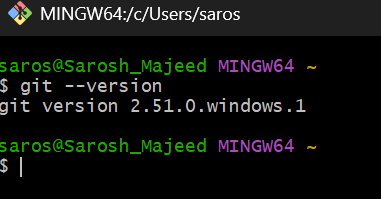
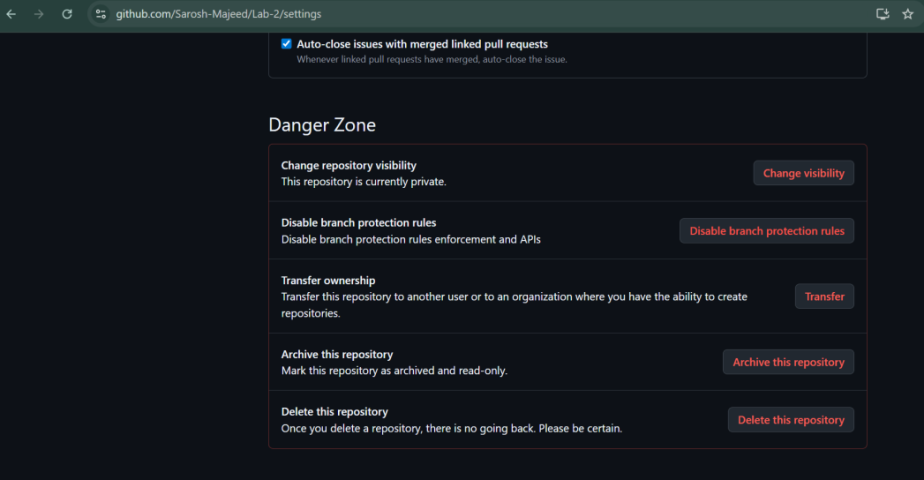
***Cloud Computing (Lab 2)***

***Name : Sarosh Majeed***

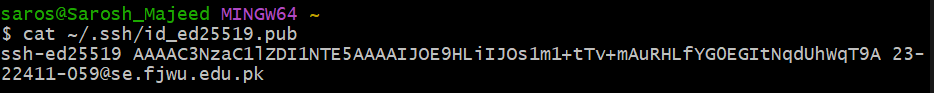
***Reg no : 2023-BSE-059***

**Task 0: Git Installation (already done)**

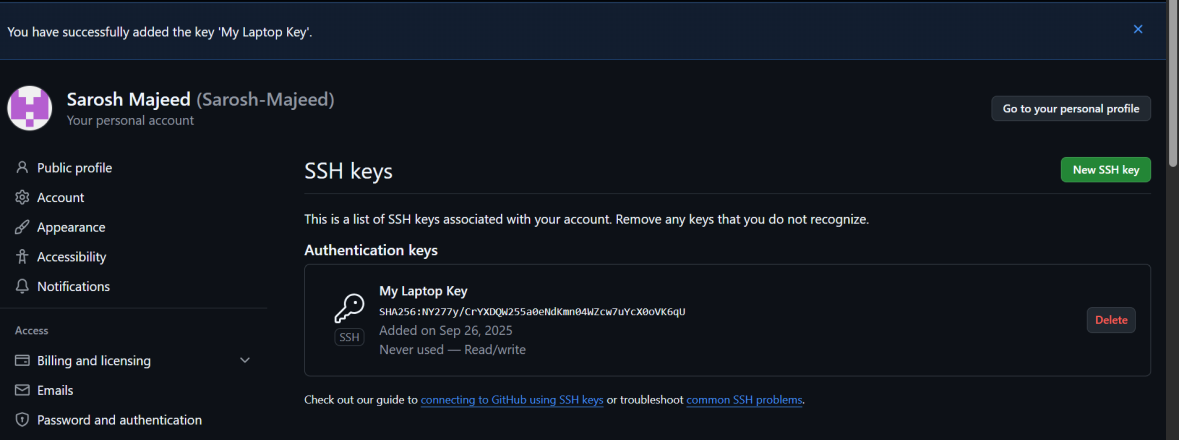


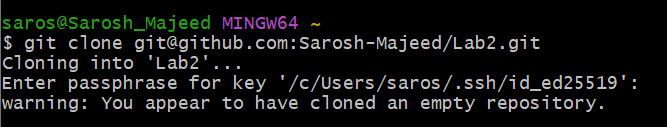
**Task 1: Create Private GitHub Repository**

**Task 2: Connect Repository via SSH**

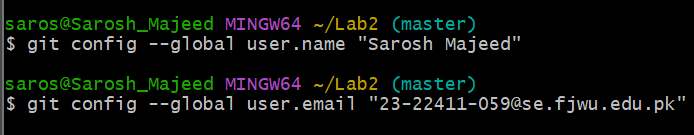
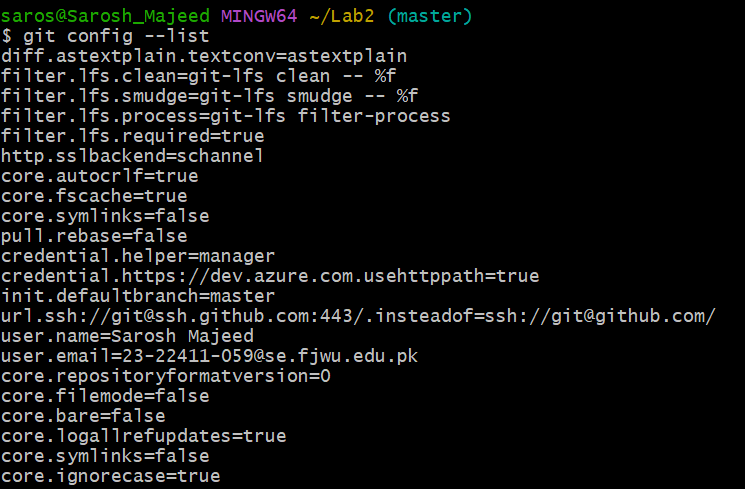
****

1. Go to **GitHub → Settings → SSH and GPG keys → New SSH key**.

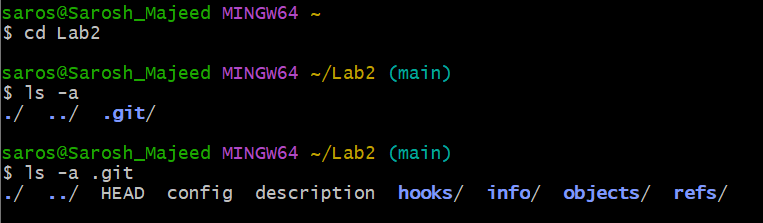


1.  Clone repo using SSH:

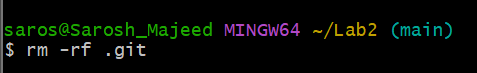
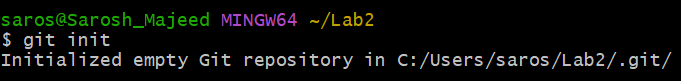
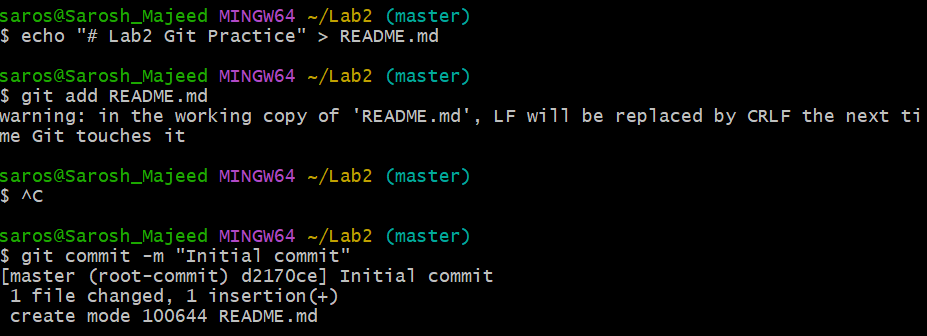
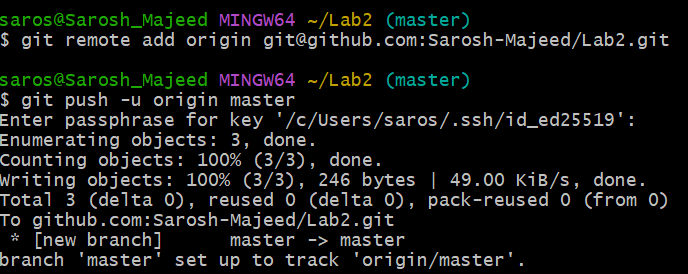
**Task 3: Configure Git Username and Email**

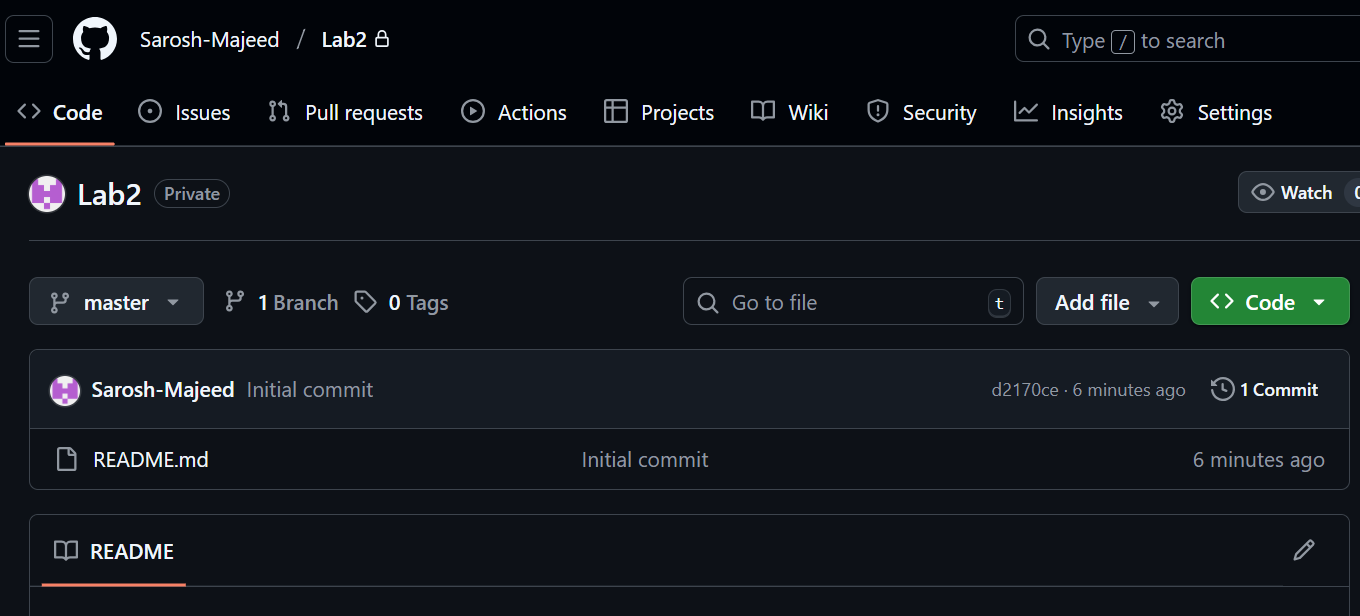
1. Set global config:
2. Verify:

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

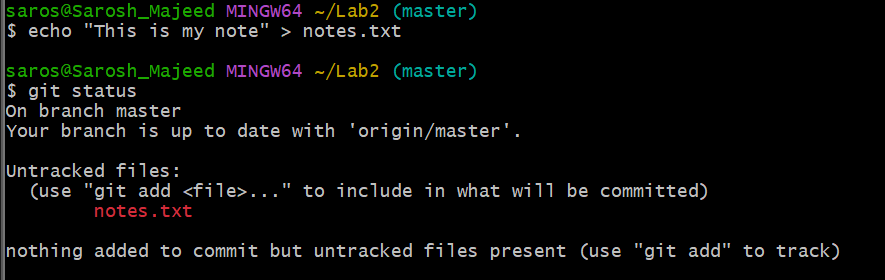
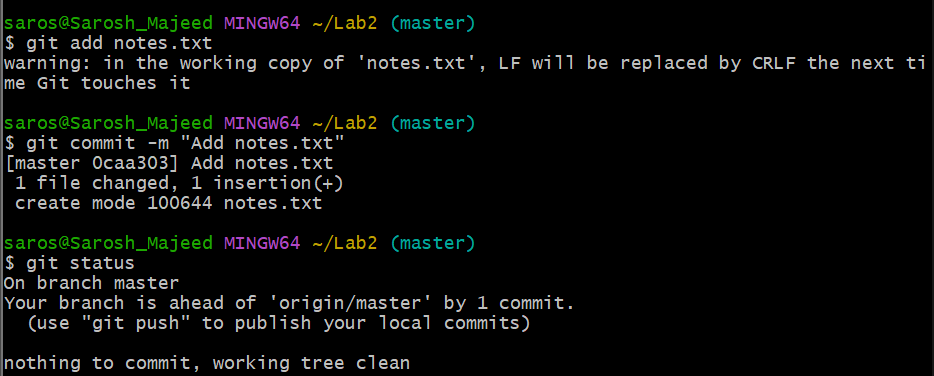
1. Go into your cloned repo
2. Explore inside

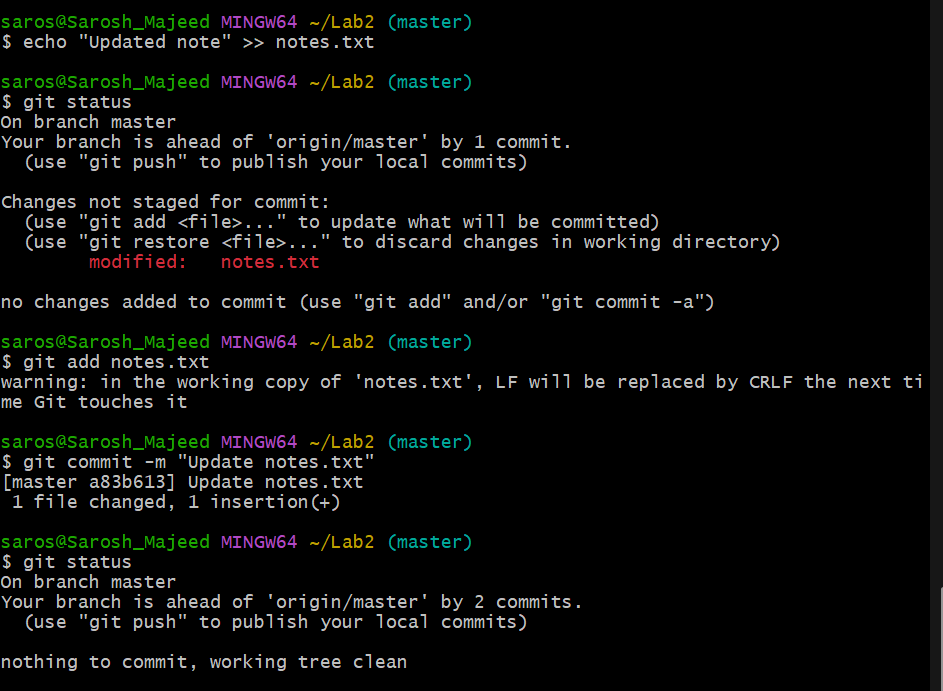
**Task 5: Local Repository Management**

1. Delete .git
2. Re-initialize
3. Add README.md
4. Connect remote & push:

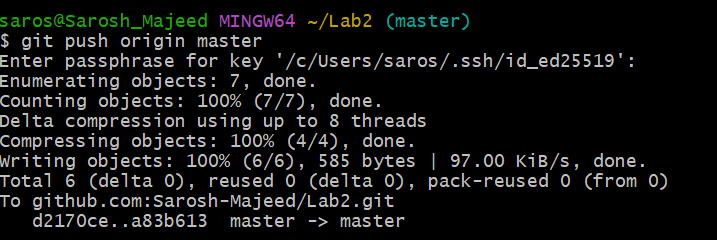
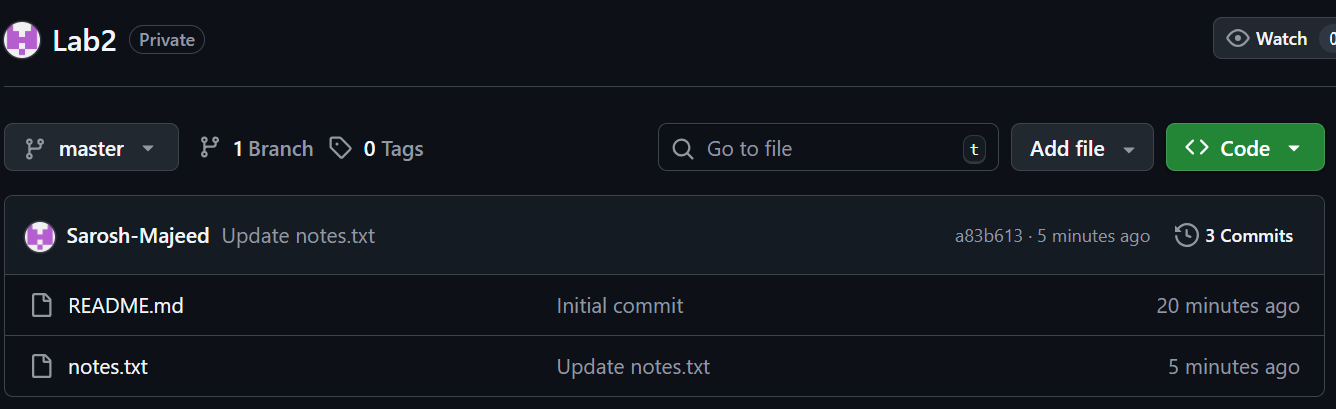
****

