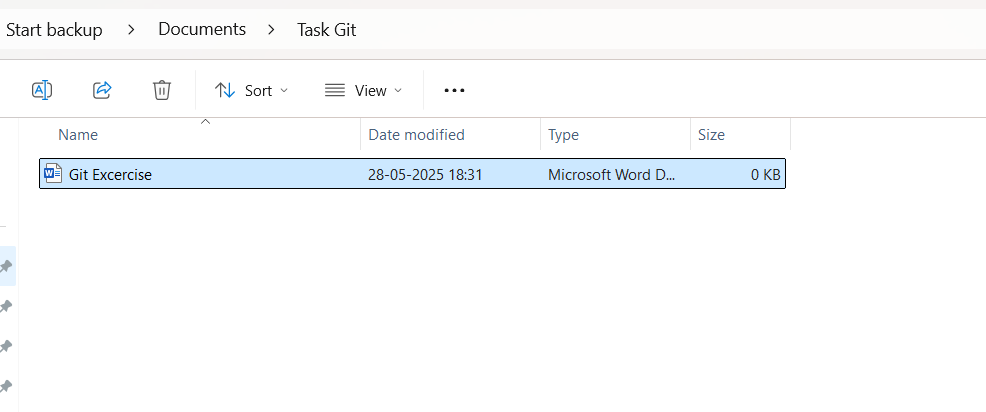
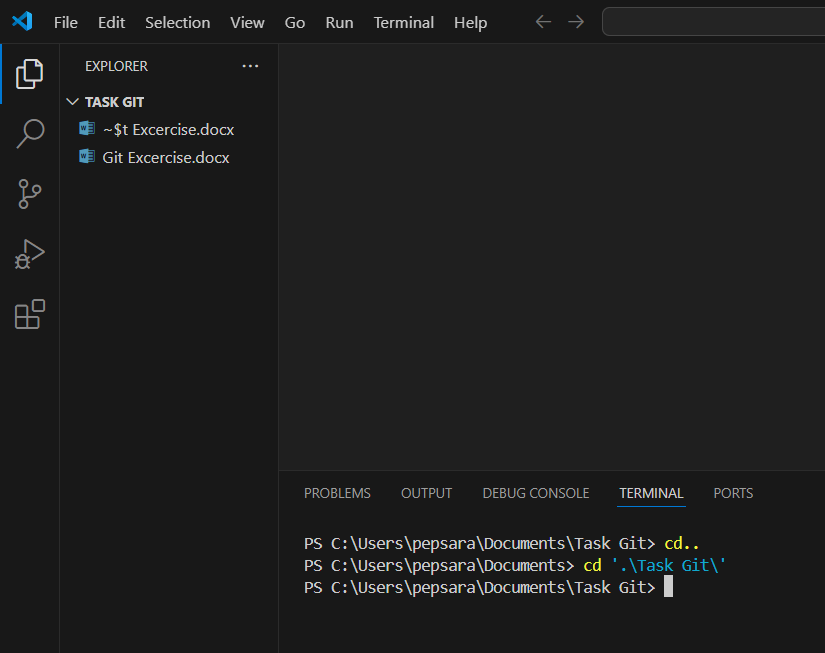
