Fatima Jinnah Women University

*Department of Software Engineering*



*---------------------------------------------------------------------------------------------------------------------------------------*

LAB #03

**SUBJECT: CLOUD COMPUTING**

**SUBMITTED TO: SIR MUHAMMAD SHOAIB**

**SUBMITTED BY: UMBER QASIM**

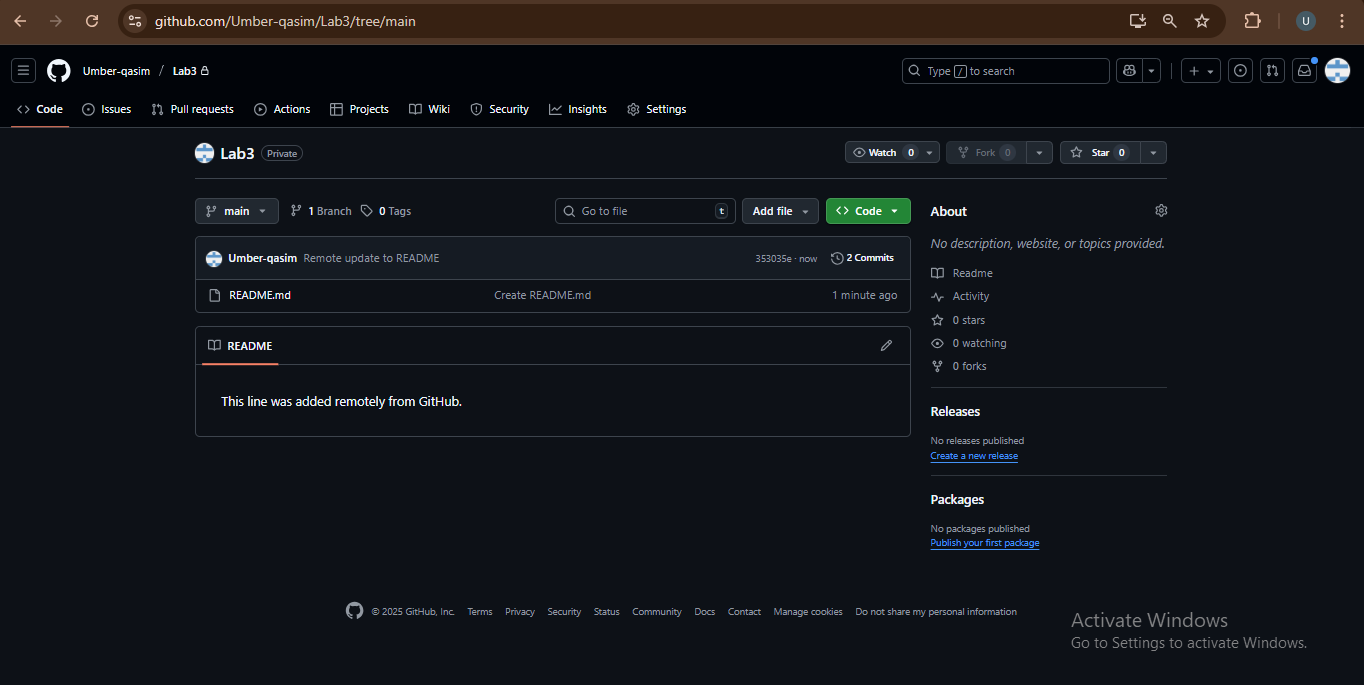
**REGISTRATION NO: 2023-BSE-066**

**CLASS: BSSE V-B**

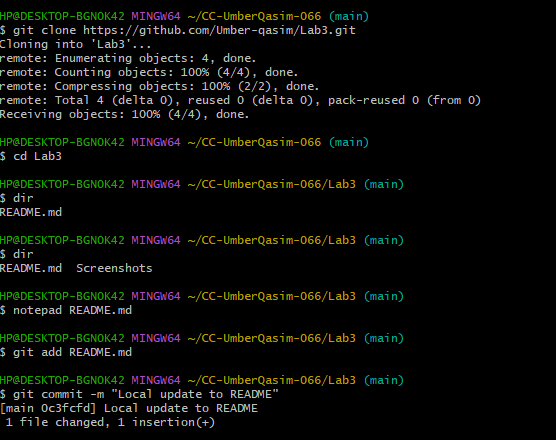
***Working with Git History, Stashing, and Reverting Commits***

**Task#01: Handling Local and Remote Commit Conflicts (Pull vs Pull --rebase)**

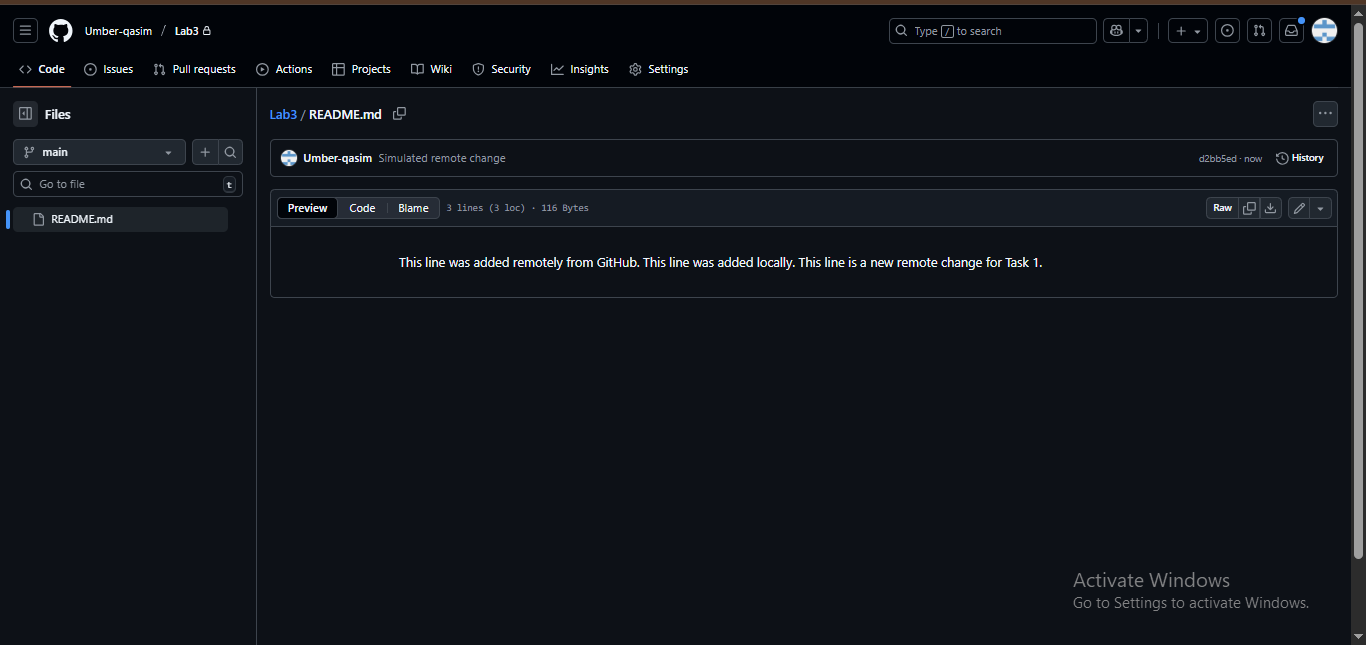
*Remote edit on GitHub*

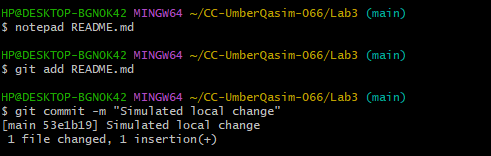


*Make a local change*

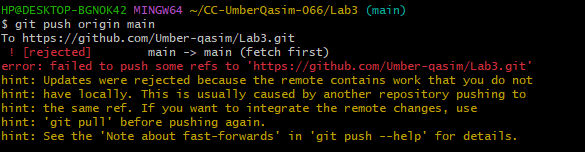


*This time it was pushed successfully so we try again:*

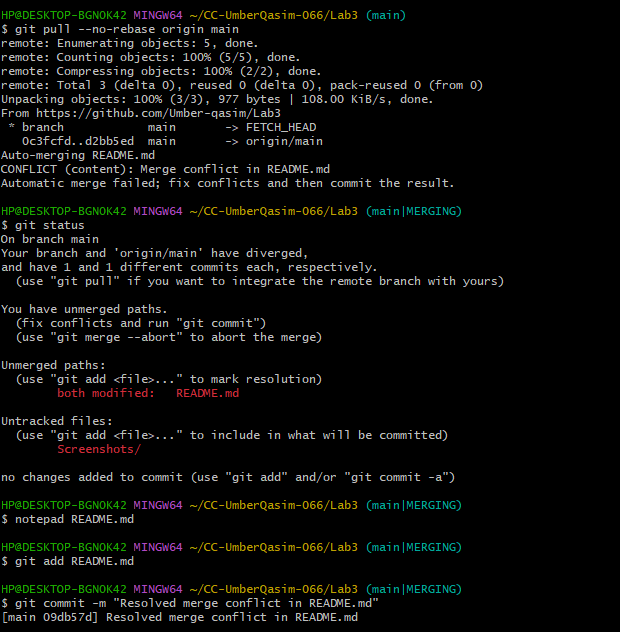




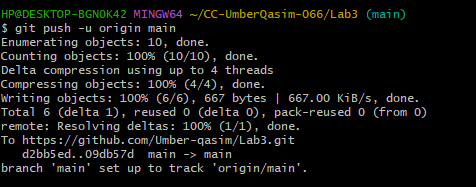
*Try pushing local change*



*Pull with merge*

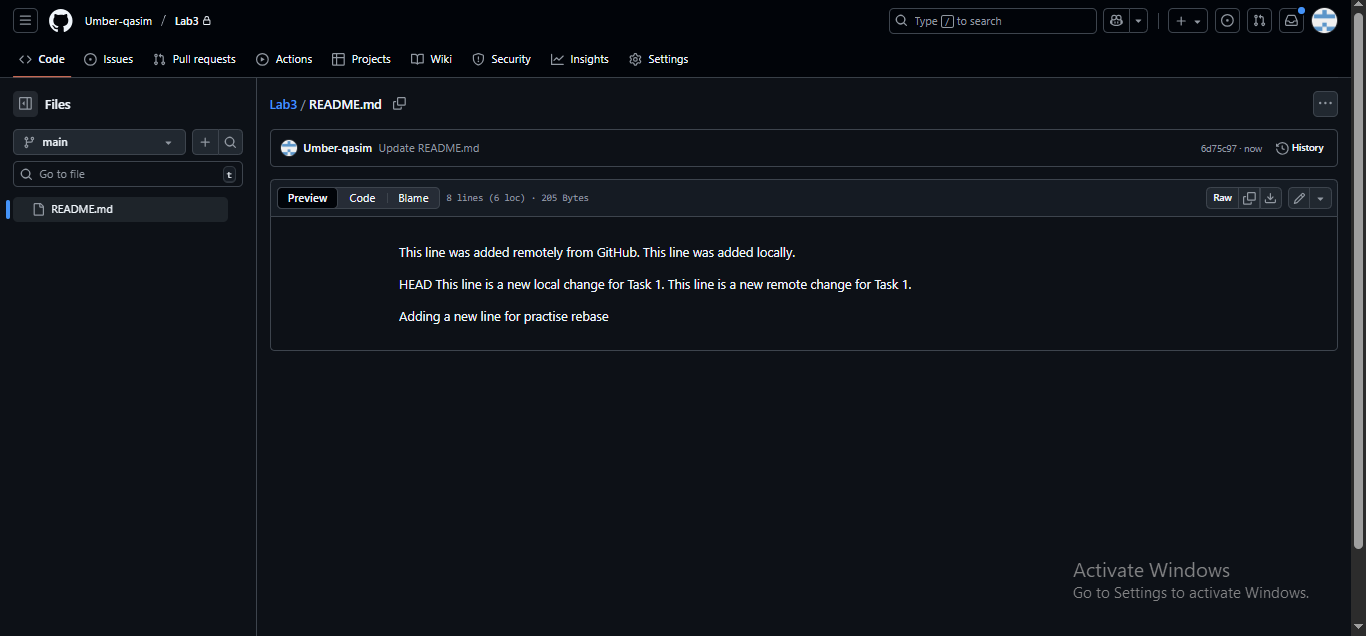


*Push after merge*

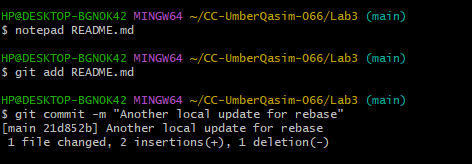


*Rebase practice*

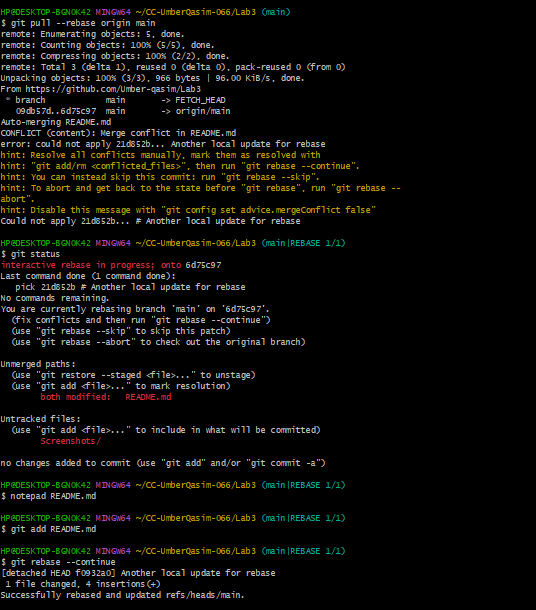
**Make another remote edit** on GitHub README.md