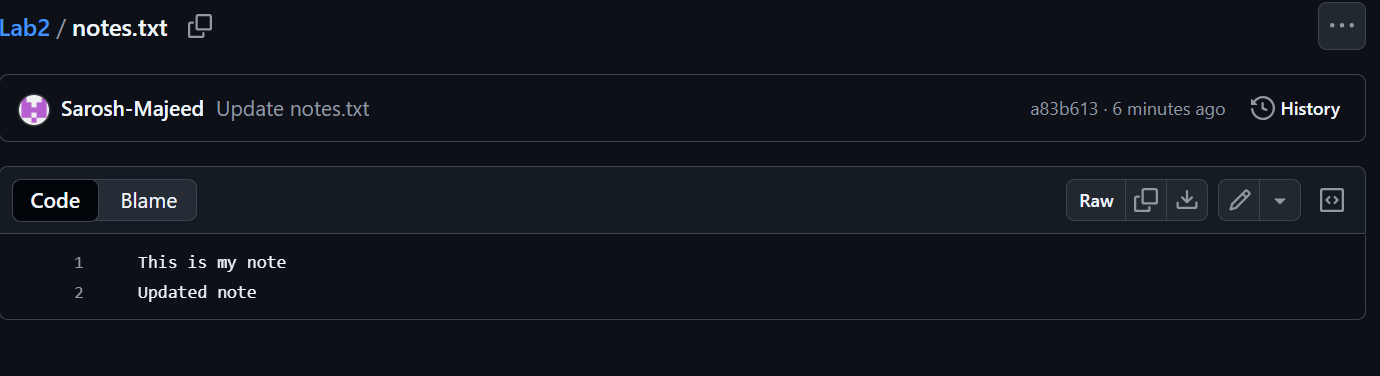
**Task 6: File Status & Staging**

1. Create a file
2. Stage & commit



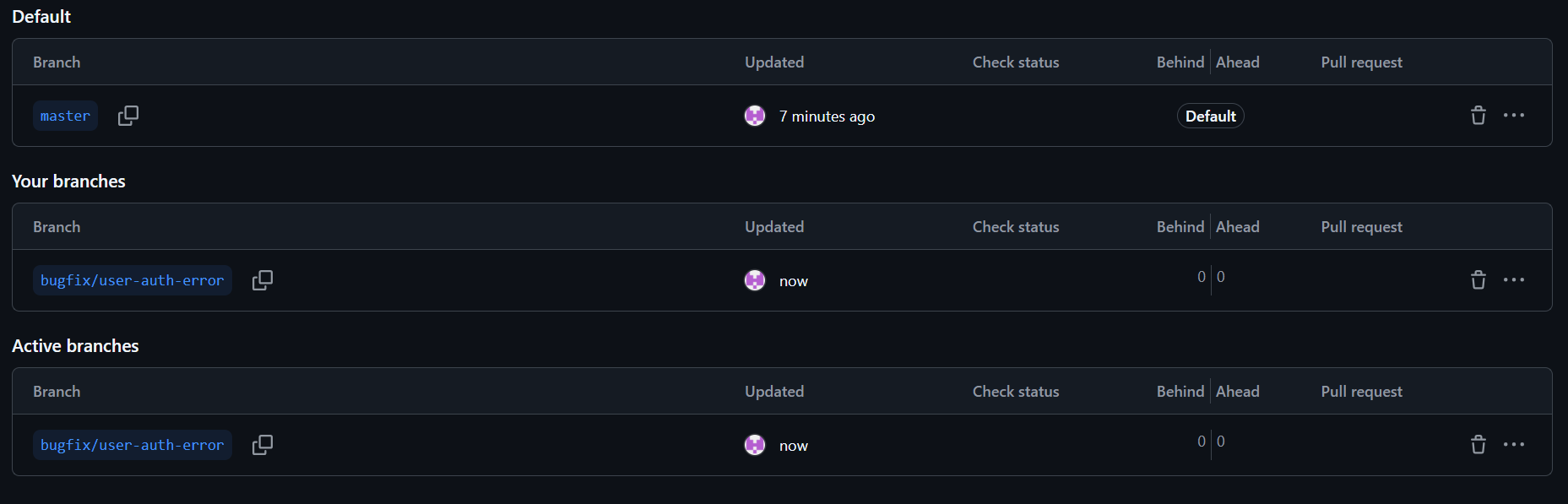
1. Edit file
2. Push notes

****

****

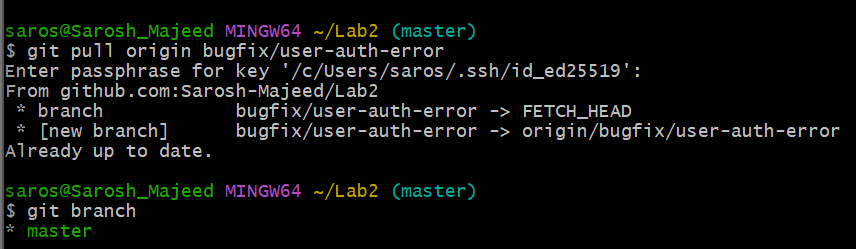
**Task 7: Branch Creation (GitHub GUI)**

1. Create branch **bugfix/user-auth-error**.

📸 GUI screenshot → bugfix\_branch\_gui.png.

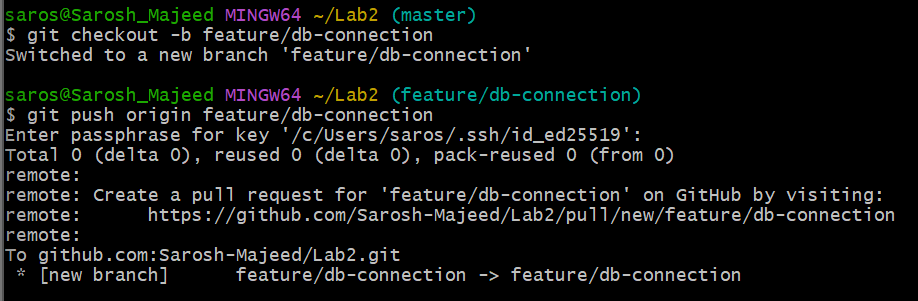
1. Pull branch to local

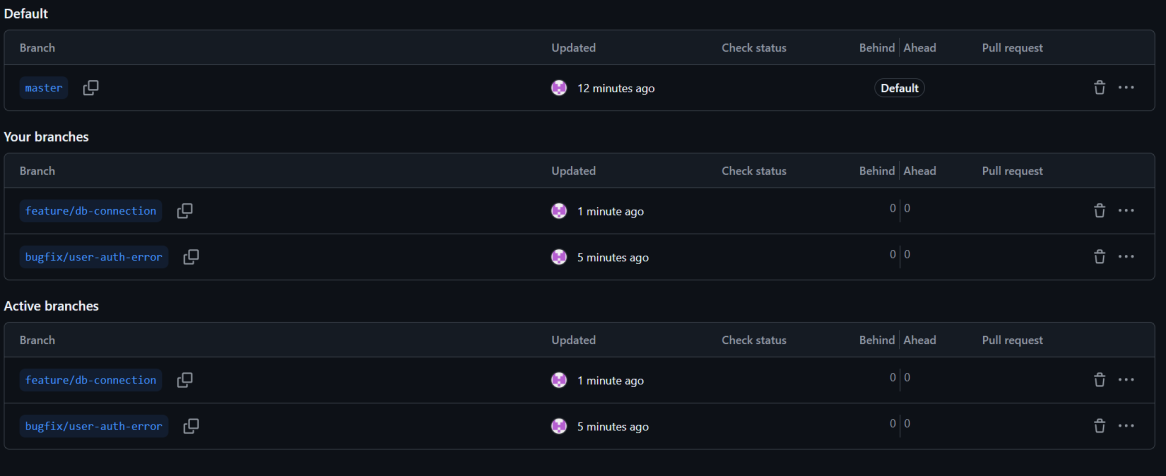
📸 Local screenshot → bugfix\_branch\_local.png.



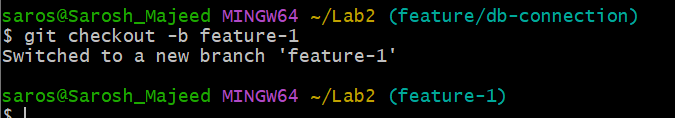
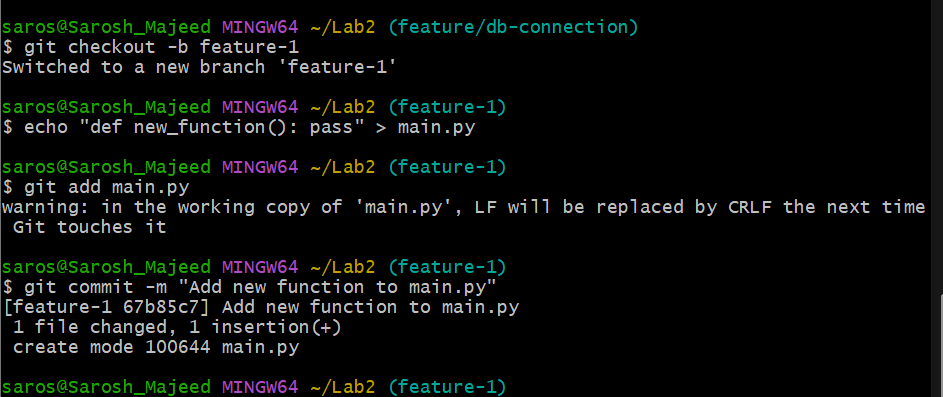
**Task 8: Branch Creation & Push (Git Bash)**

