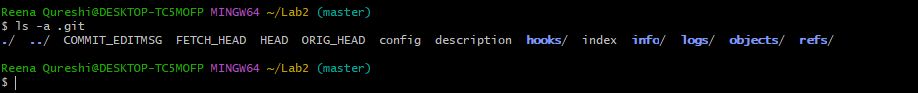
**Task 4: Explore the .git Folder**

Navigate into your cloned repository folder.

Show hidden files and locate the .git directory.

Explore what’s inside using:

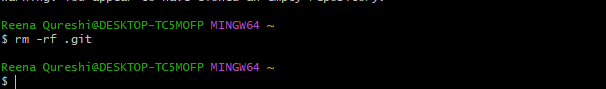
**ls -a .git**



**Task 5: Local Repository Management**

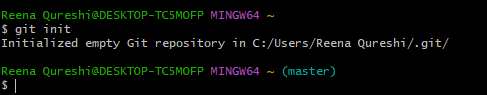
**Delete the existing .git folder from your cloned repo using Git Bash:**

**rm -rf .git**



Re-initialize the local git repository

**git init**

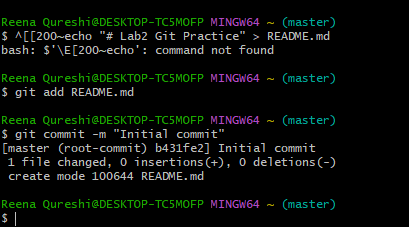


Add a file named README.md and commit it:

**echo "# Lab2 Git Practice" > README.md**

**git add README.md**

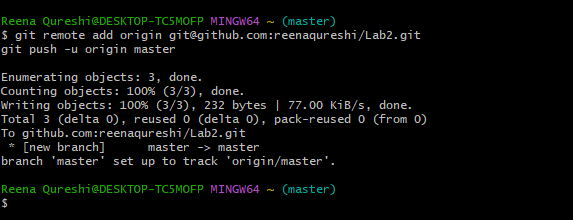
**git commit -m "Initial commit"**



**Connect your local repo to GitHub and push:**

**git remote add origin git@github.com:<yourusername>/Lab2.git**

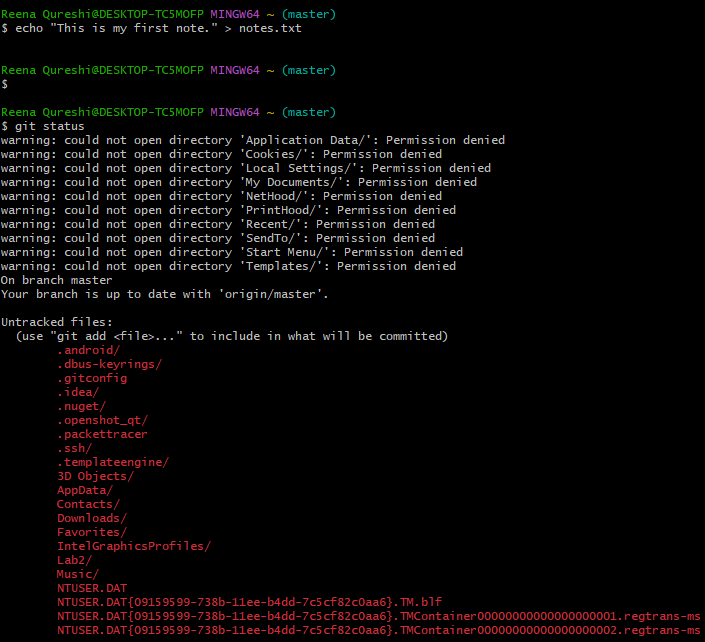
**git push -u origin main**



**Task 6: File Status & Staging**

**Create a new file notes.txt and write a note.**

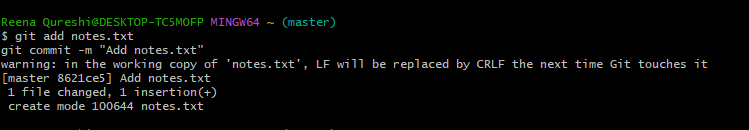
**Check status**



**Stage and commit:**

**git add notes.txt**

**git commit -m "Add notes.txt"**

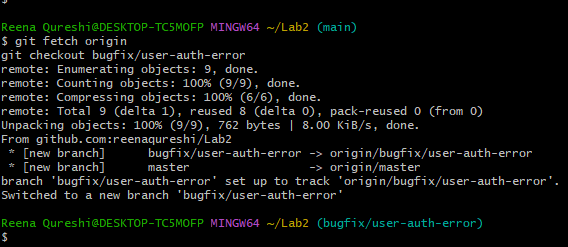
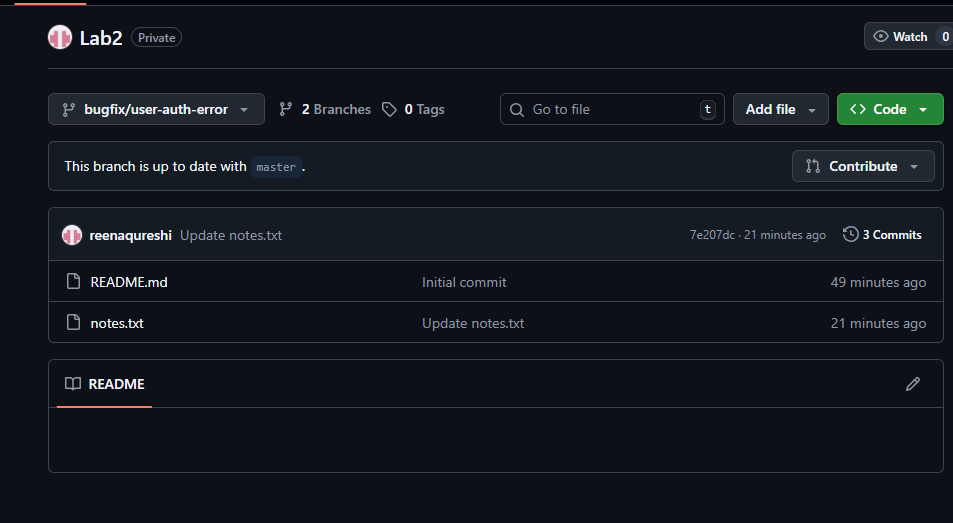


**Task 7: Branch Creation Using GitHub GUI**

**On GitHub (web interface), create a branch named**bugfix/user-auth-error**.**

**Pull the branch to your local repository to sync.**

git pull origin bugfix/user-auth-error



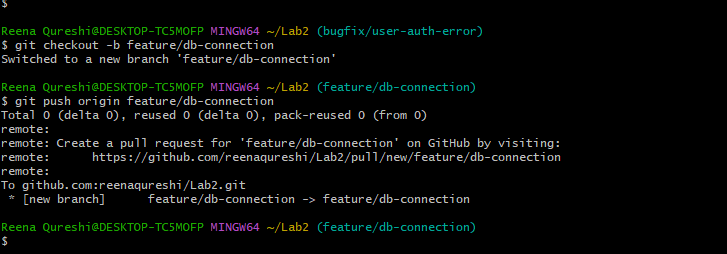
**Task 8: Branch Creation and Push Using Git Bash**

**Create a branch named feature/db-connection using Git Bash:**

**git checkout -b feature/db-connection**

**Push the branch to the remote repository:**

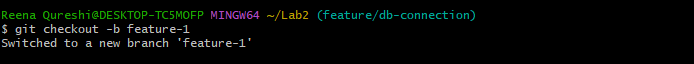
**git push origin feature/db-connection**