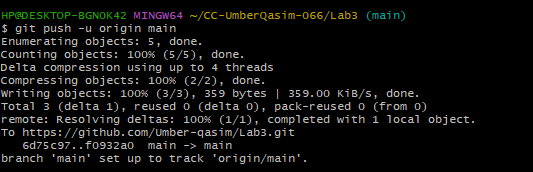
*Same as locally*



*Pull with rebase*

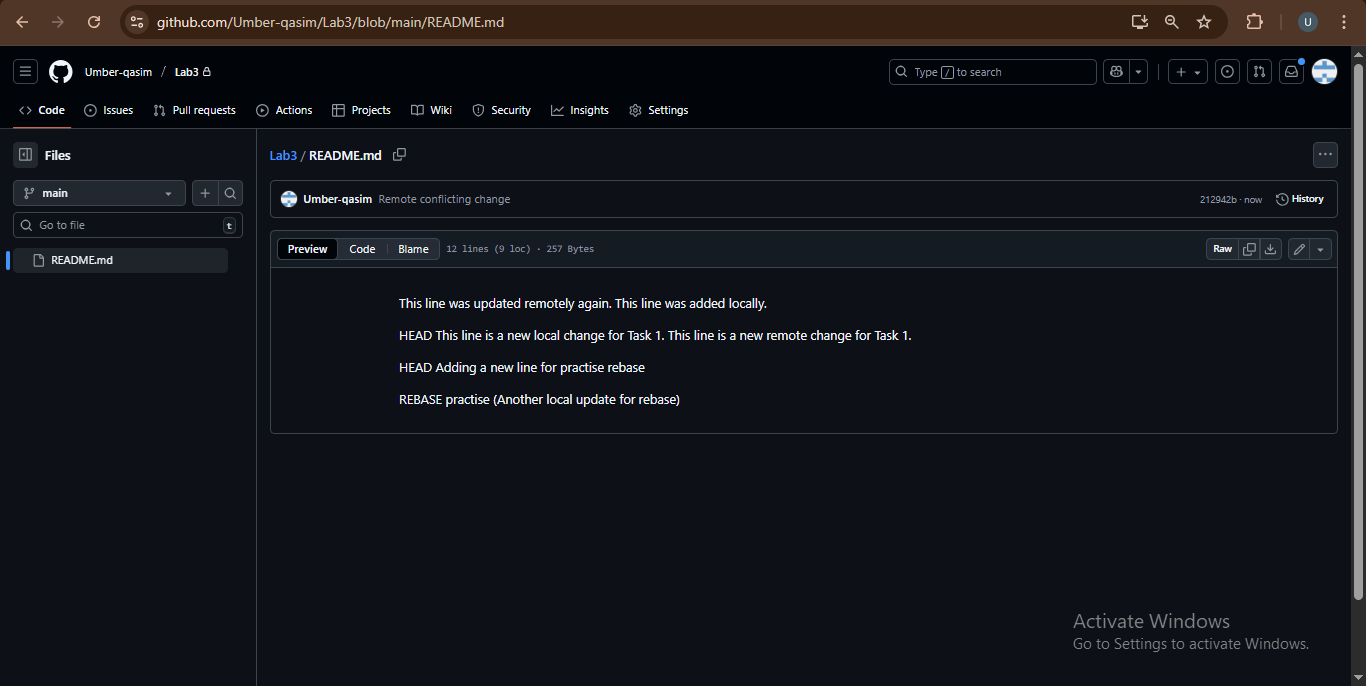


*Push after rebase*

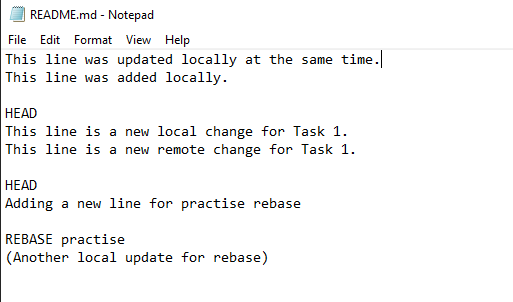


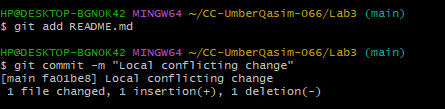
**Task#02: Creating and Resolving Merge Conflicts Manually**

*Remote conflicting edit (GitHub)*

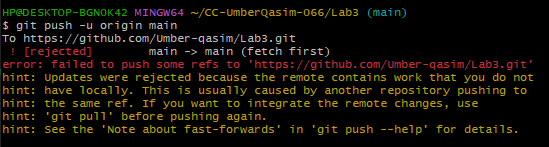


*Local conflicting edit*

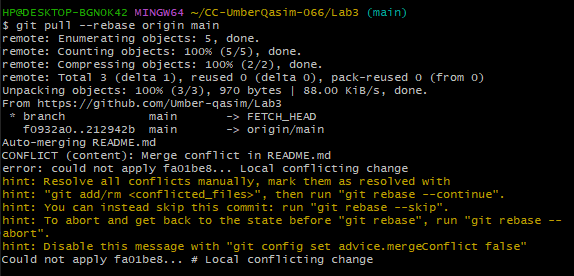




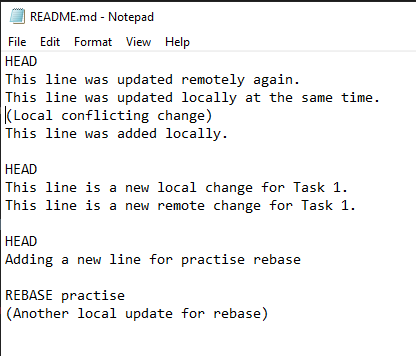
*Try to push*



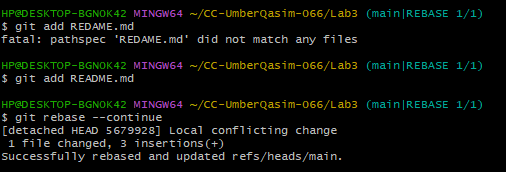
*Pull with rebase*



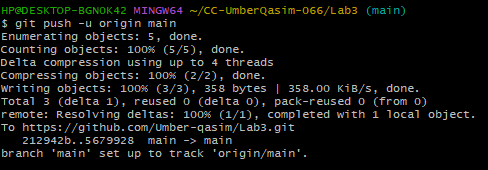
*Resolve conflict manually*



*Mark conflict as resolved & continue rebase*

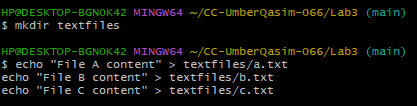


*Push after resolving conflict*

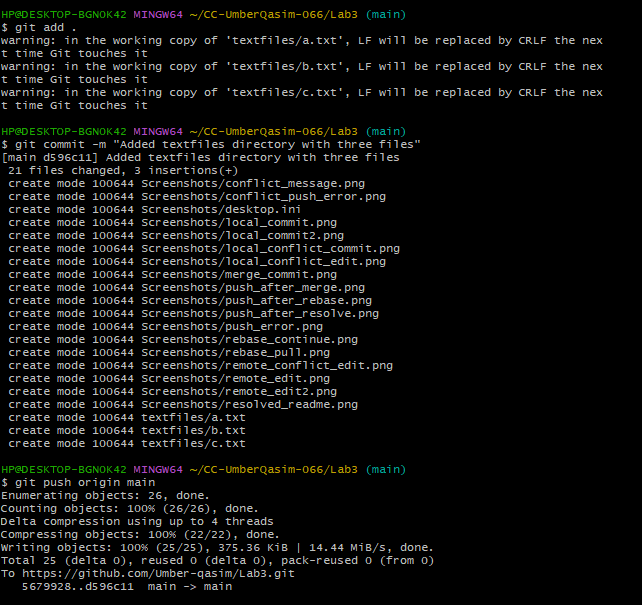


**Task#03: Managing Ignored Files with. gitignore and Removing Tracked Files**

*Create a folder textfiles with three text files inside it*



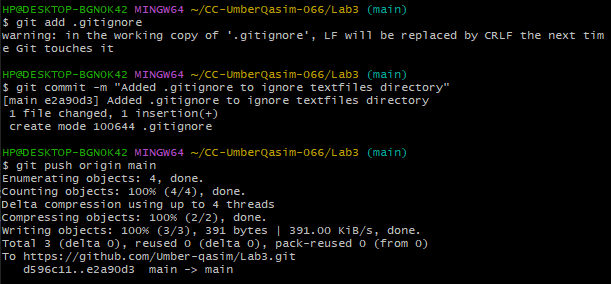
*Stage and commit the new directory*



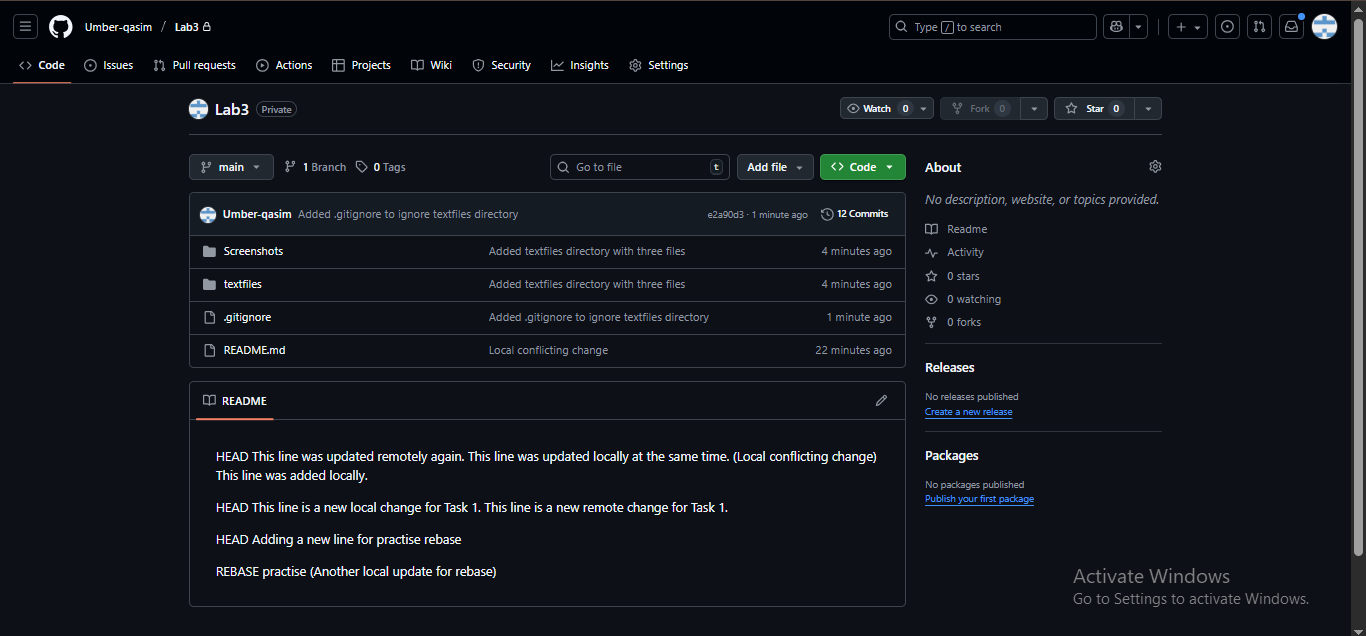
*Create a .gitignore file to ignore textfiles*