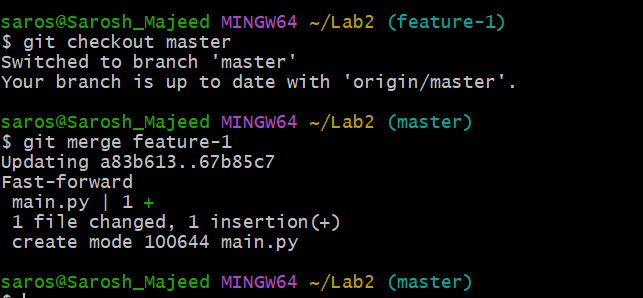
1. Create branch
2. Push branch



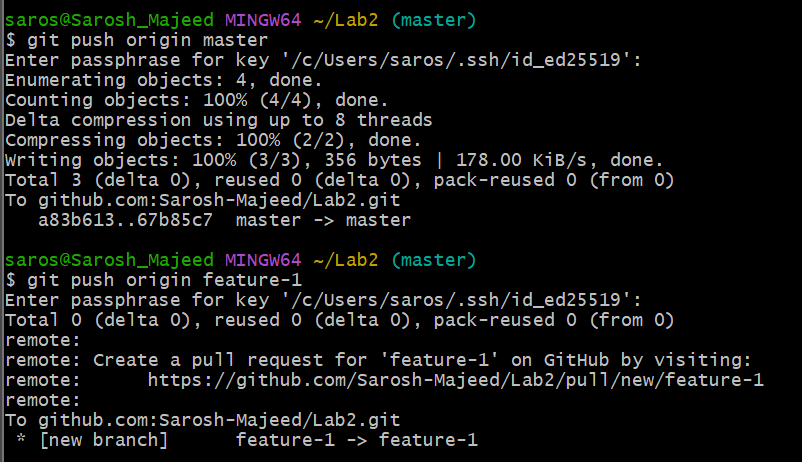


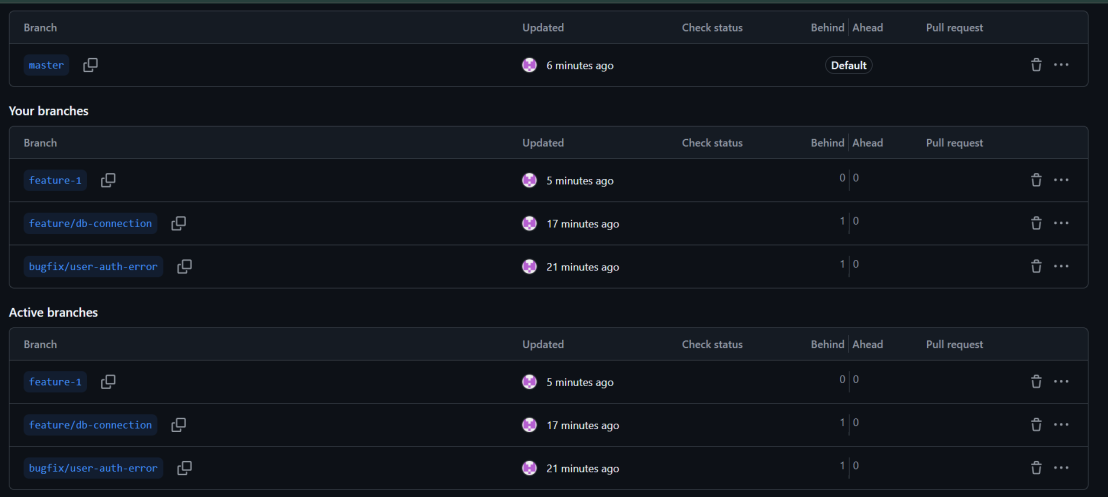
**Task 9: Branching & Merging**

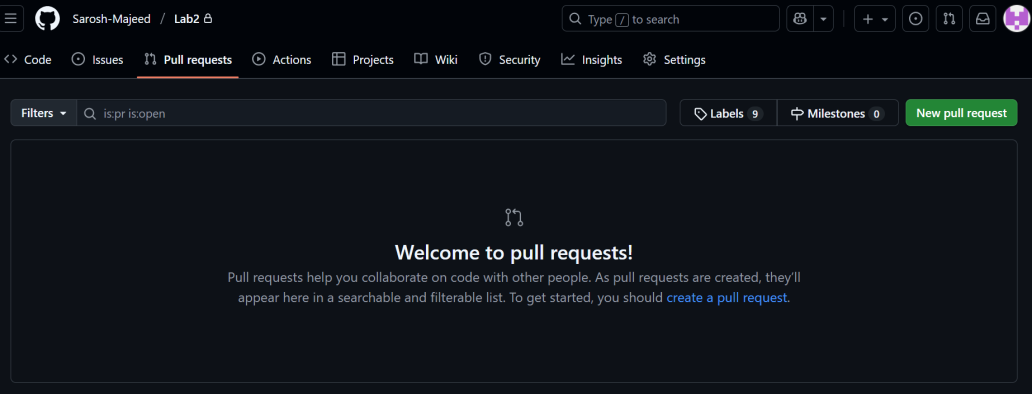
1. Create & switch
2. Edit main.py
3. Merge