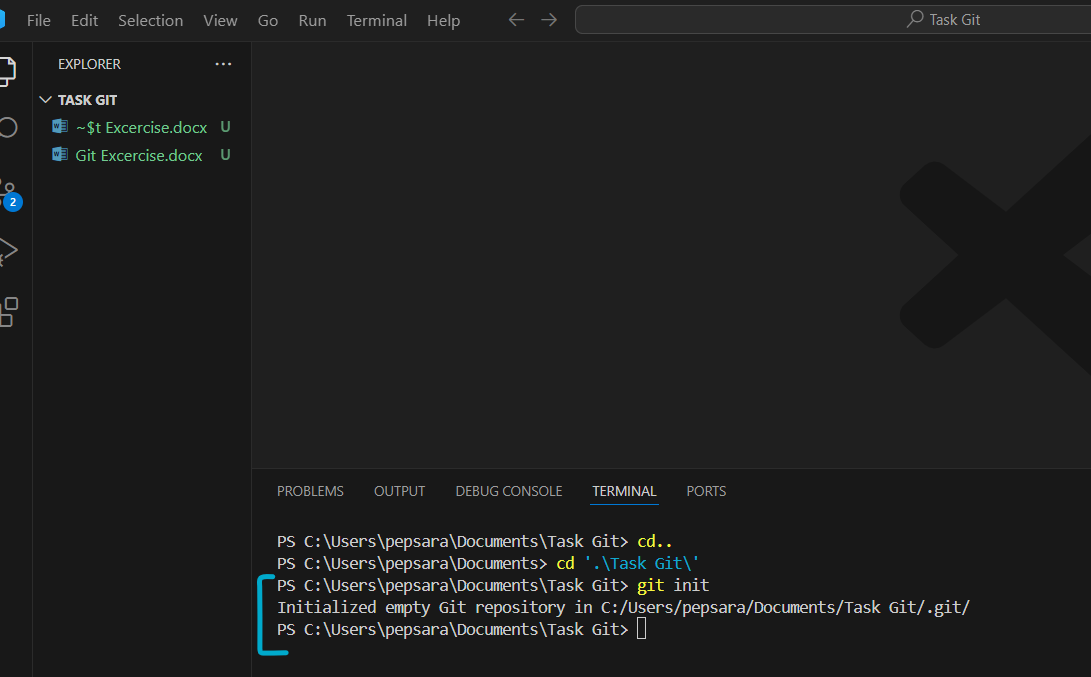
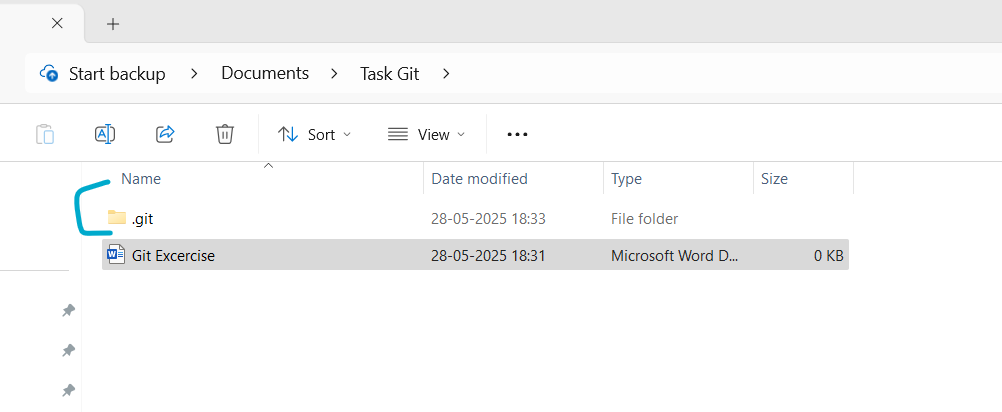
1. Create a Project Folder