*Stage & commit .gitignore*

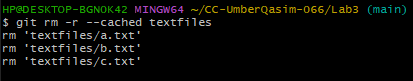


*Check repo on GitHub*

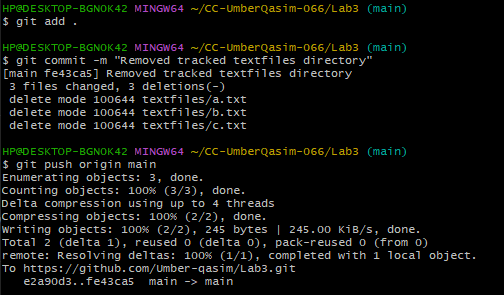


Reason: **Git already tracked the files before .gitignore**

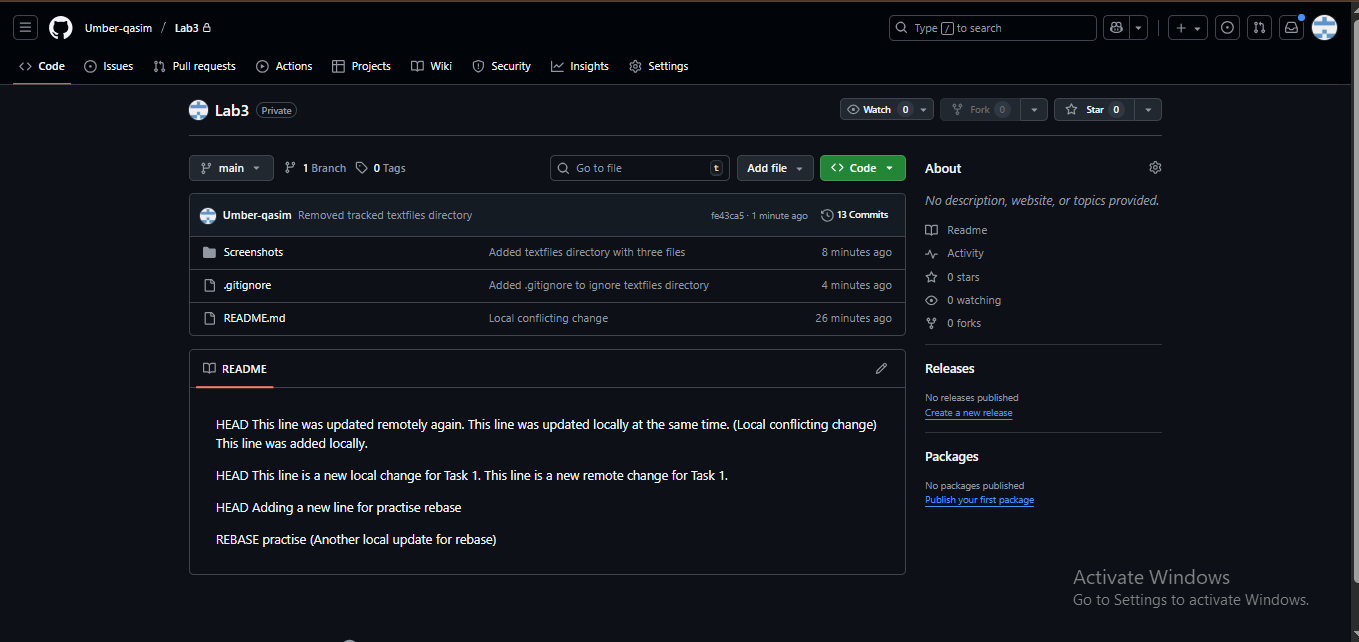
*Remove tracked files but keep locally*



*Commit & push changes*

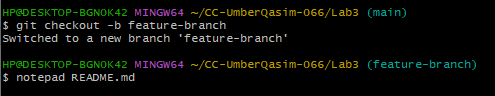


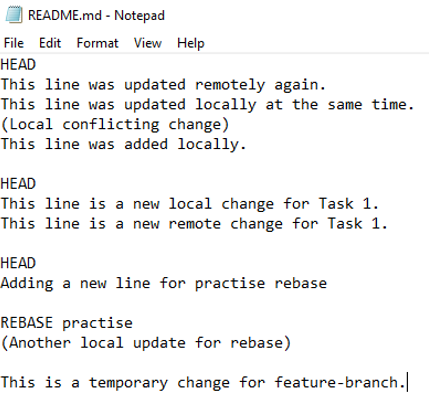
*Check GitHub repo*



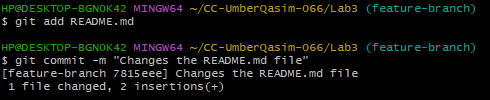
**Task#04: Create Temporary Changes and Use git stash**

*Create a new branch feature-branch then modify README.md and commit*



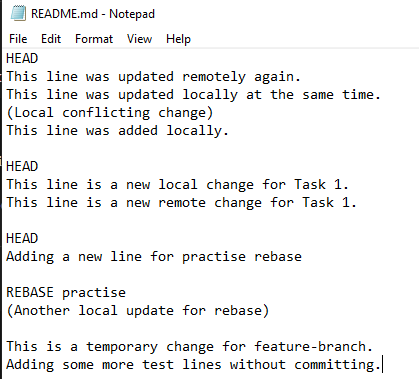


*Stage & commit:*

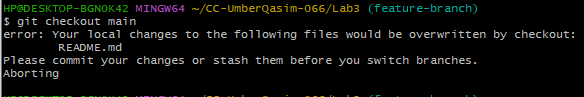


*Make new uncommitted changes*

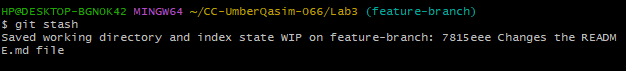




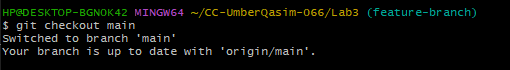
*Try switching branch*



*Stash your changes*



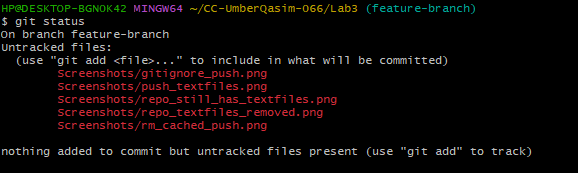
*Switch branch (now it works)*



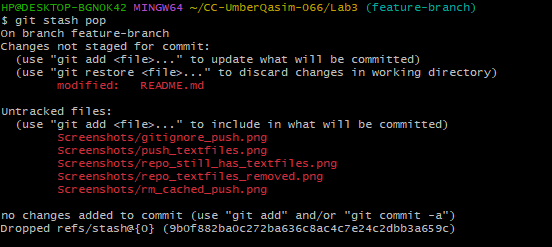
*Return to feature-branch*



*Check working directory status*

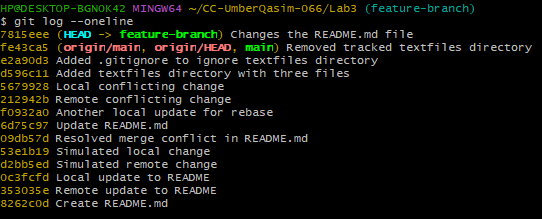


*Restore stashed changes*

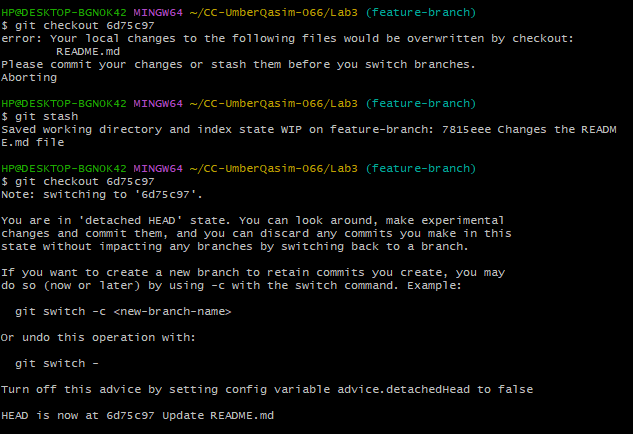


**Task#05: Checkout a Specific Commit Using git log**

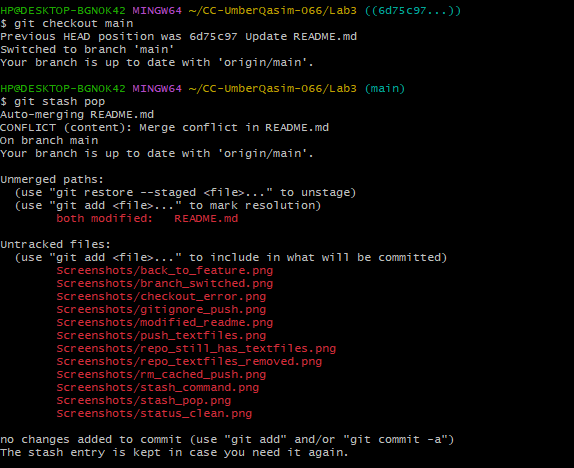
*View commit history*



Copy any **previous commit hash**

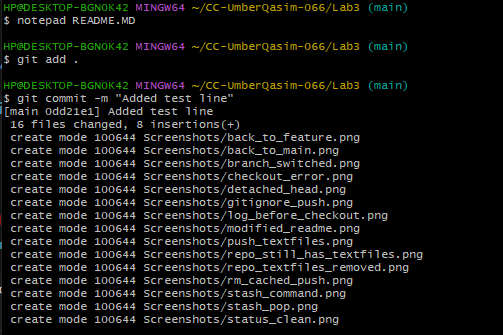


*Return to main branch*

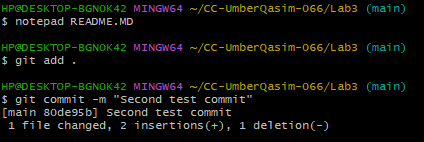


**Task#06: Resetting Commits (Soft vs Hard Reset) (With Verification Steps)**

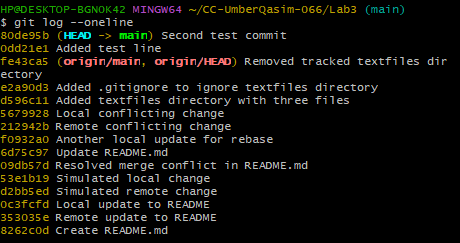
*Add a new line in any file and commit it*



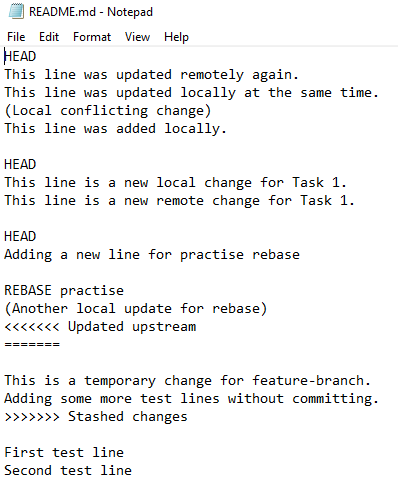
*Add another change and commit again*



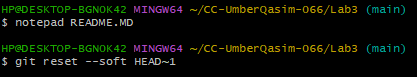
*View commit history before reset*



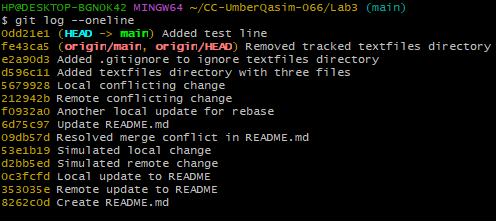
*Confirm both changes are in the file*



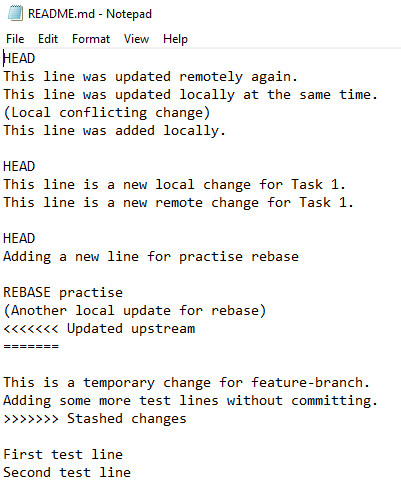
*Perform a soft reset*



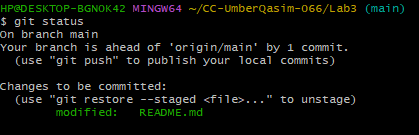
*Check commit history again*



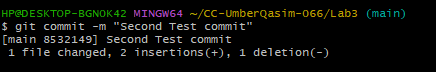
*Check file content after soft reset*



*Check git status*



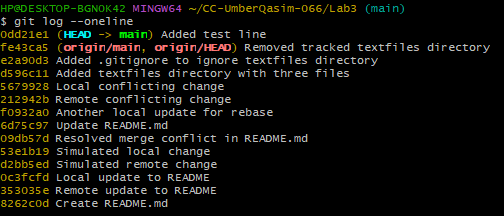
*If it shows staged changes, you can re-commit:*