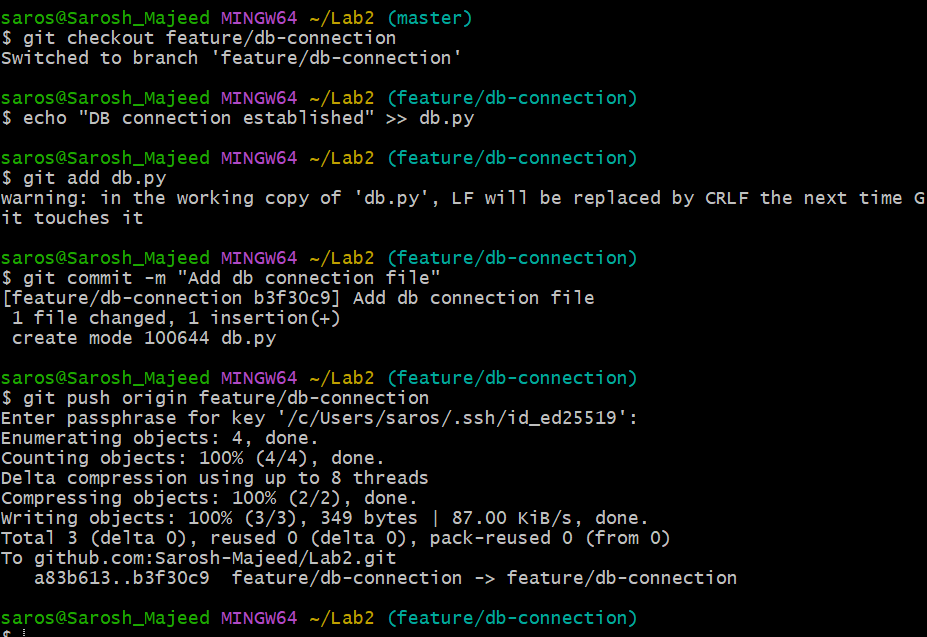


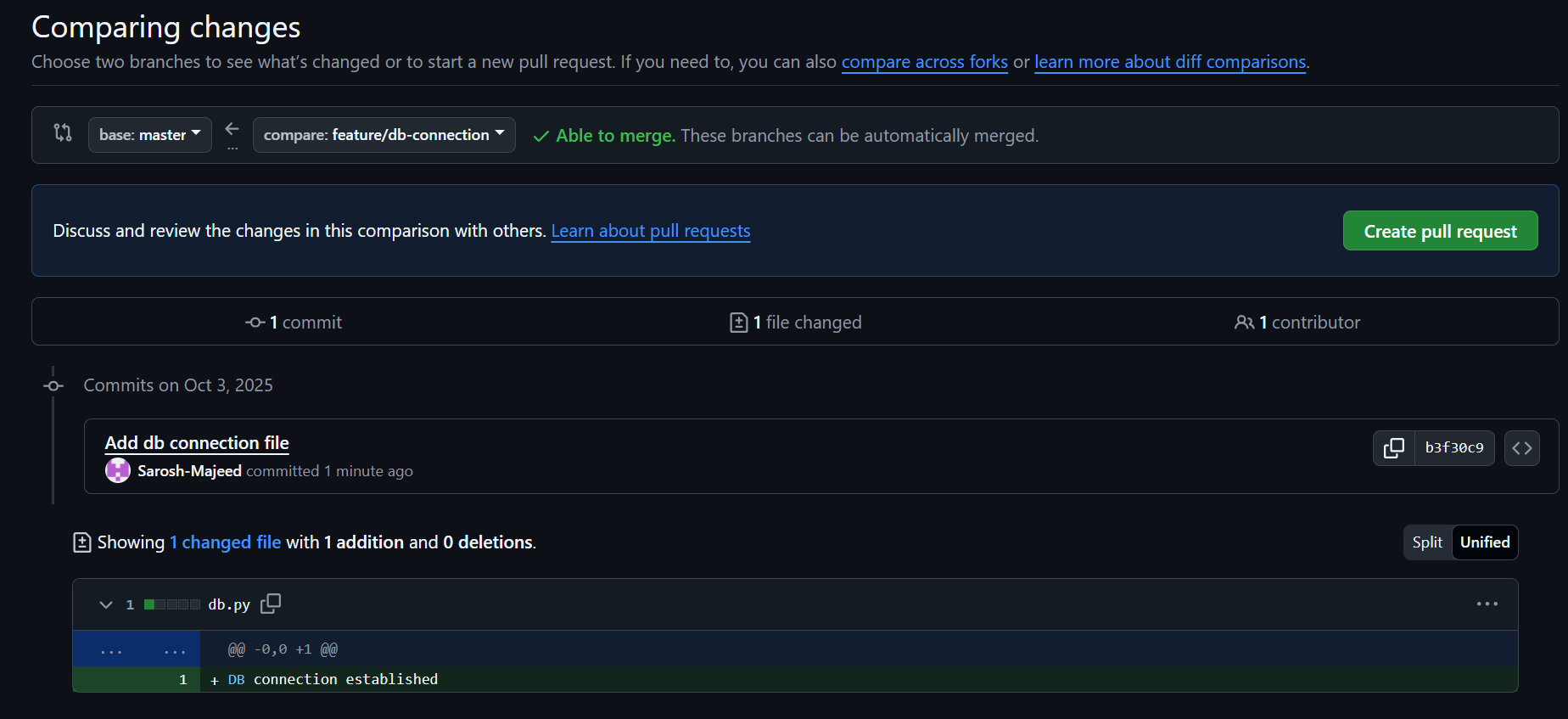
1. Push

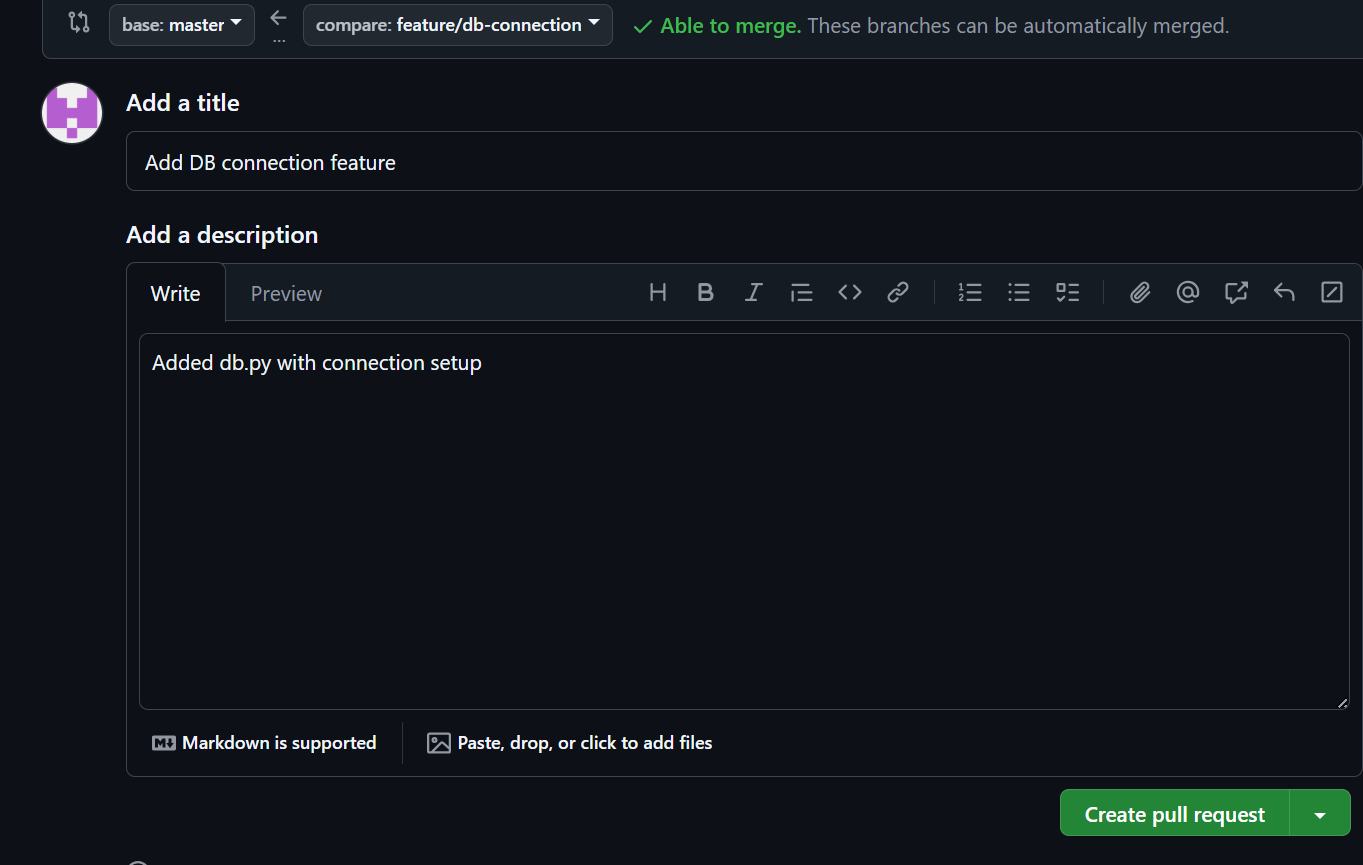


**Task 10: Pull Request (GitHub GUI)**

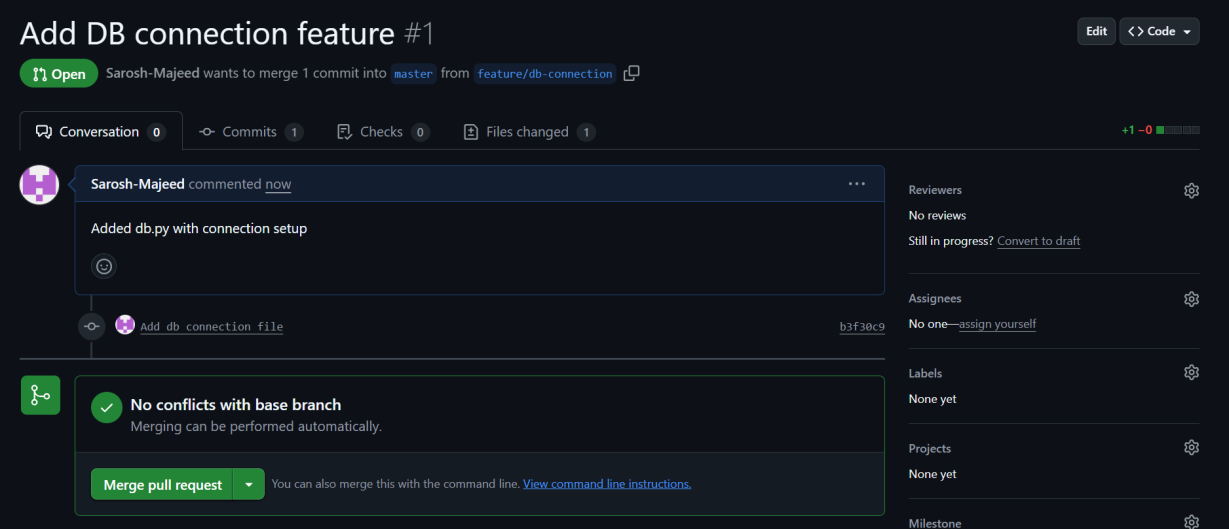
1. Creation

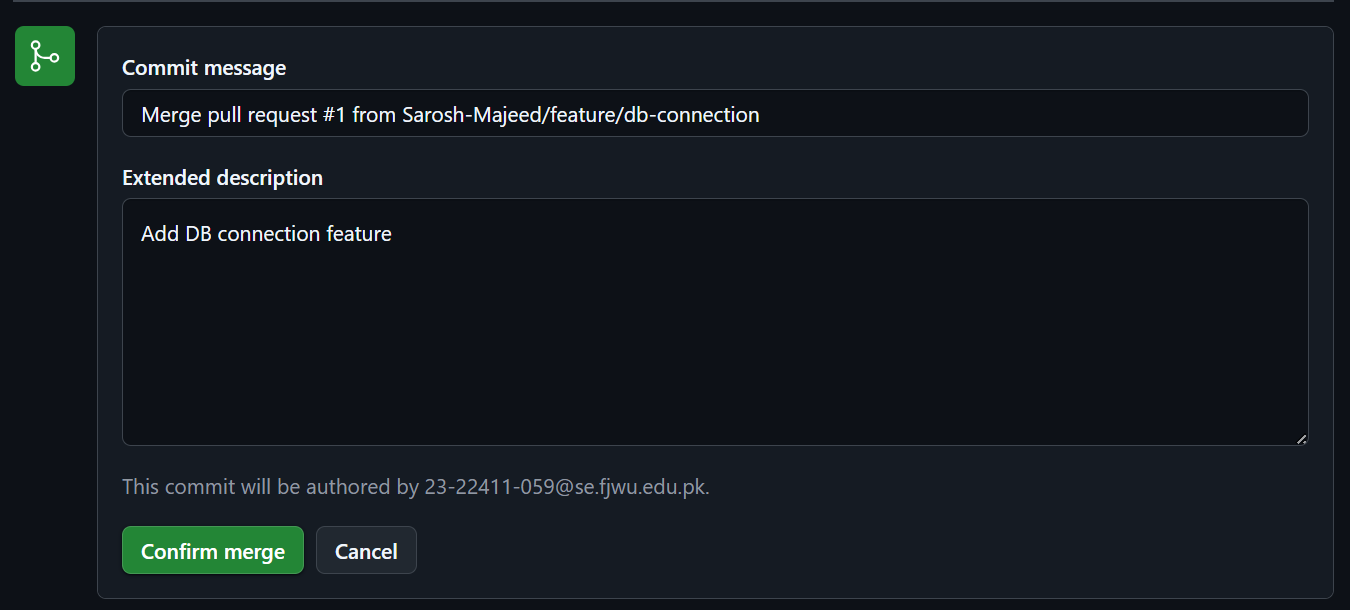


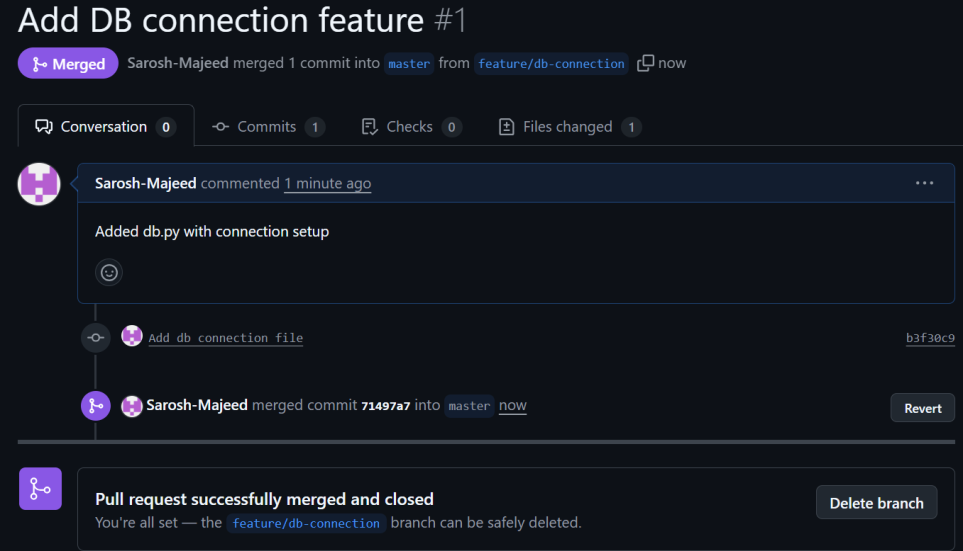


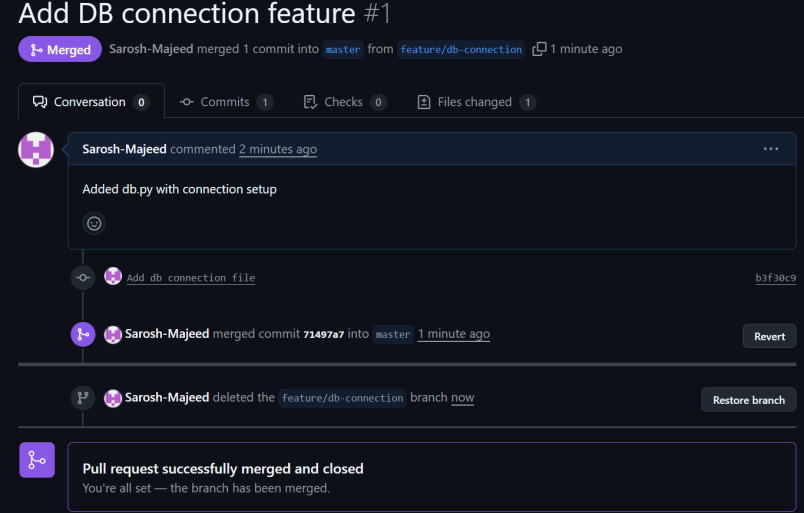


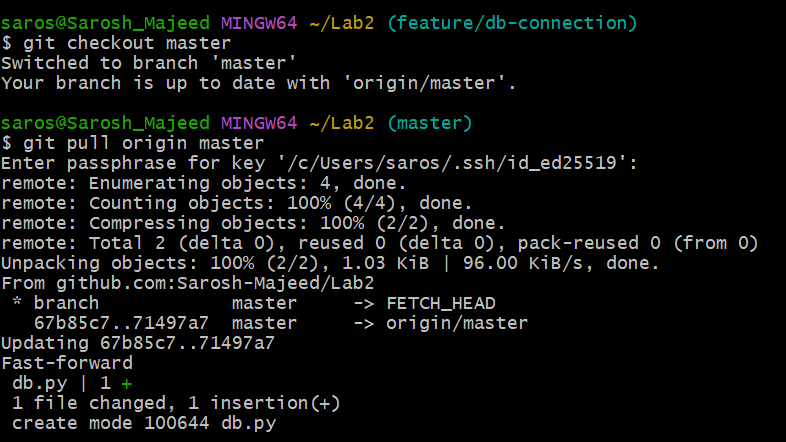
1. Review & merge.