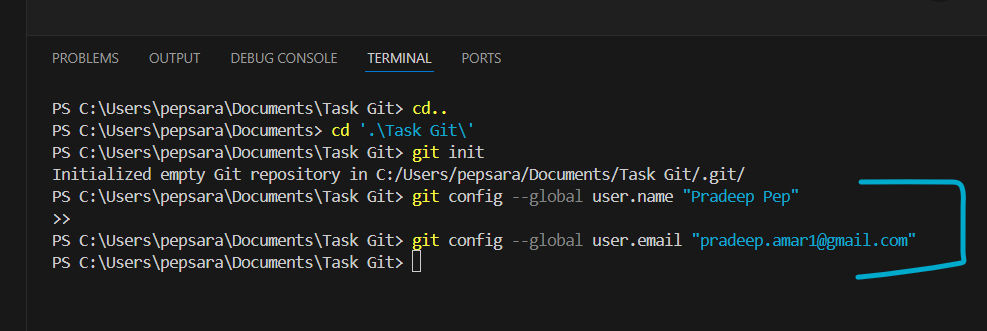


1. Initialize a repository

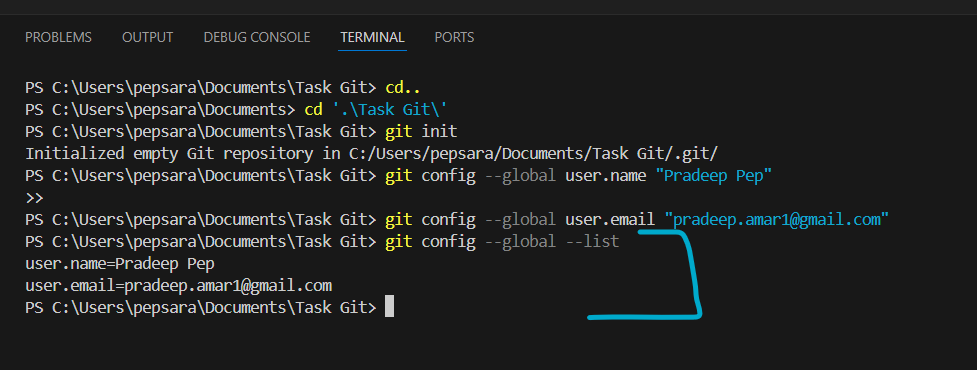




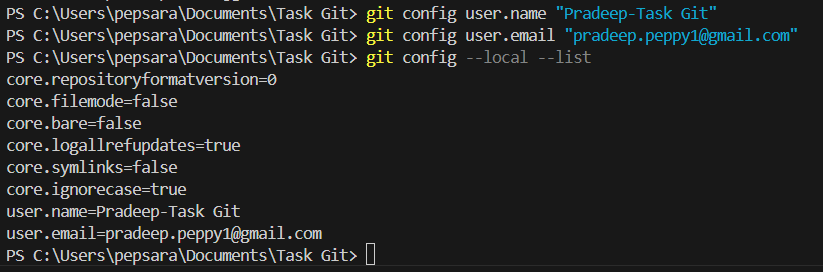