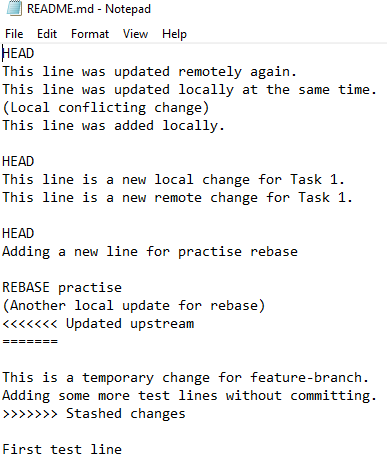
*Perform a hard reset*



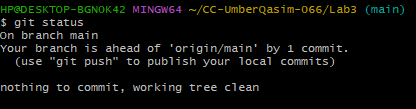
*Check commit history again*



*Open file again*



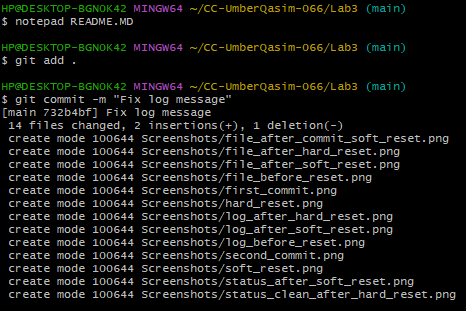
*Check git status*



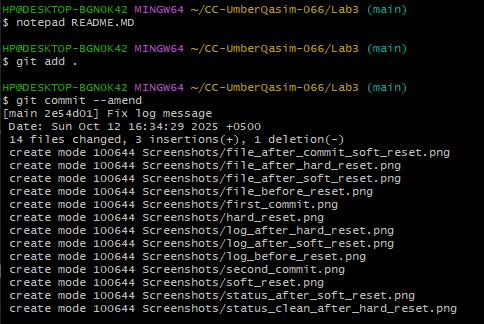
**Task#07: Amending the Last Commit**

Amending the Last Commit

**Make a small change in README.md file and Stage & commit normally**



Move back to the same file and add another new line then Stage the new change and amend your last commit