**Task 9: Branching & Merging**

**Create and switch to a branch feature-1:**

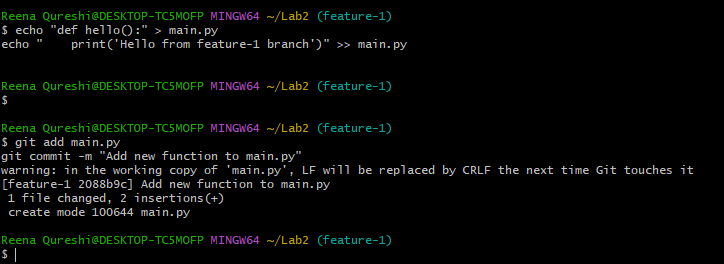
**git checkout -b feature-1**



**Modify main.py (add a function) and commit.**

**git add main.py**

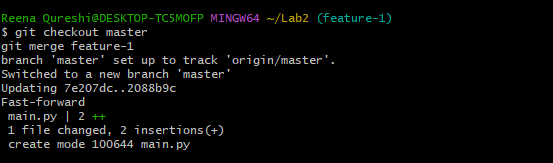
**git commit -m "Add new function to main.py"**



Switch back to main and merge:

**git checkout main**

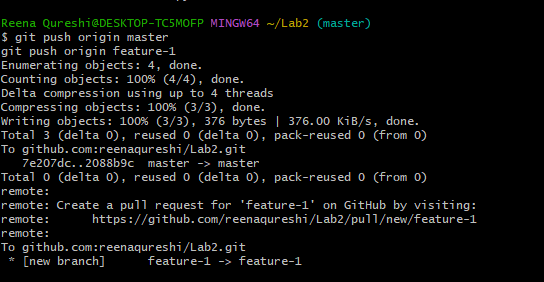
**git merge feature-1**



Push all branches:

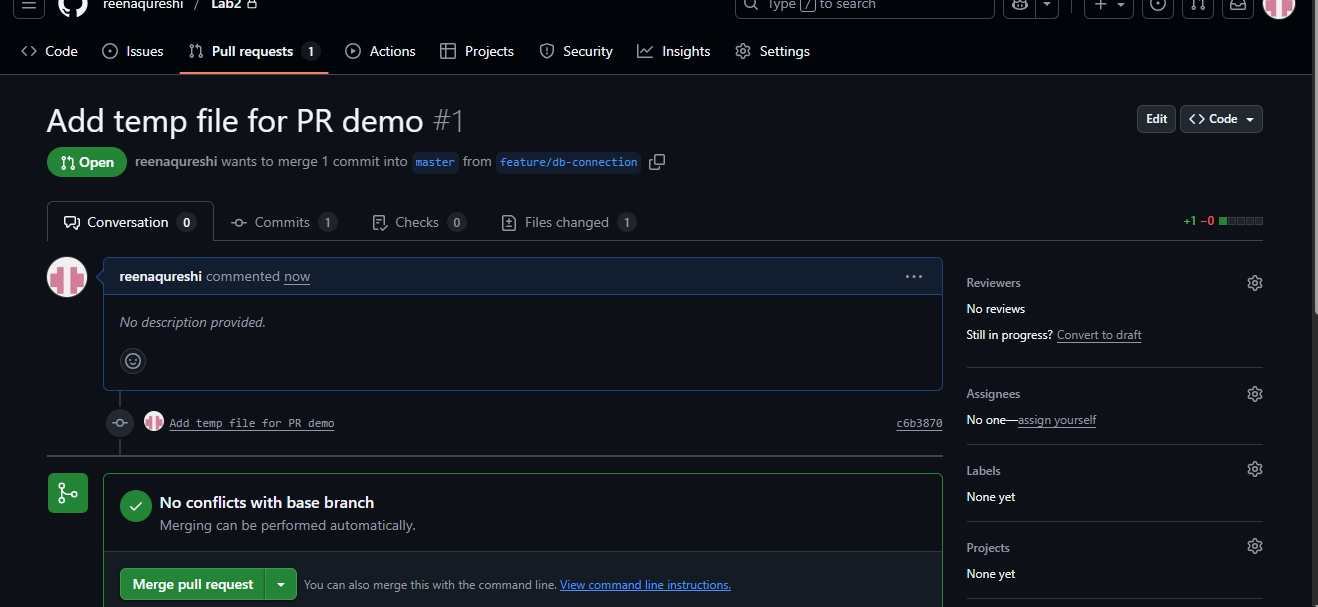
**git push origin main**

**git push origin feature-1**

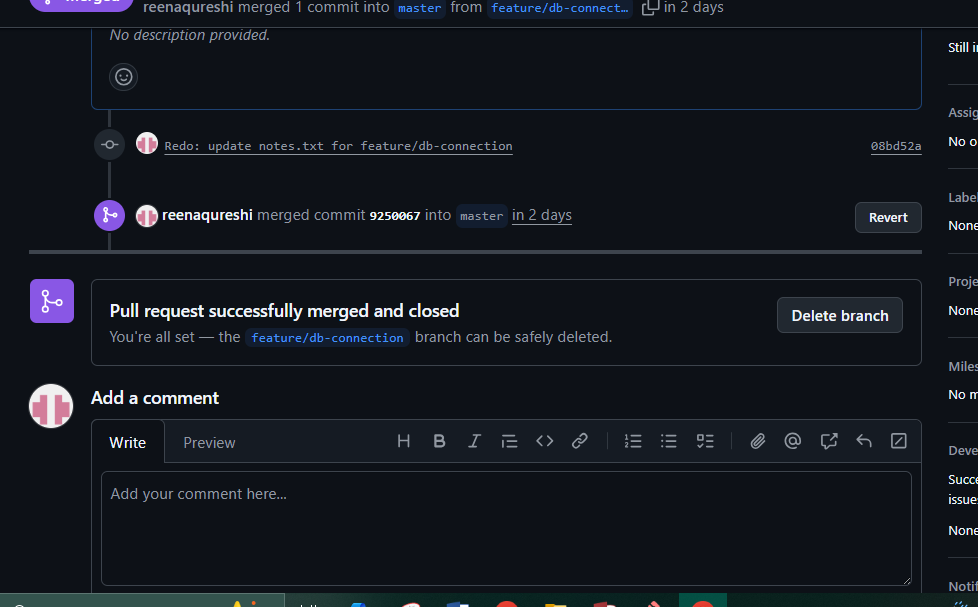


**Task 10: Pull Request and Branch Review (GitHub GUI)**

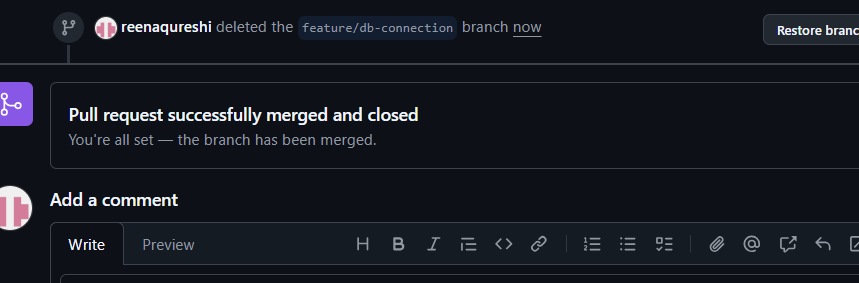
**On GitHub, create a Pull Request from the branch feature/db-connection to main.**



**Review the Pull Request and merge it using the GitHub GUI.**



After merging, delete the feature/db-connection branch using the GitHub GUI.



**Task 11: Detailed Branch Strategy (Develop/Staging)**

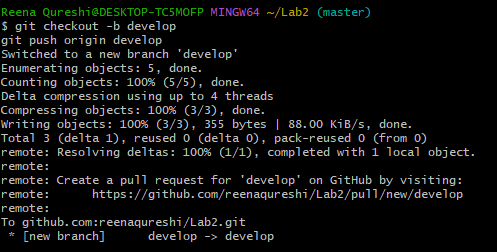
**Create the following branches** to simulate a professional branching strategy:

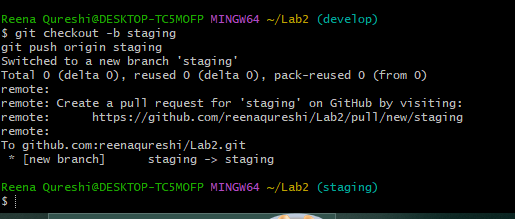
develop

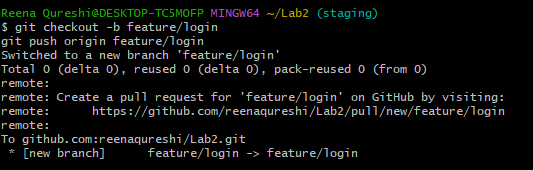
staging

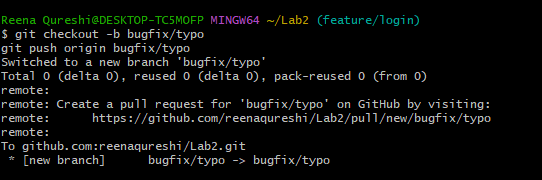
feature/\*

bugfix/\*

****

****

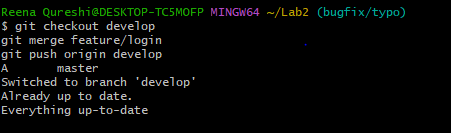
****

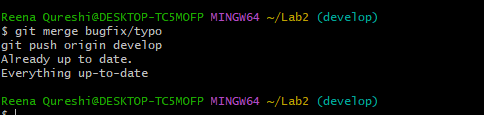
****

**Example Workflow:**

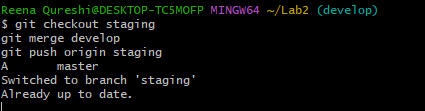
Developers work on feature/\* and bugfix/\* branches.

Merge completed work into develop.

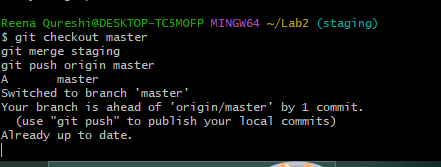
****

****

**Merge develop into staging for testing.**

****

**Finally, merge staging into main (production).**

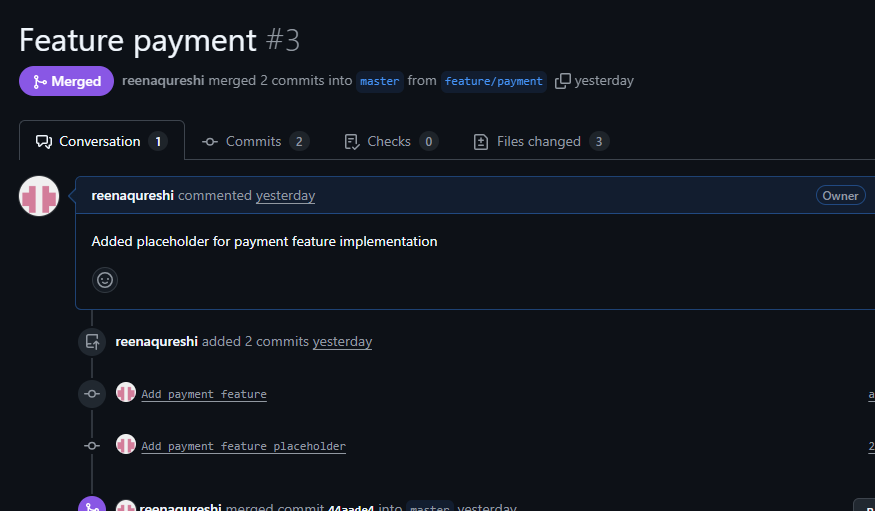
****

**Task 12: Code Review Workflow**

**Create a Pull Request (PR) / Merge Request (MR):**

From a feature branch into main.

Add a clear title and description summarizing the changes.

****

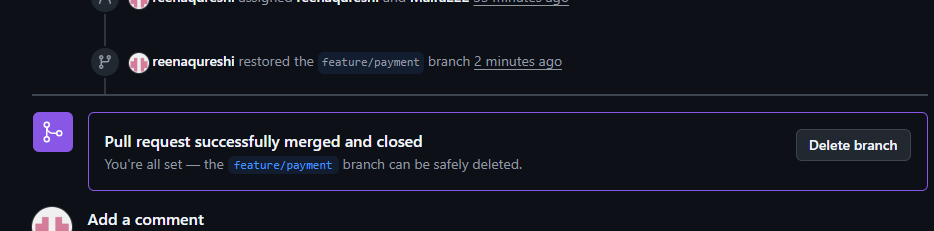
Assign a reviewer (teammate or second account).