1. Setup User details globally



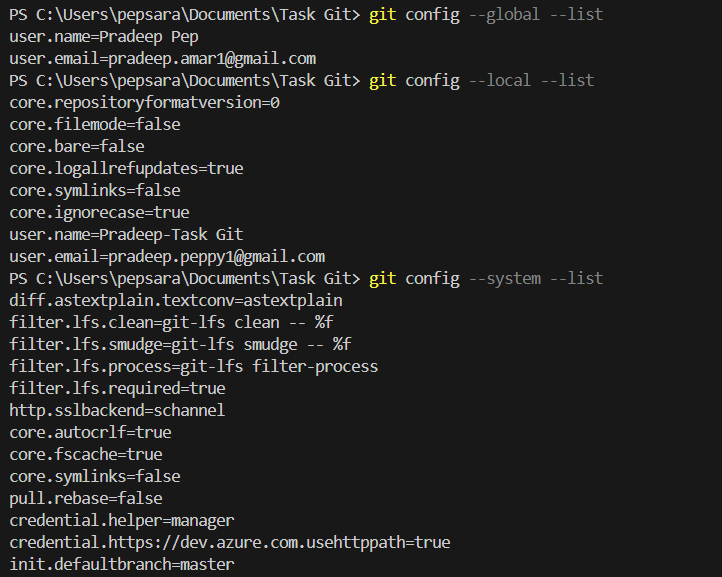
View it



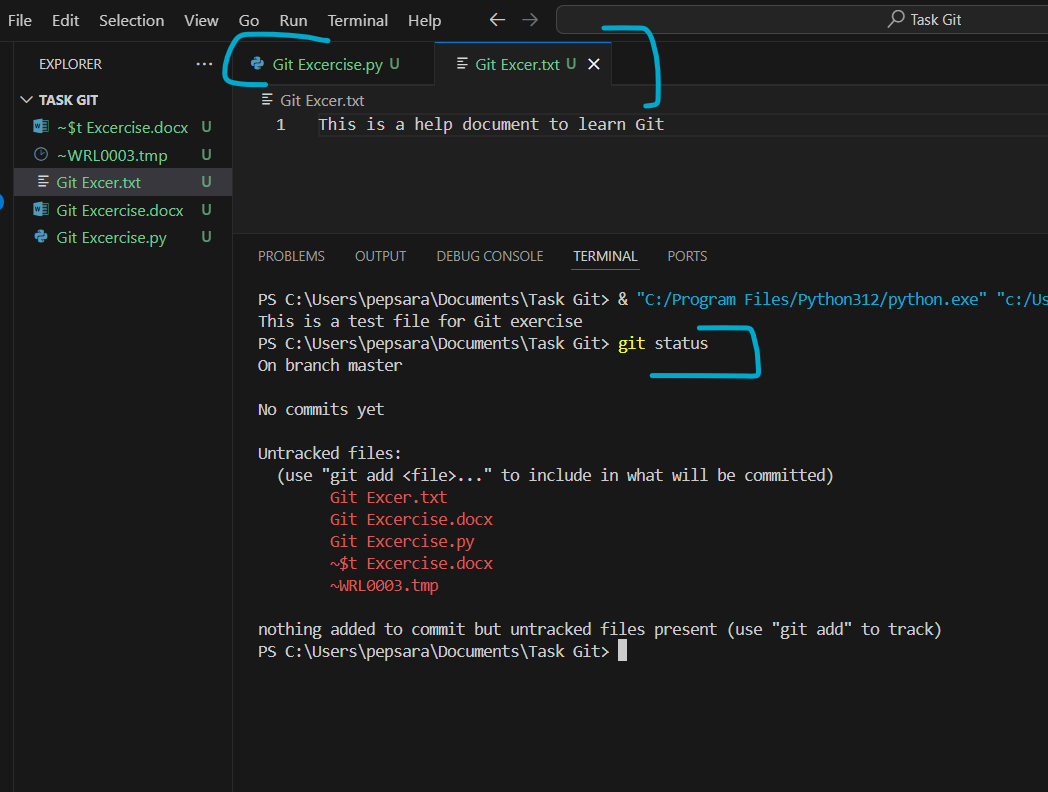
1. Setup & View Project specific user details