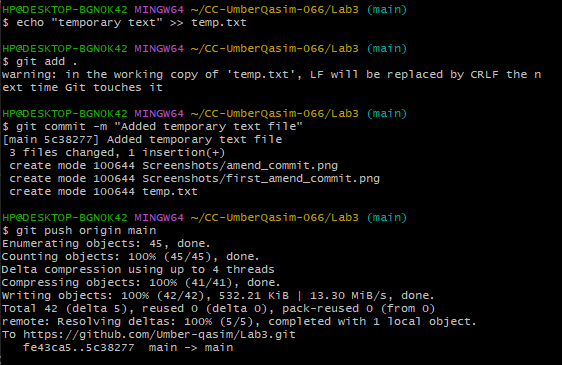


**Amended commit is complete and active**.

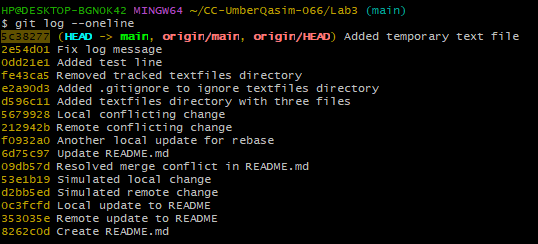


**Task#08: Reverting a Commit (Safe Undo on Remote Branch)**

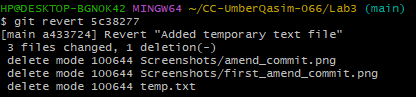
Make a new file and commit it



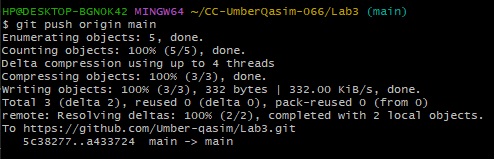
Check commit history



Revert that commit



Push the revert commit

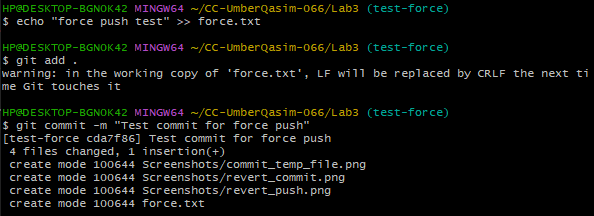


**Task#09: Force Push (With Caution)**

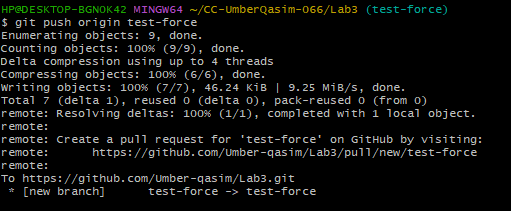
Create a new branch

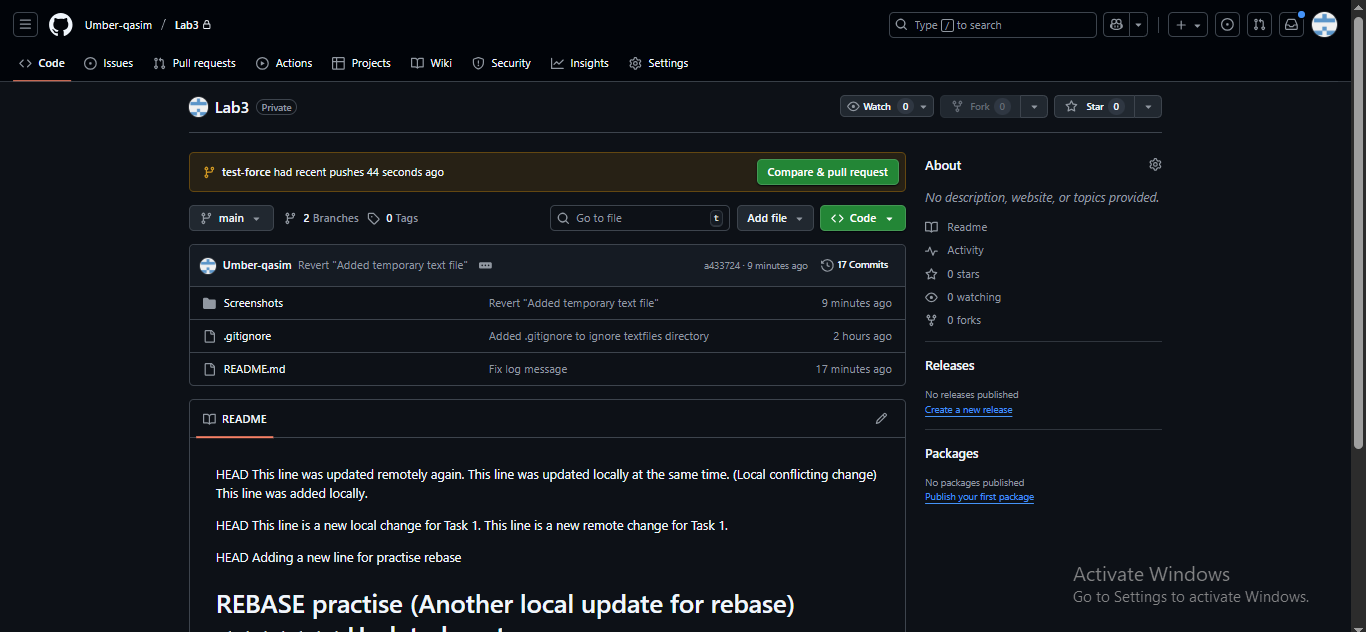


Make a small change



Push your new branch

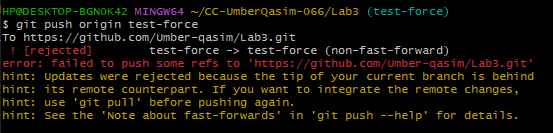




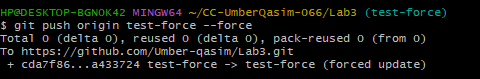
Perform a hard reset



Try normal push

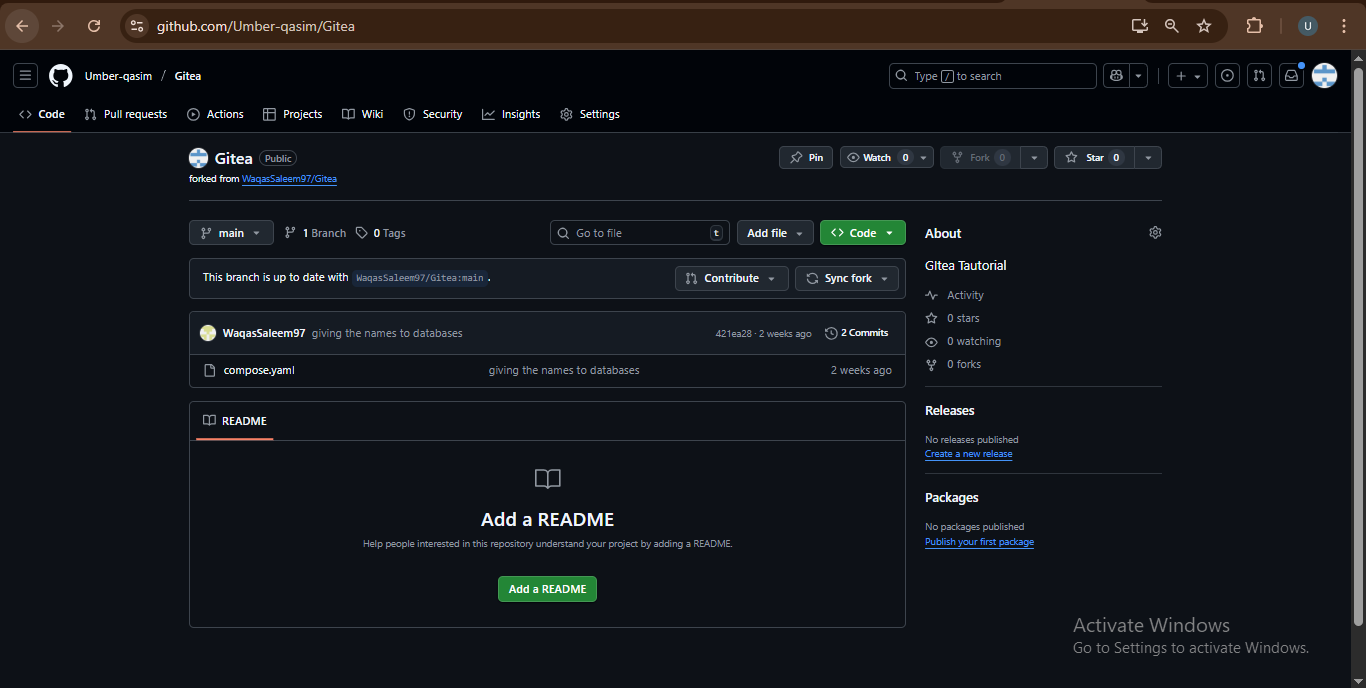


Force push

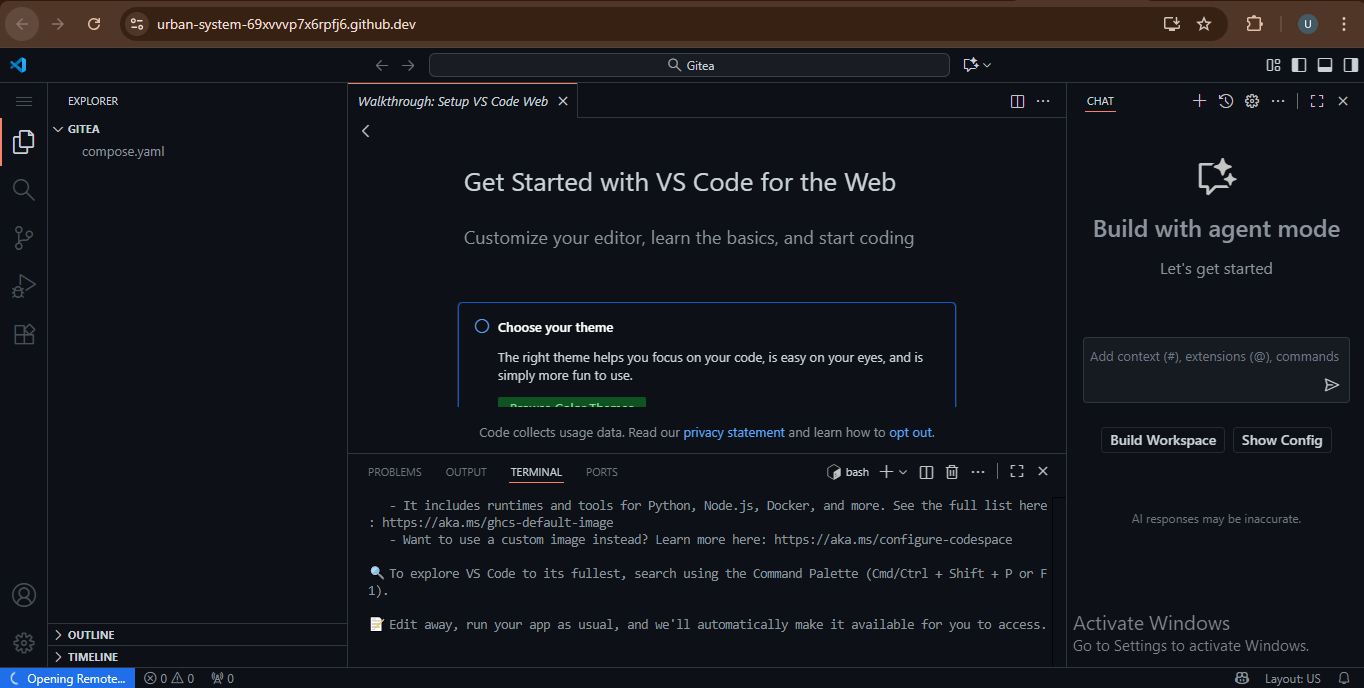


**Task#10: Running Gitea in GitHub Codespaces via Docker Compose**

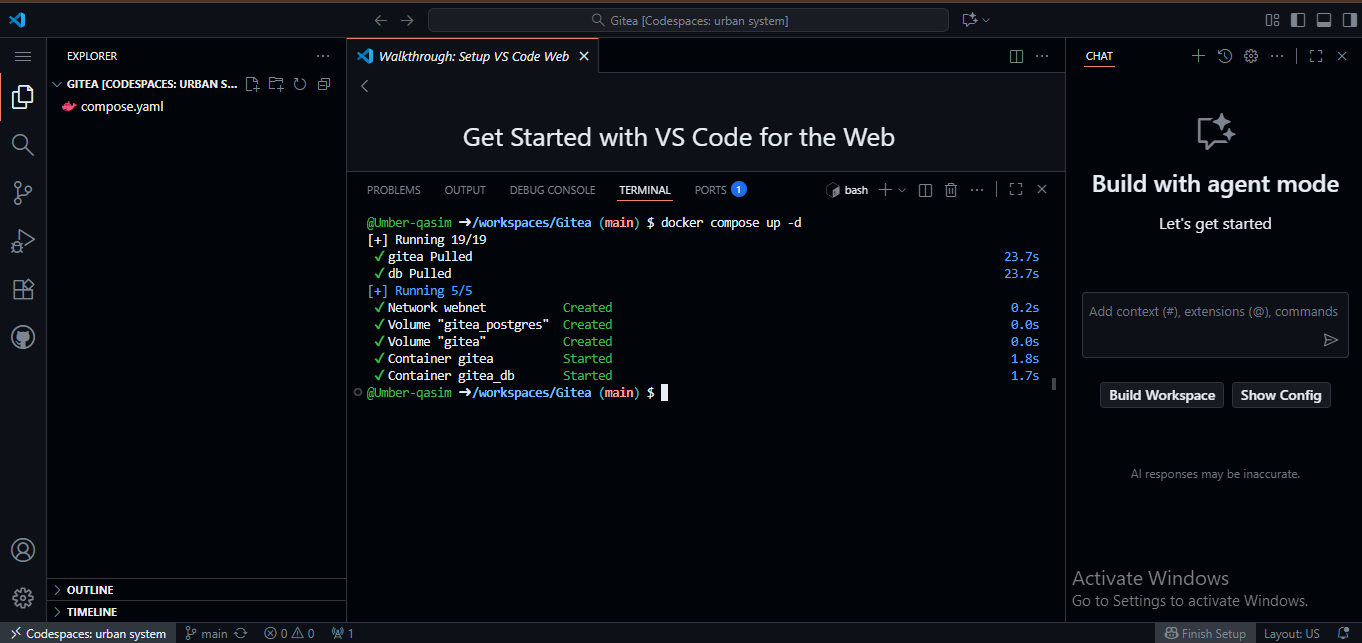
Fork the Repository



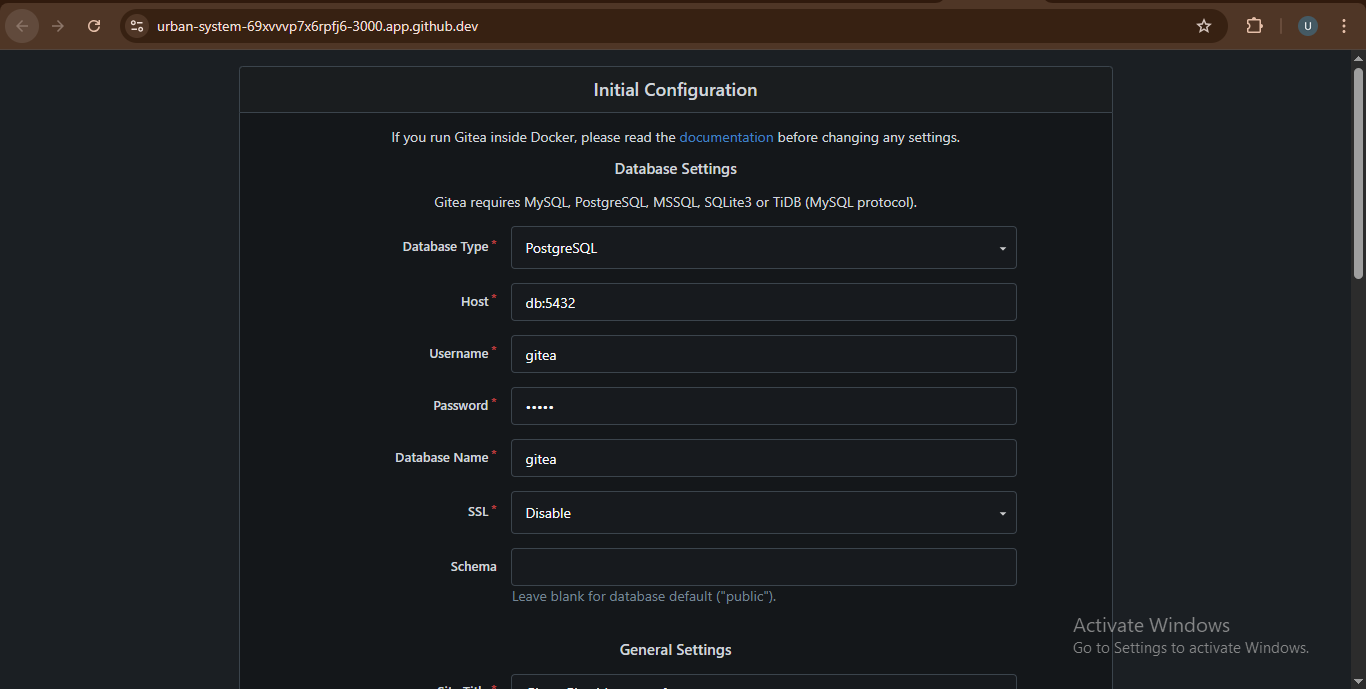
Open the Forked Repo in GitHub Codespaces