1. Delete branch on GitHub.

****

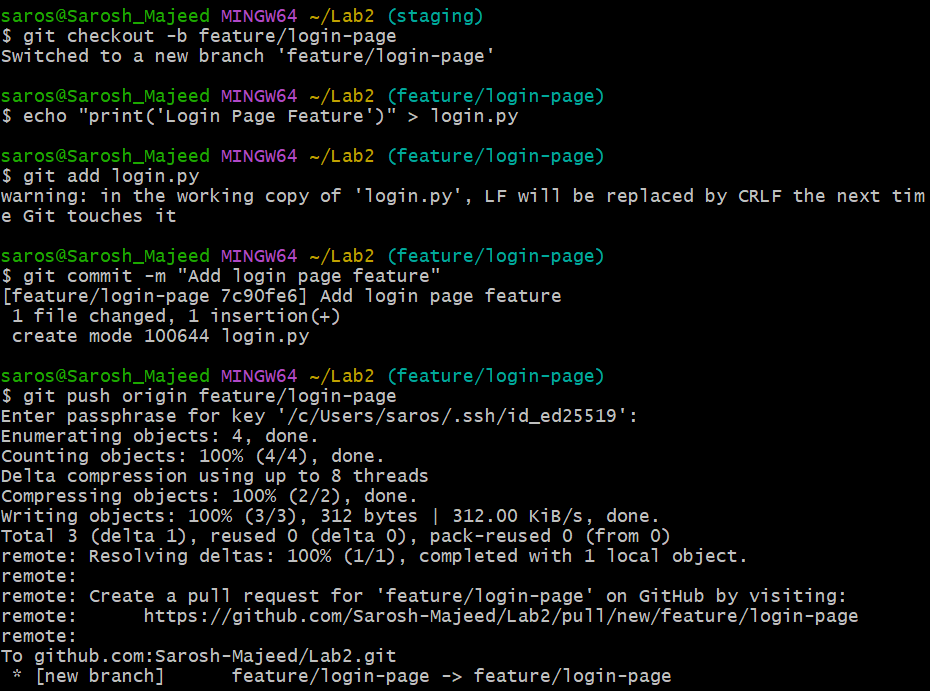
**Task 11: Branch Strategy**

1. Create branches

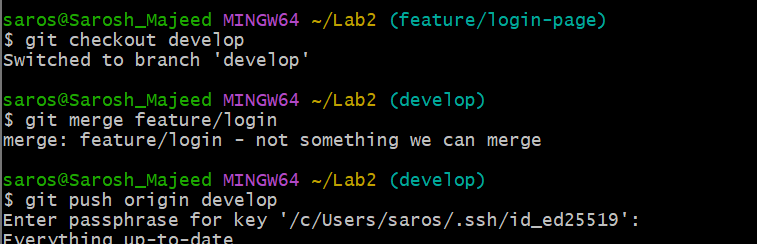
### 36.branch_strategy1.pngStep 1: Make develop branch

### 37.branch_strategy2.pngStep 2: Make staging branch

### Step 3: Create a new ****feature branch**** (simulating developer work)

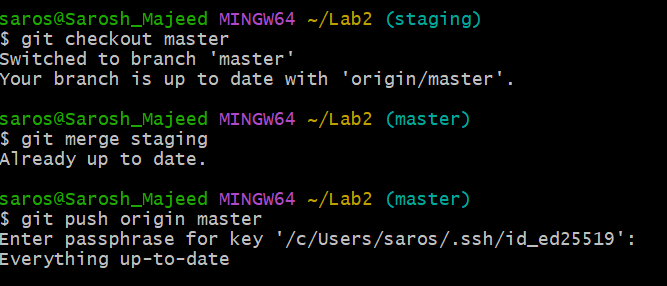


### Step 4: Merge feature/login-page → develop



### 40.branch_merger2.pngStep 5: Merge develop → staging

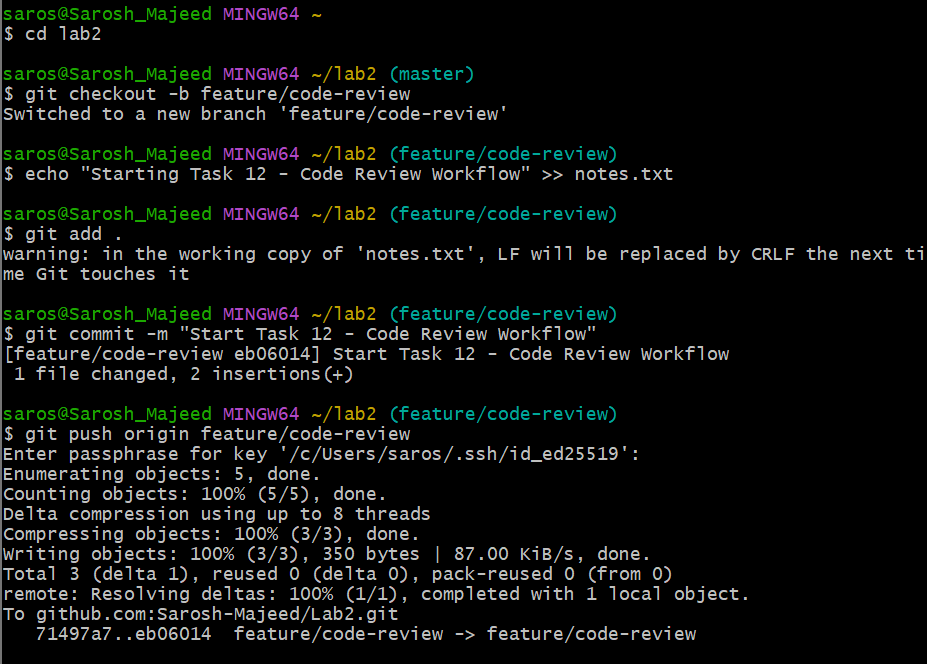
### Step 6: Merge staging → master (final release to production)



## ****Task 12: Code Review Workflow****

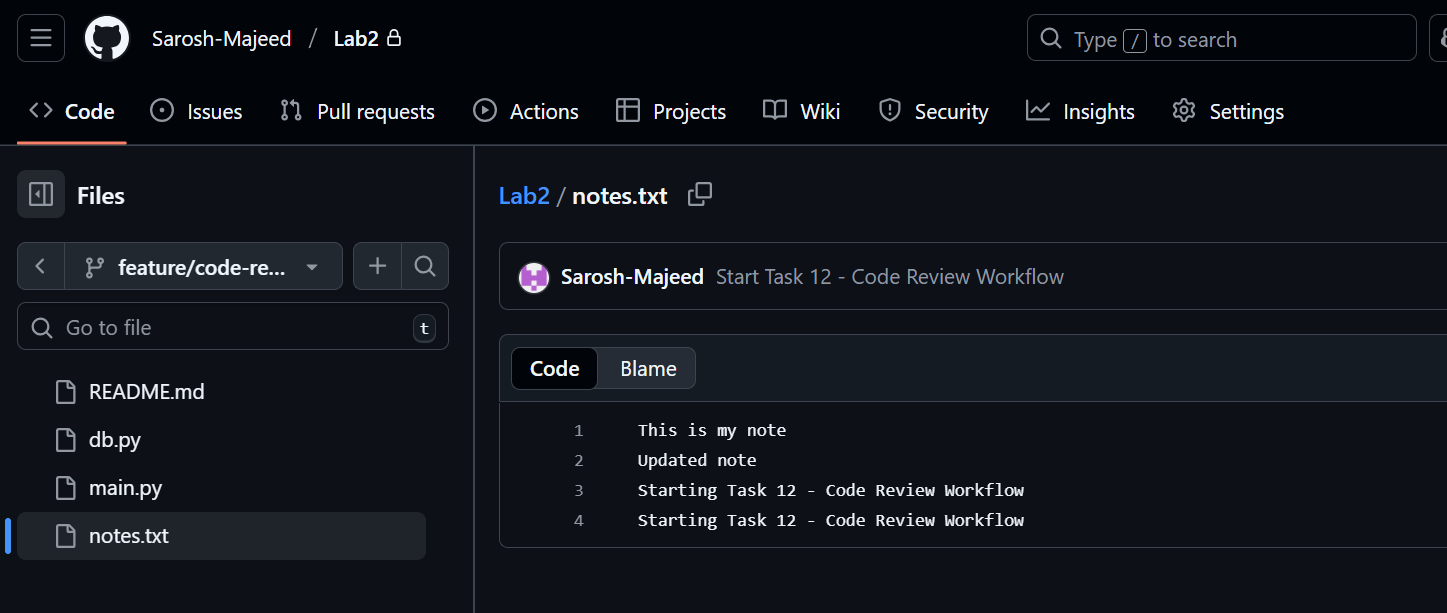
This task demonstrates how to create and manage a pull request in GitHub, simulate a code review, make requested changes, and finally merge and delete the feature branch.