****

Reviewer Actions

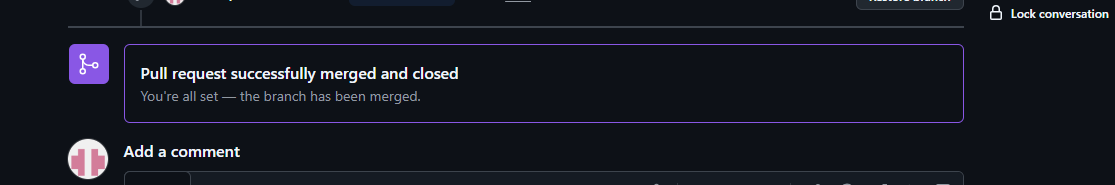
##### 

Merge the PR:

****

**Task 13: Branch Cleanup Best Practices**

Delete Remote Branch After Merge:

****

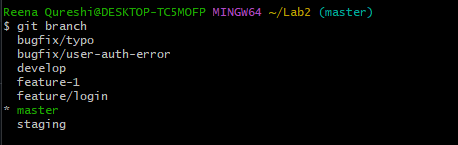
**Update Local Repository:**

**git checkout main**

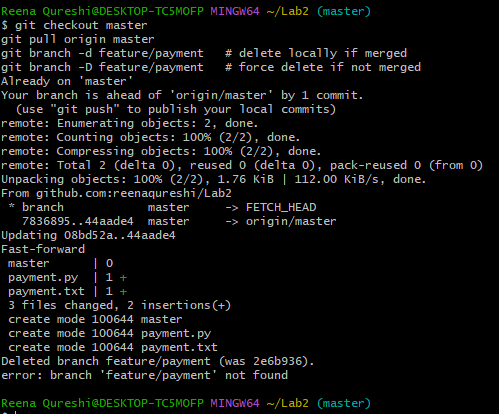
**git pull origin main**

**git branch -d <branch-name>**

**git branch**

****

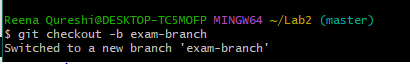
**Branch “feature/payment deleted”**

****

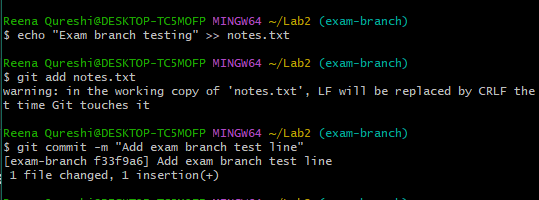
Exam Evaluation Questions

**Advanced Branching & Merge Verification**

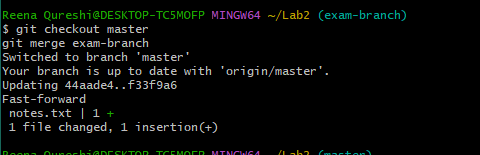
**Create a new branch in your repository**

****

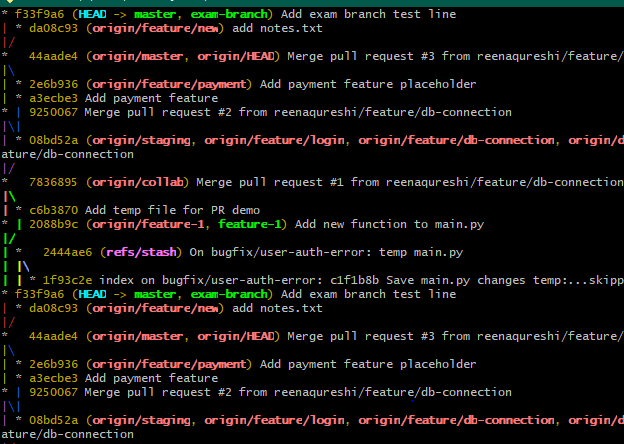
**Make a small change in a file and commit it.**