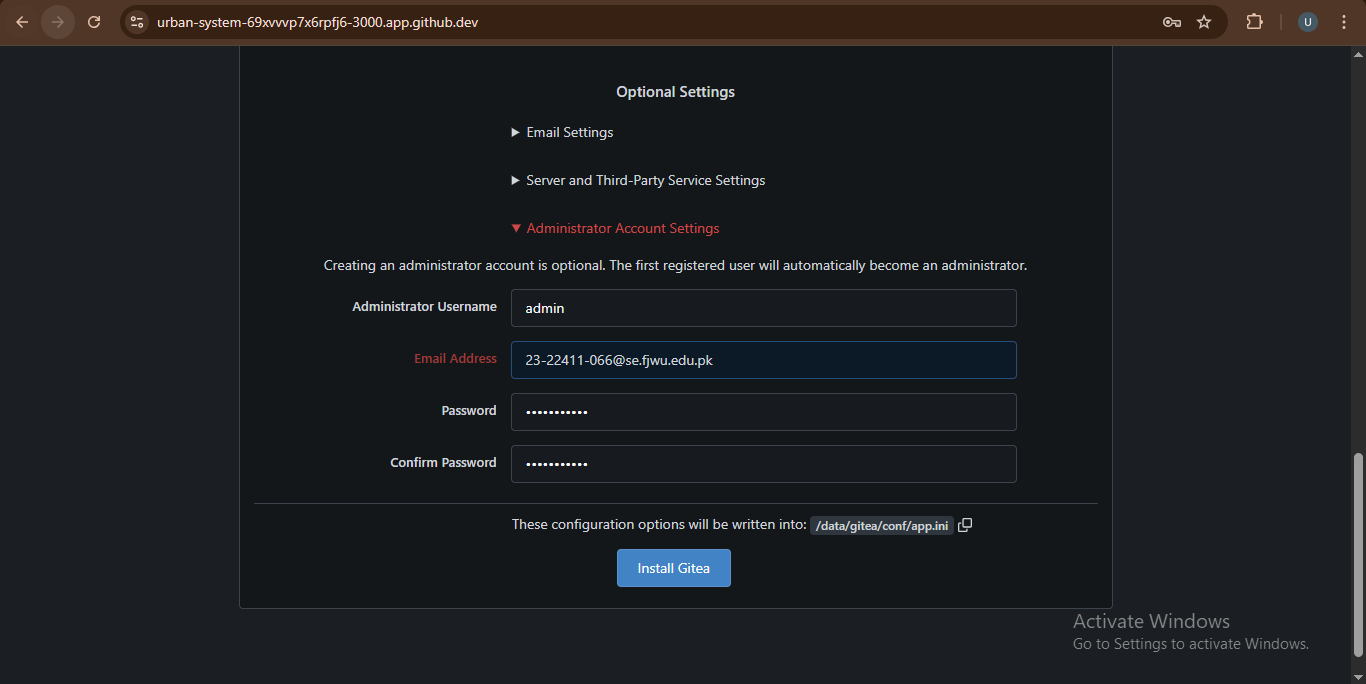
Start Gitea with Docker Compose



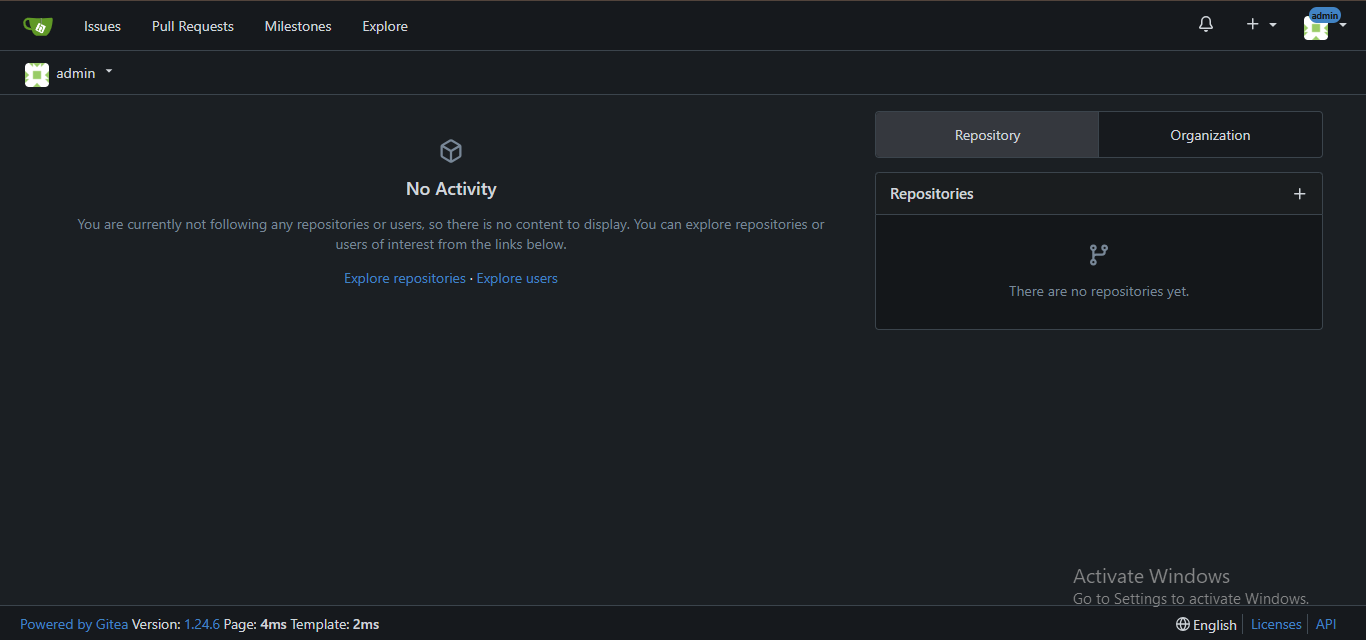
Access Gitea Web Interface



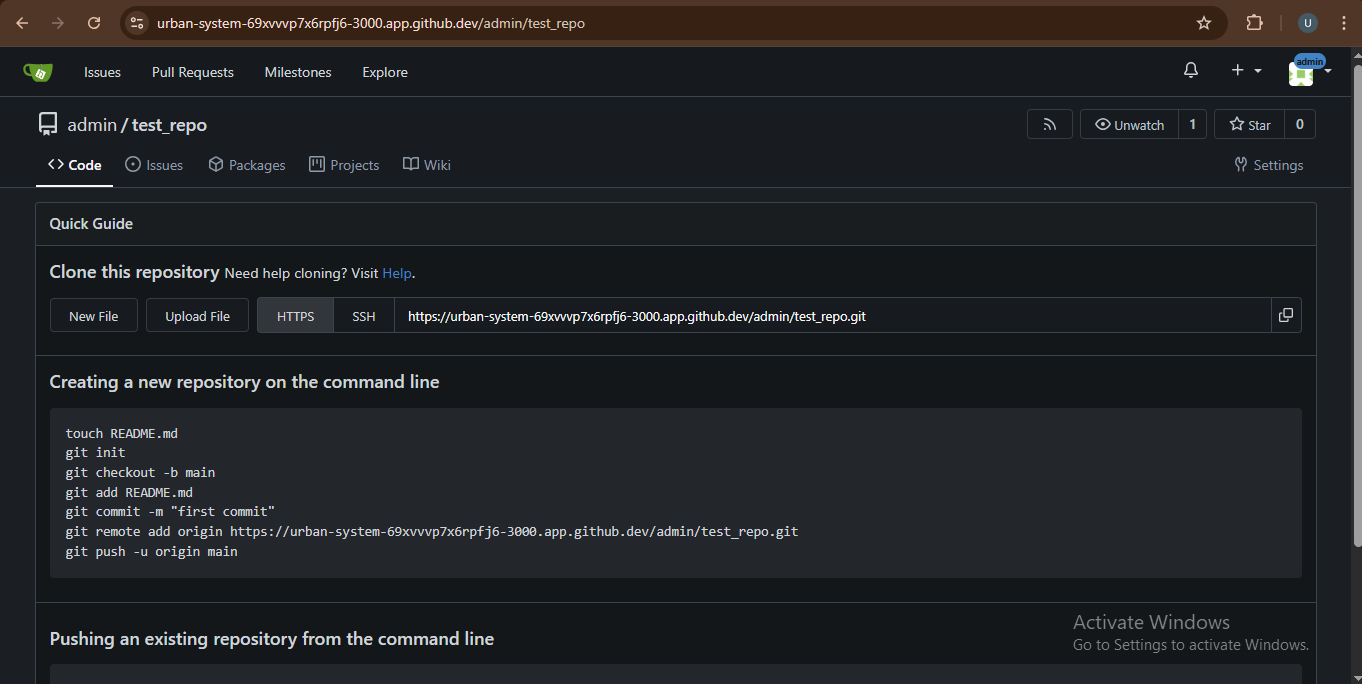
Install Gitea



Log In

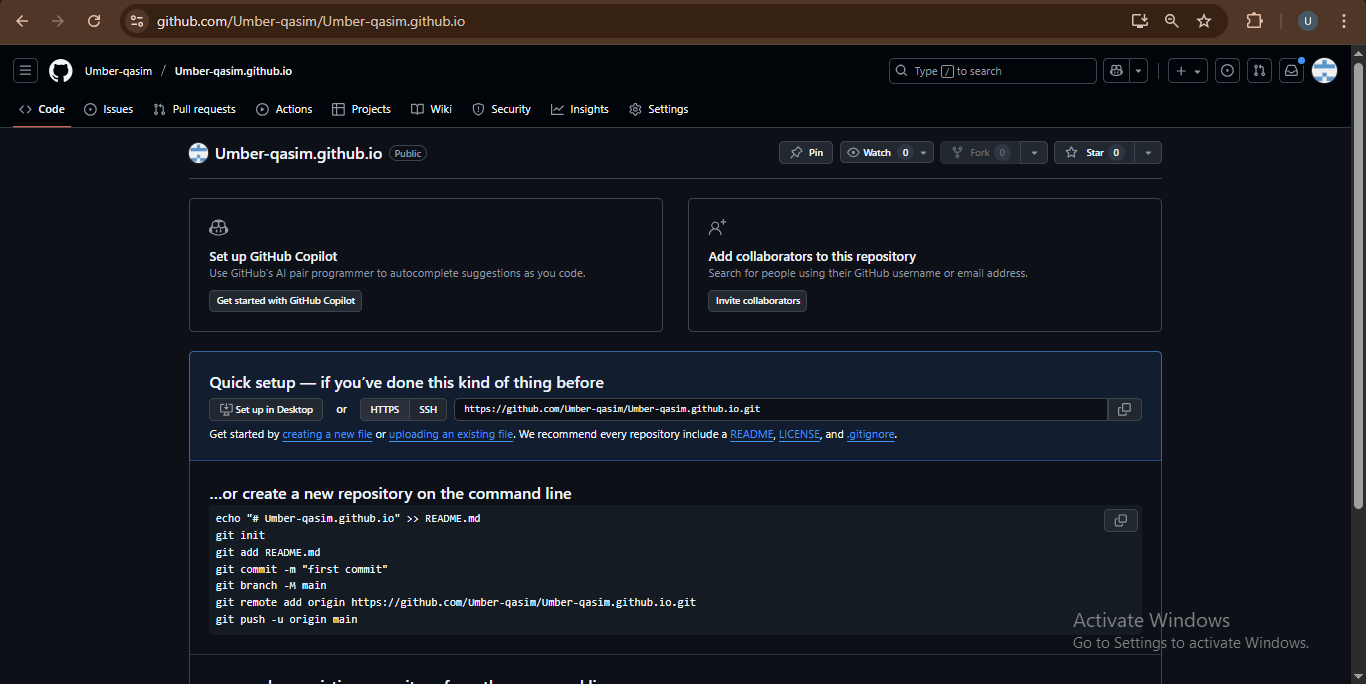


Create a New Repository

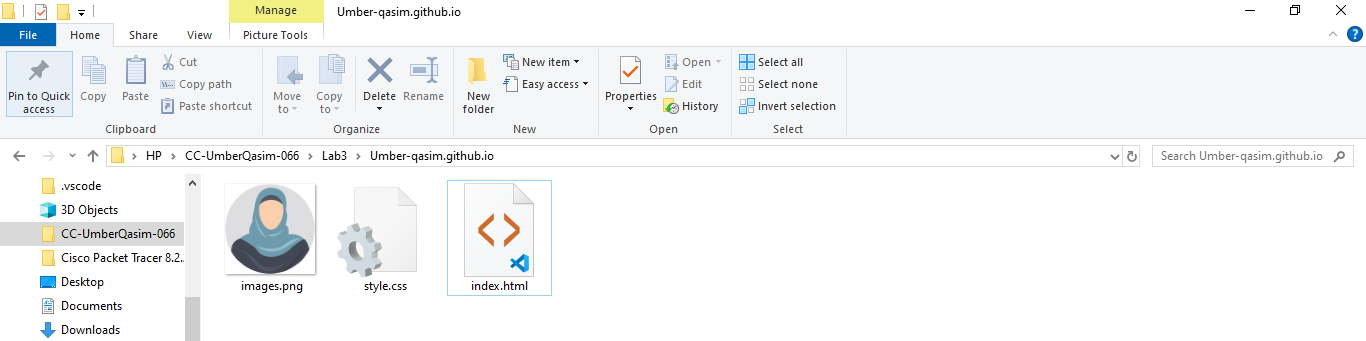


**Task#11: Creating a GitHub Pages Portfolio Site**

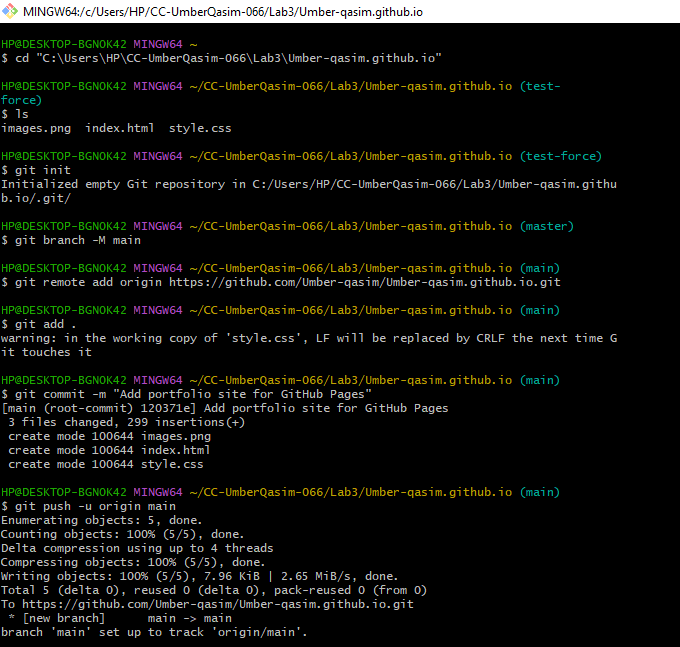
Create a GitHub Pages Repository



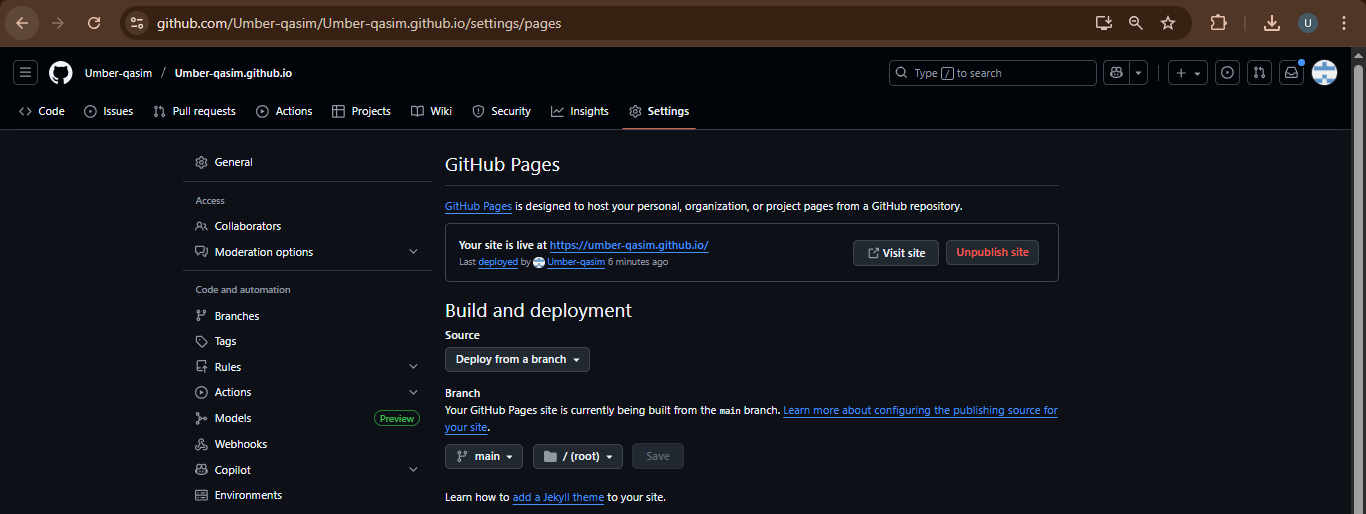
Add Static Website Code



Push the Files to GitHub



Check GitHub Pages Settings



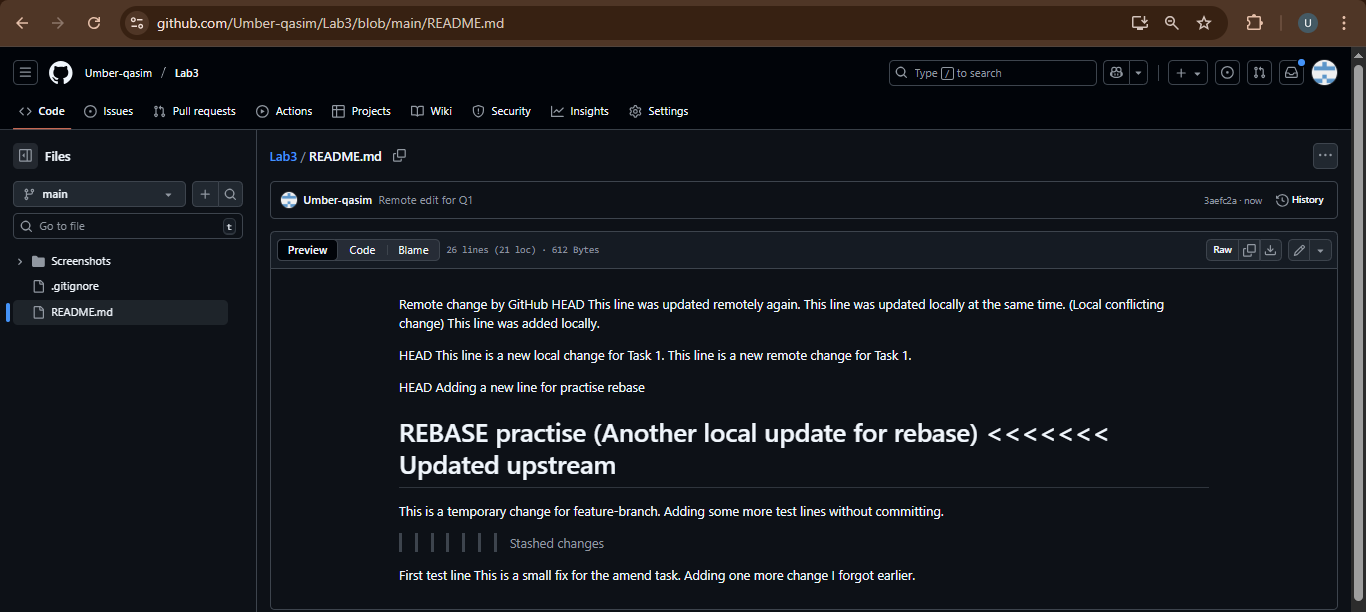
Visit Your Live Site



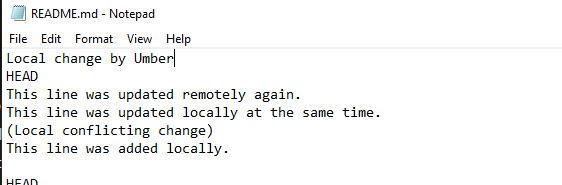