### ****1.Creating a New Branch and Pushing It (Git Bash)****

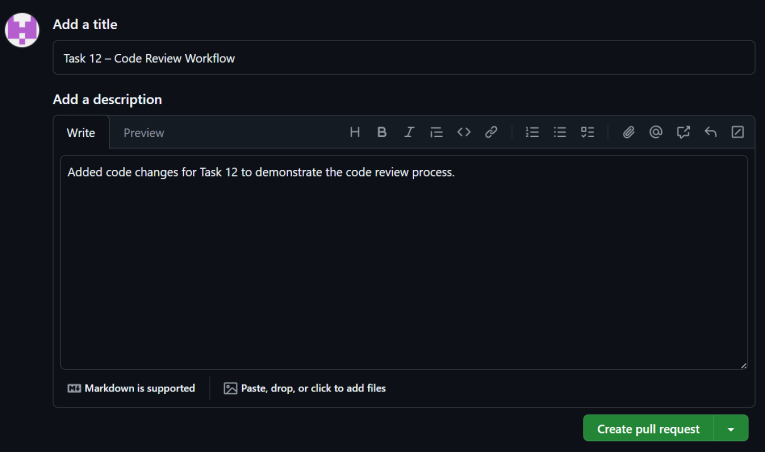
This screenshot shows the creation of the feature/code-review branch and pushing it to GitHub from Git Bash.

### ****2. Verifying the New Branch on GitHub****

This image shows the new feature/code-review branch successfully created and visible in the GitHub repository.



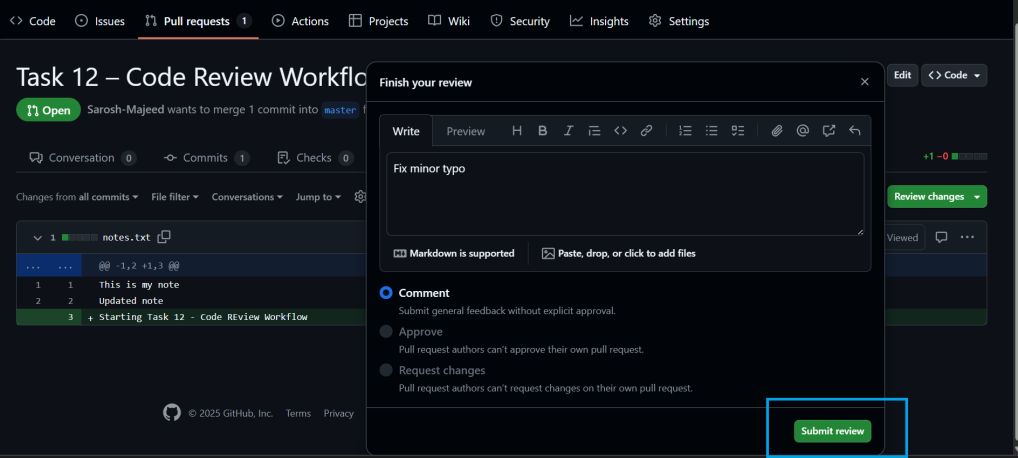
### ****3. PR Creation —**** pr\_create\_details.png

This screenshot shows the pull request being created from feature/code-review into the master branch with a title and description.

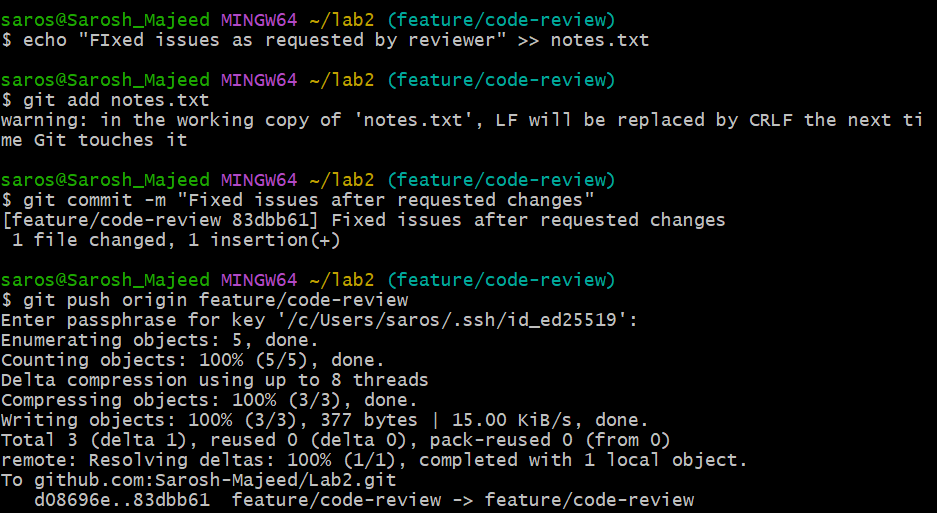
### ****4. Assigning a Reviewer —**** pr\_assigned\_reviewer.png

This shows assigning a reviewer (or yourself) to the pull request to simulate the review workflow.

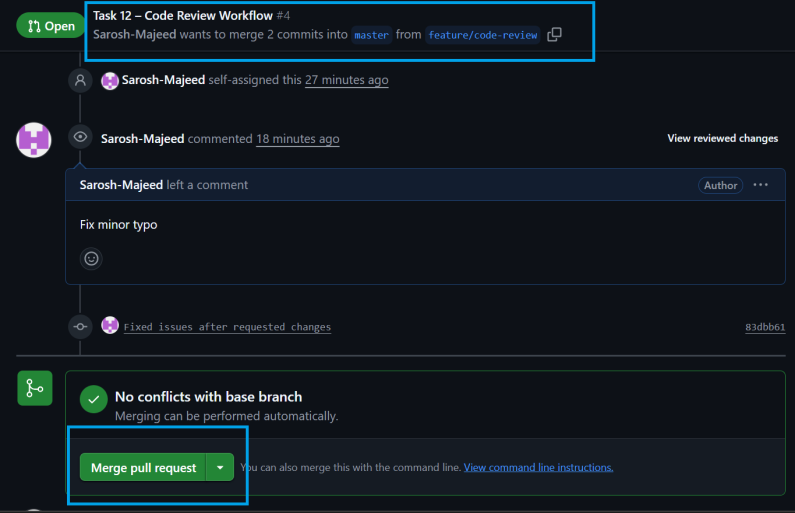
### ****5. Requesting Changes —**** pr\_request\_changes.png

This screenshot shows the reviewer requesting changes and adding feedback on the pull request.

### ****6. Updating the PR After Fix —**** pr\_updated\_with\_commits.png

This shows the author pushing a new commit after fixing the requested changes, which automatically updates the pull request.

### ****7. Merge Pull Request (Part 1) —**** pr\_merge\_confirm\_part1.png

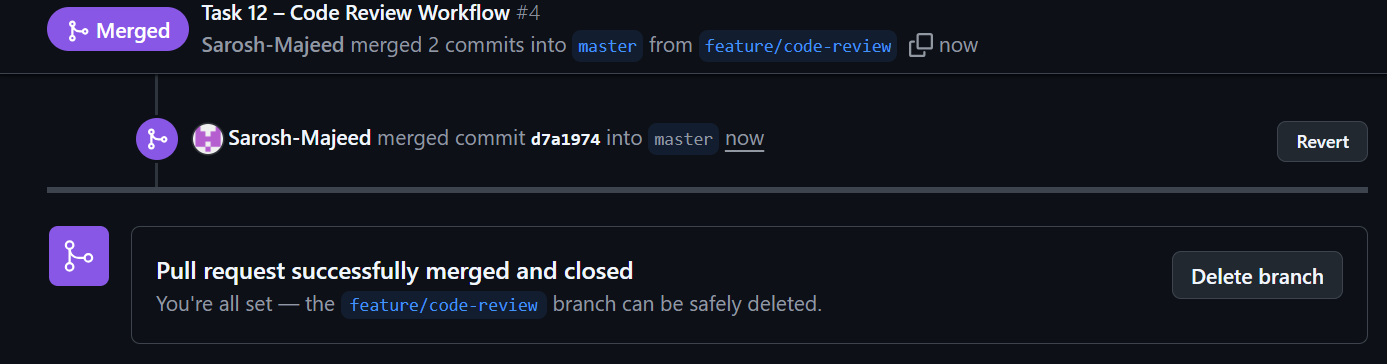
This screenshot shows the start of the merge process by clicking the **Merge pull request** button.

### ****8. Merge Confirmation (Part 2) —**** pr\_merge\_confirm\_part2.png

This shows the confirmation step where the merge is approved by clicking **Confirm merge** on GitHub.