****

**Merge this branch back into the main branch**

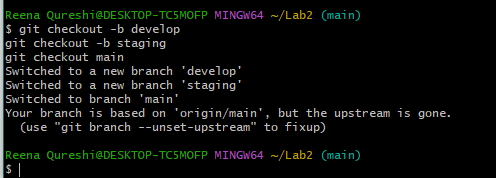
****

**Show the history of commits in a way that verifies the merge**

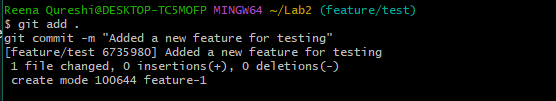
****

**Multi-Stage Workflow Simulation**

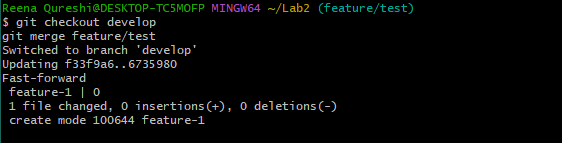
**Set up a branching workflow with three branches: main, develop, and staging.**

****

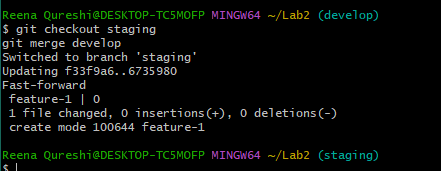
**Create a feature branch from develop, make changes, and commit them**

****

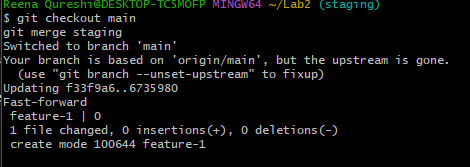
**Merge the feature branch into develop**

****

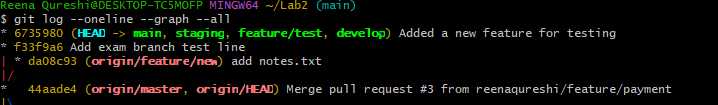
**Merge develop into staging**

****

**Merge staging into main**

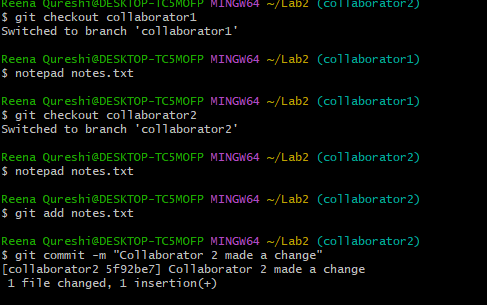
****

**Provide proof that each stage contains the updated changes before it reaches main**

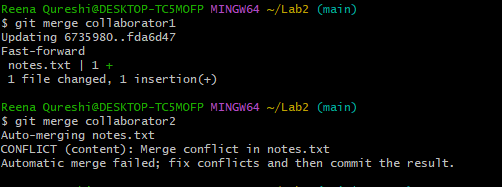
****

**Collaboration & Conflict Resolution**

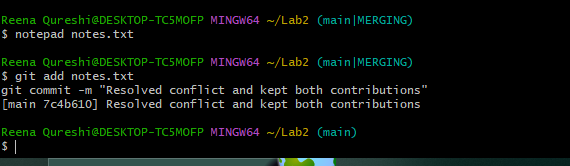
**Work with a collaborator: both contributors should modify the same file but in separate branches**

****

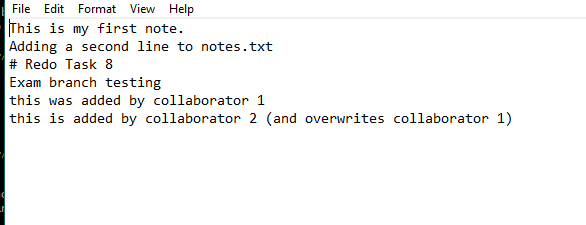
**Attempt to merge the branches and capture the conflict.**

****

**Resolve the conflict so that both contributions are preserved**

****

**Provide evidence that the final version of the file contains both collaborators’ changes**

****