**Exam Evaluation Questions**

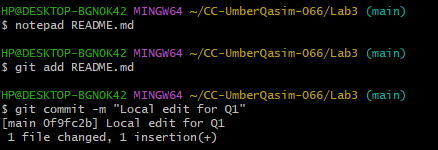
1. **Local vs Remote Conflict Resolution**

Make a Remote Edit on GitHub

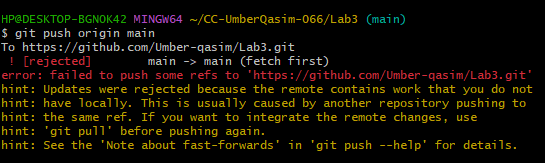


Make a Local Edit (to create a conflict)

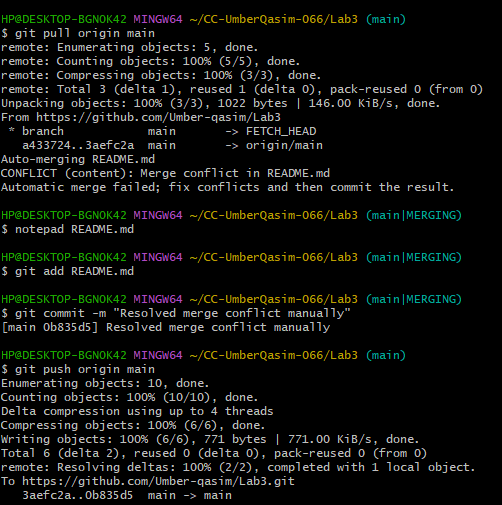




Try to Push

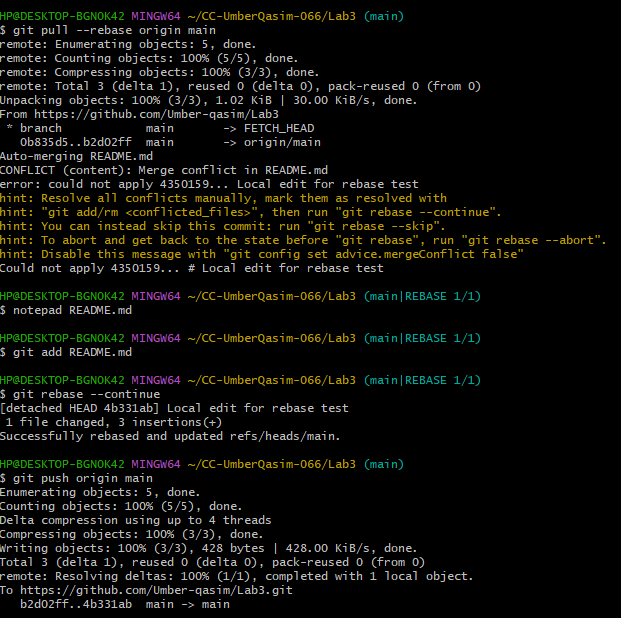


Resolve Conflict using git pull (Merge)



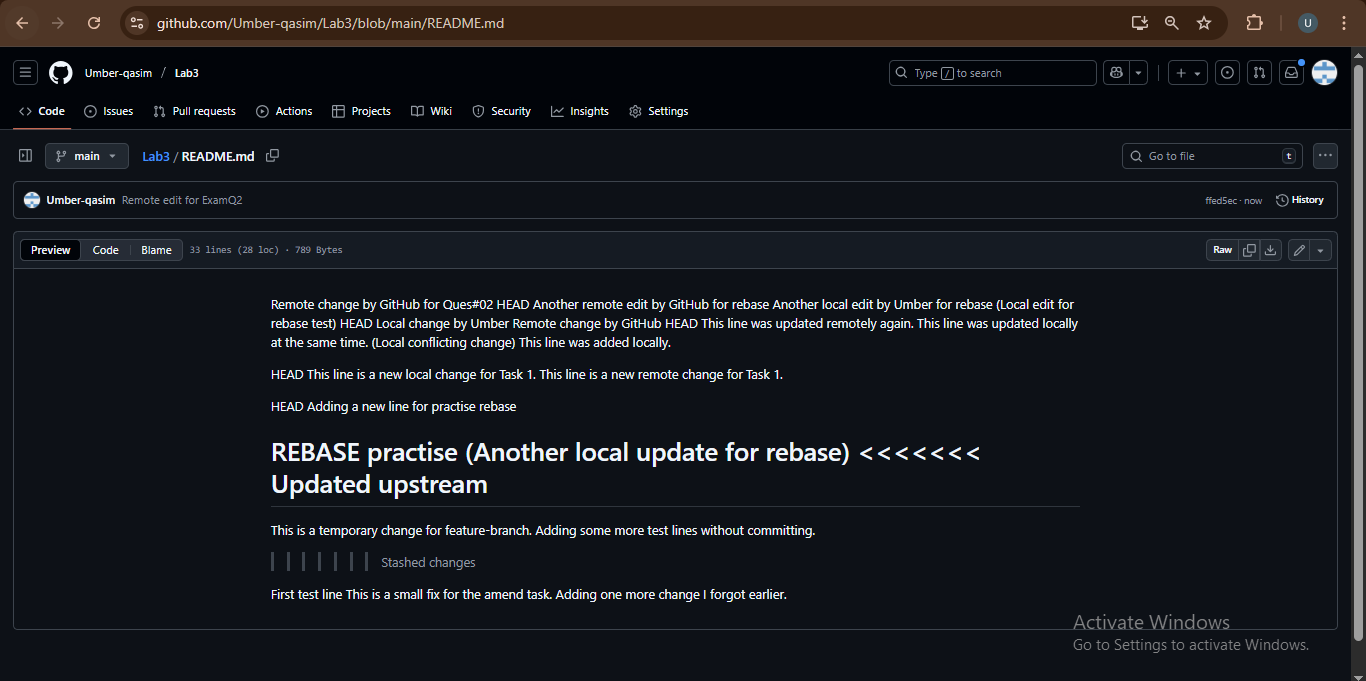
REBASE resolution

**Perform same work as above till push command**

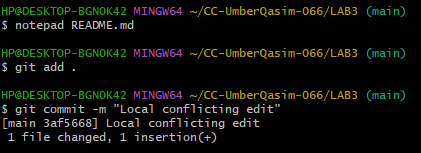


1. **Manual Merge Conflict Handling**

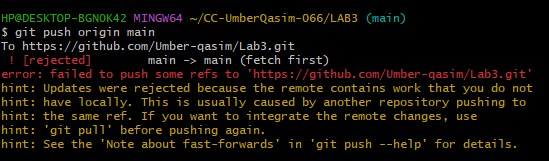
Remote change (on GitHub)



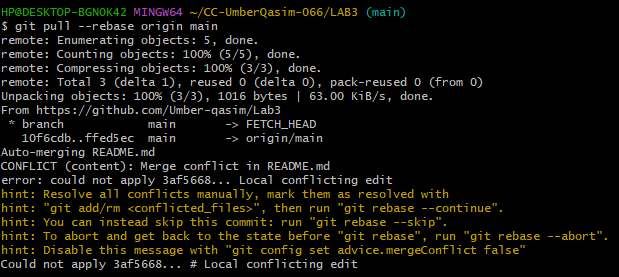
Local change (on your PC)