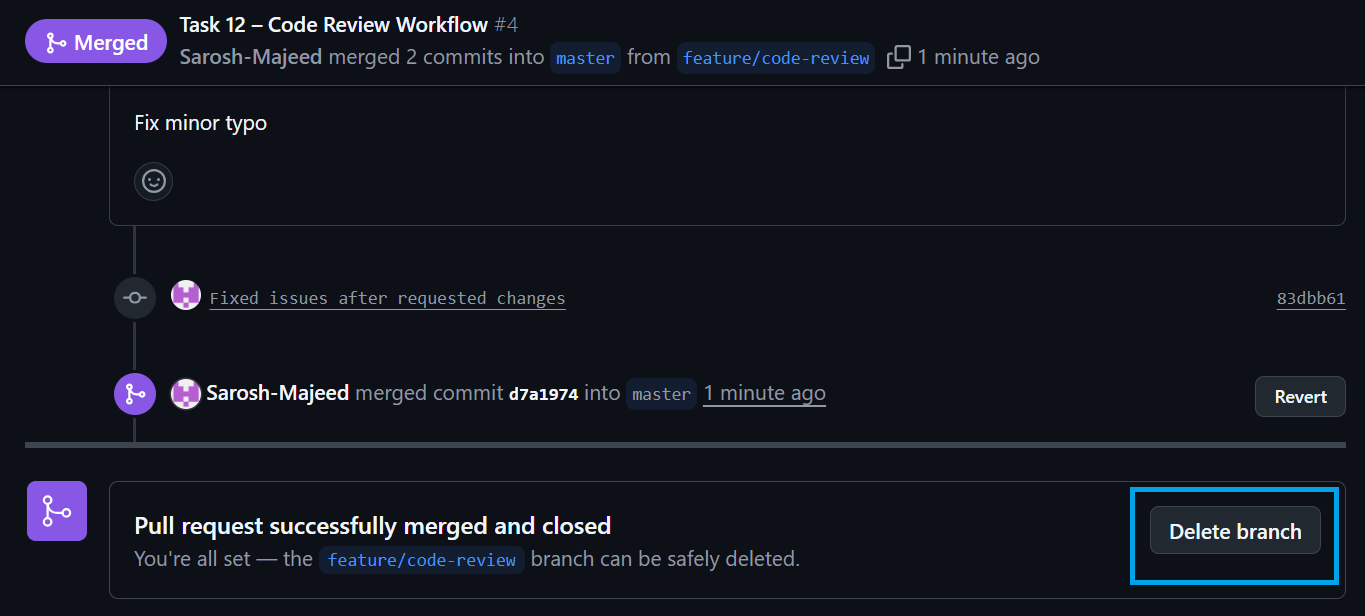
### 50.pr_merge_confirm(part2).png

### ****9. Pull Request Merged —**** pr\_merged.png

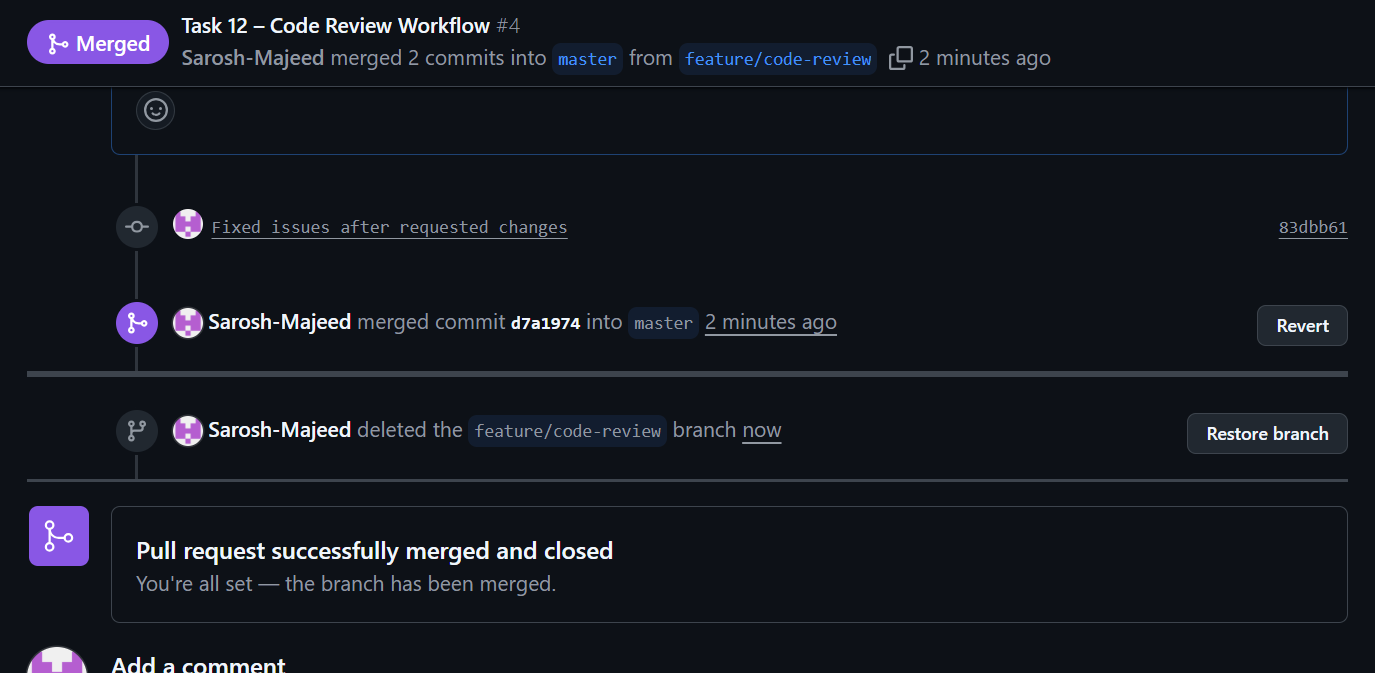
This screenshot shows the success message **“Pull request successfully merged and closed”** after merging.

### ****10. Deleting Branch (Part 1) —**** pr\_delete\_branch\_part1.png

This shows the **Delete branch** option appearing after merging the pull request.



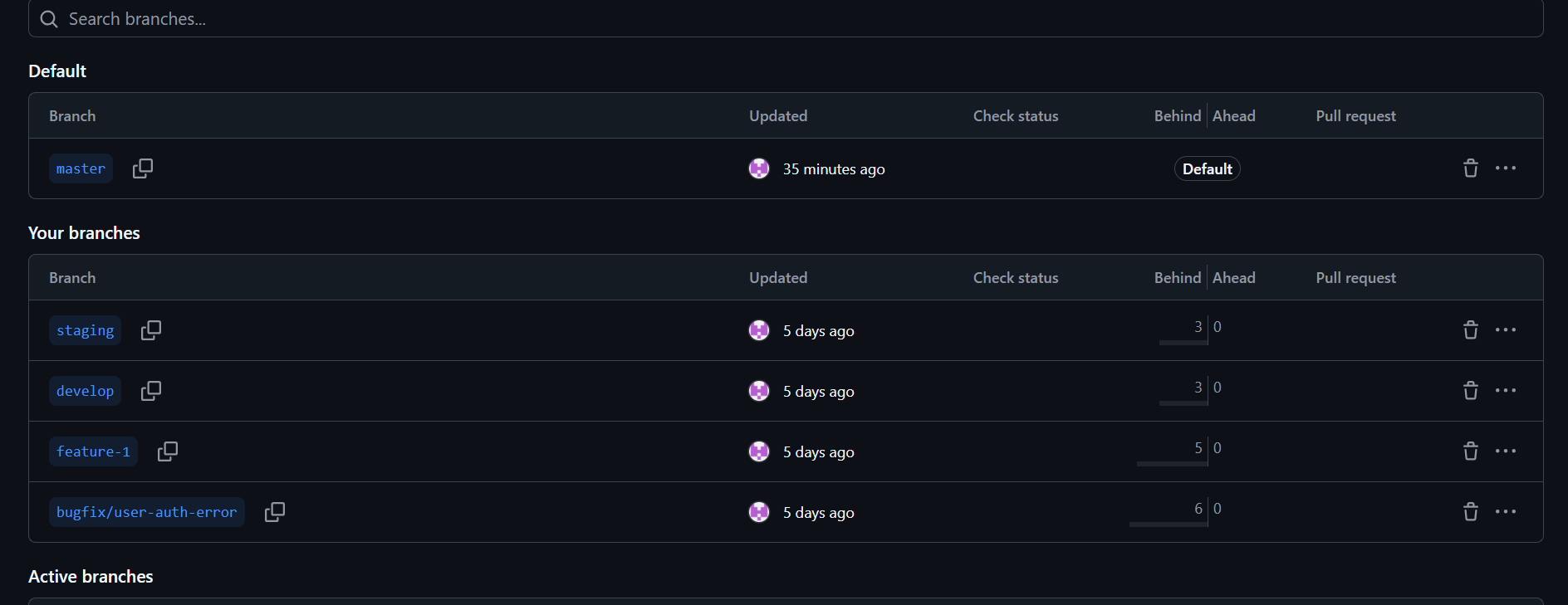
### ****11. Branch Deleted Successfully —**** pr\_delete\_branch\_part2.png

This confirms that the feature branch (feature/code-review) was successfully deleted from the remote repository.

## ****Task 13: Branch Cleanup Best Practices****

This task ensures that both **remote** and **local** branches are deleted after they’ve been successfully merged, keeping the repository clean and well-organized.

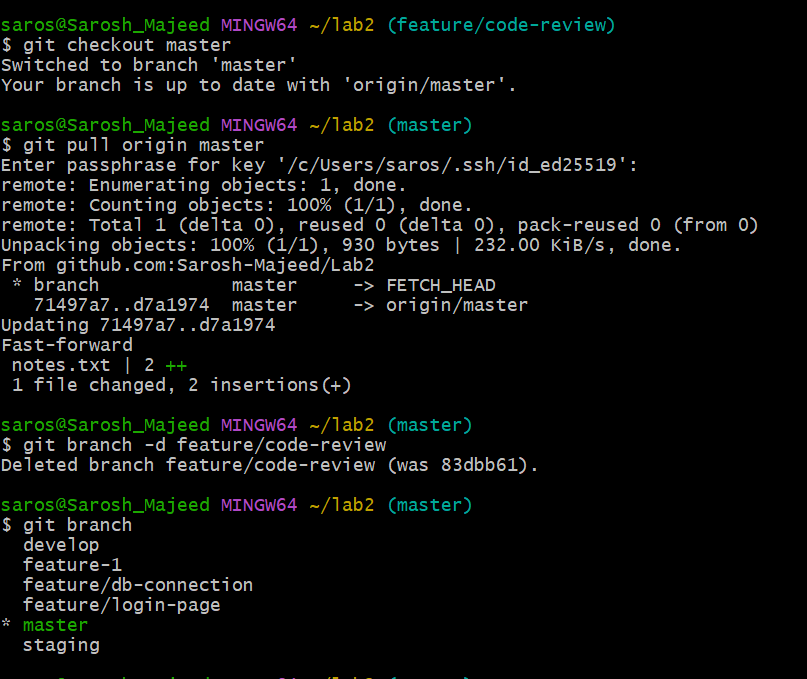
### ****1.Deleting Remote Branch on GitHub —**** remote\_branch\_deleted.png



**What was done:**  
The merged branch feature/code-review was deleted directly from the GitHub interface by clicking the icon in the **Branches** tab.

**Why:**  
Removing the branch from GitHub prevents clutter in the remote repository and makes sure that only active development branches remain visible.

### ****2. Deleting Local Branch Using Git Bash —**** remote\_branch\_delete\_cmd.png

****

**What was done:**  
The same branch feature/code-review was deleted locally using the commands:

git checkout master

git pull origin master

git branch -d feature/code-review

**Why:**  
This step removes the merged branch from the local system to free space and avoid confusion when switching or creating new branches in the future.

**Result:**  
After completing these steps, both the **remote** and **local** versions of feature/code-review were successfully removed, confirming a clean and professional branch management process.