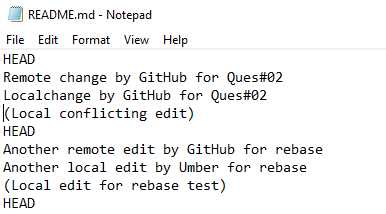
Push and see conflict



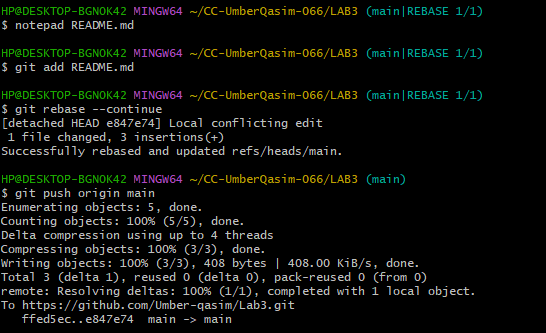
Pull with rebase



Resolve conflict manually

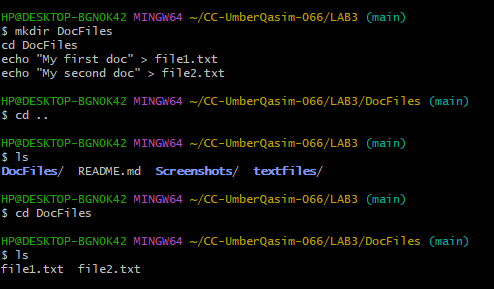


Mark as resolved and finish rebase

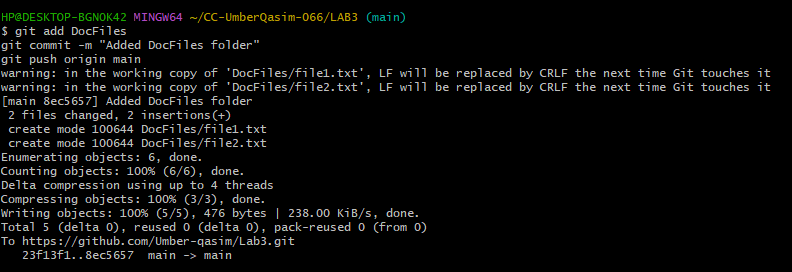


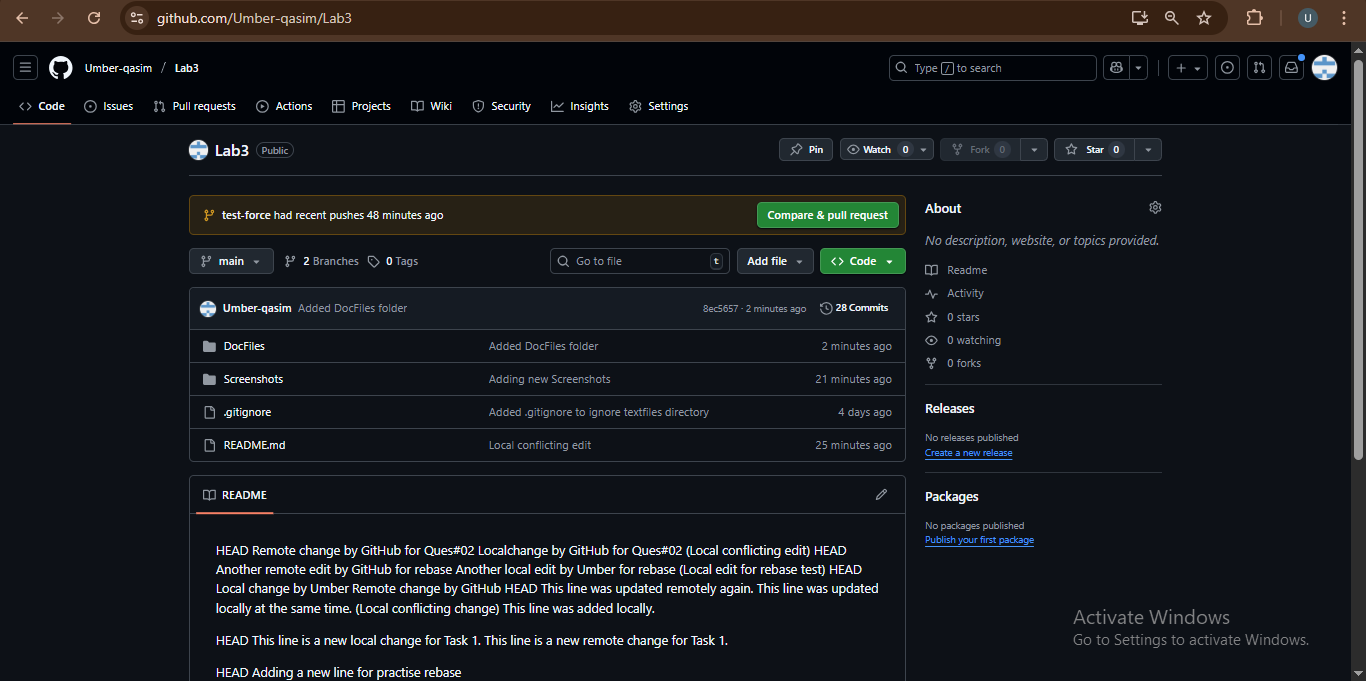
1. **Managing Ignored and Tracked Files**

Create folder and files

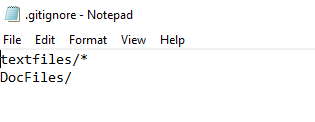


Add, commit, push (to GitHub)

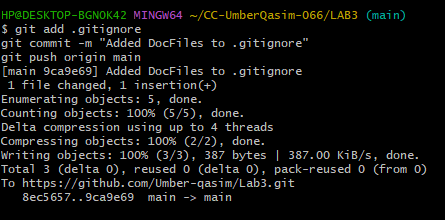




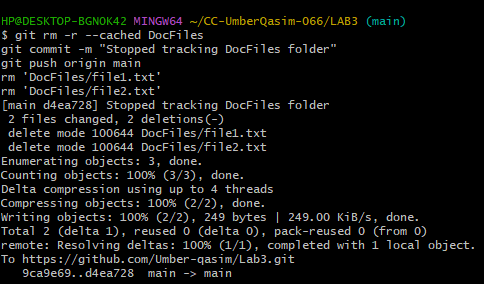
Add to .gitignore



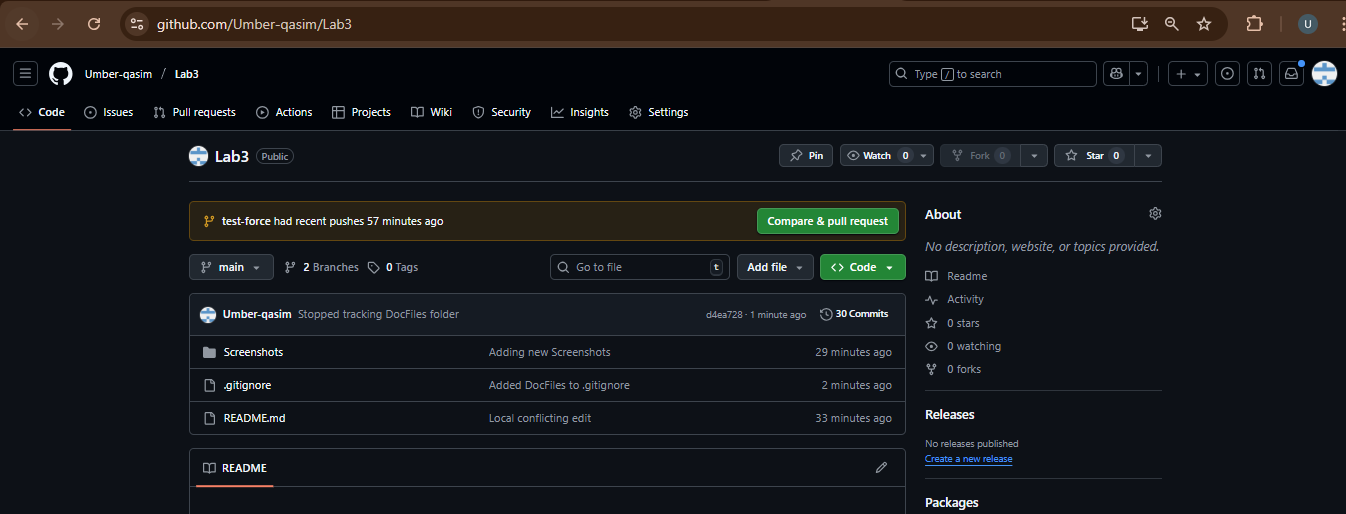
Commit and push .gitignore



Remove folder from Git tracking

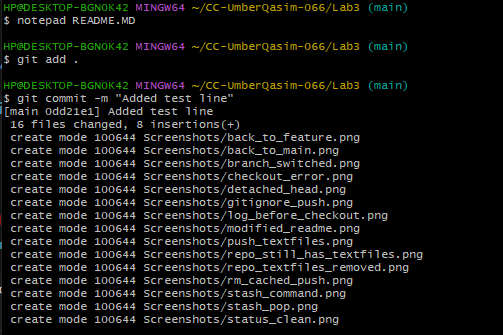


Verify

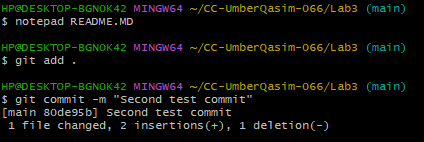


1. **Commit History Manipulation and Recovery**

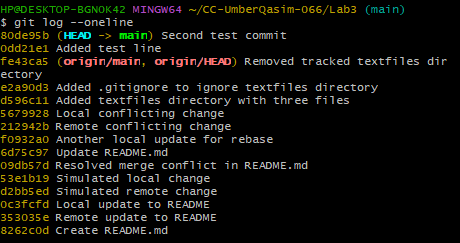
*Add a new line in any file and commit it*



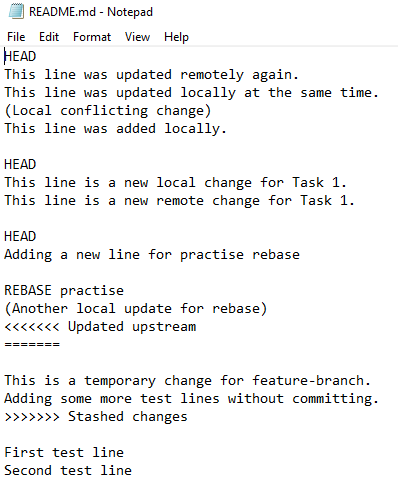
*Add another change and commit again*



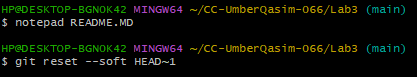
*View commit history before reset*



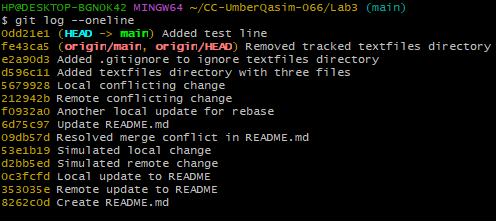
*Confirm both changes are in the file*



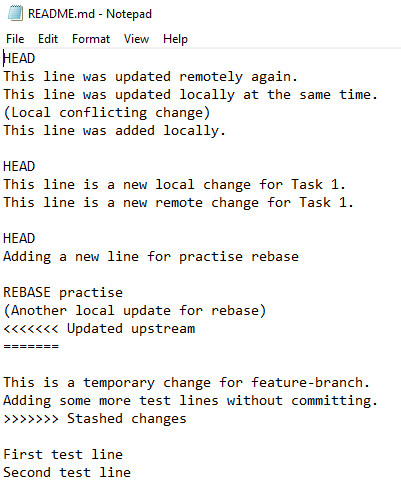
*Perform a soft reset*



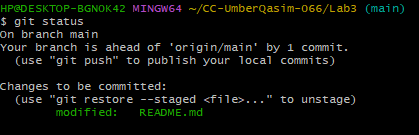
*Check commit history again*



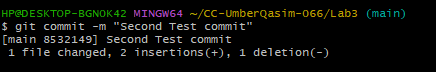
*Check file content after soft reset*



*Check git status*



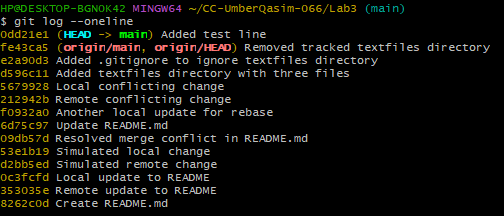
*If it shows staged changes, you can re-commit:*



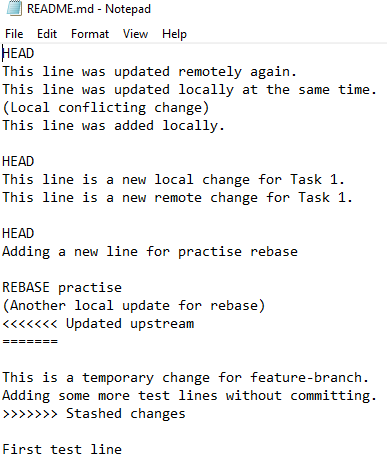
*Perform a hard reset*



*Check commit history again*



*Open file again*



*Check git status*