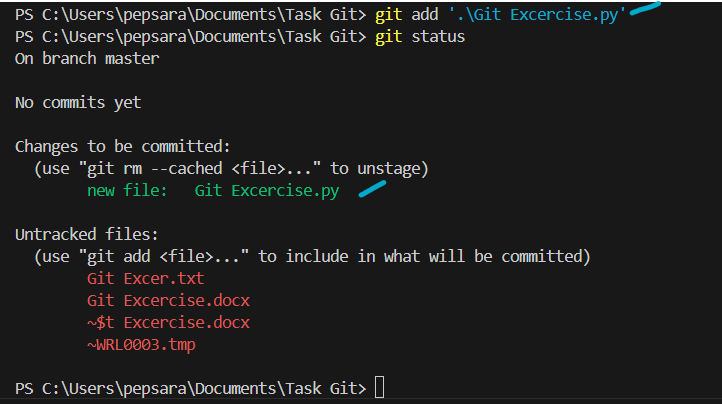
1. View the setup configuration made



1. Add specific files [ Create 2 files + Use add to stage only 1 file ]

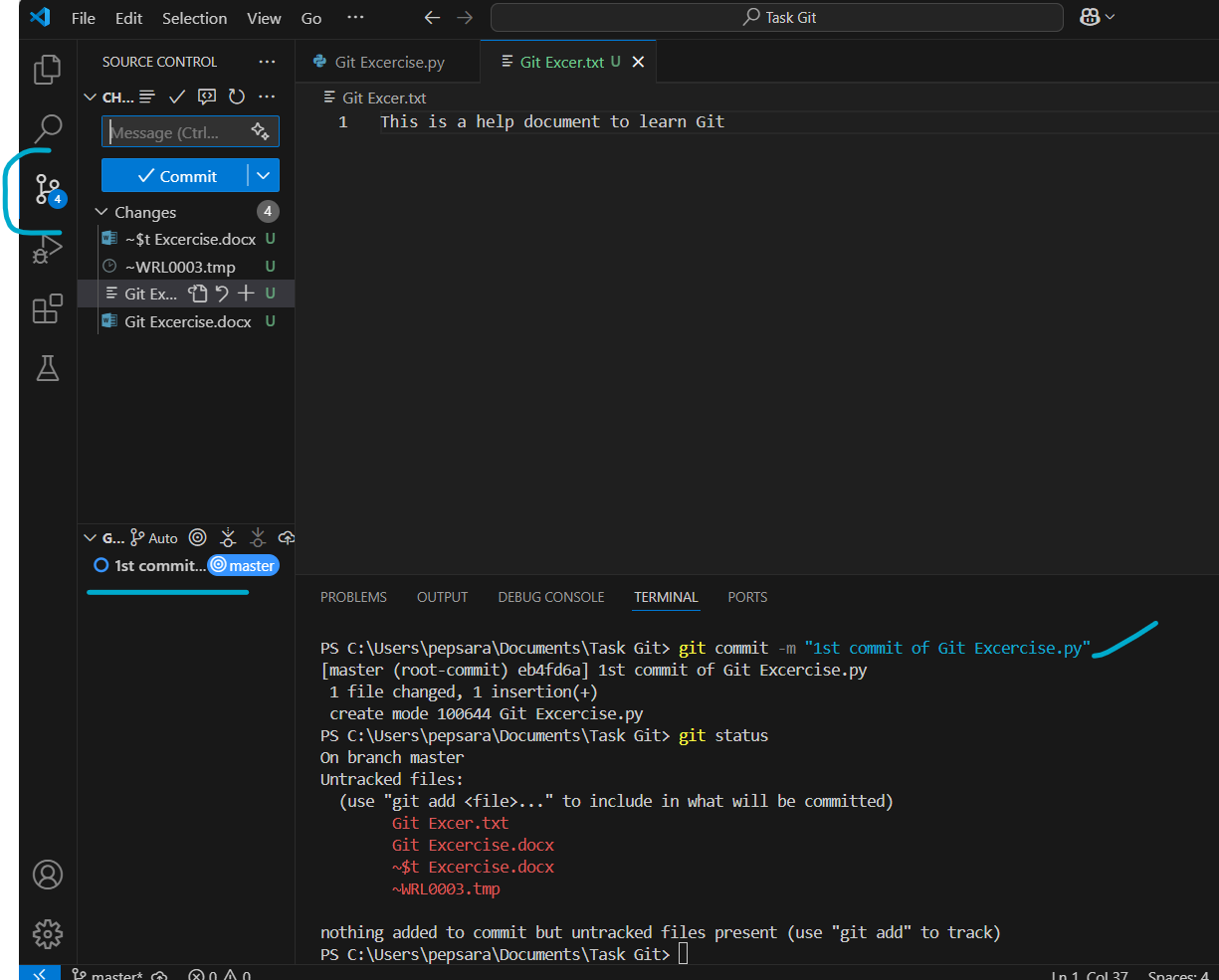


Stage all files except 1.

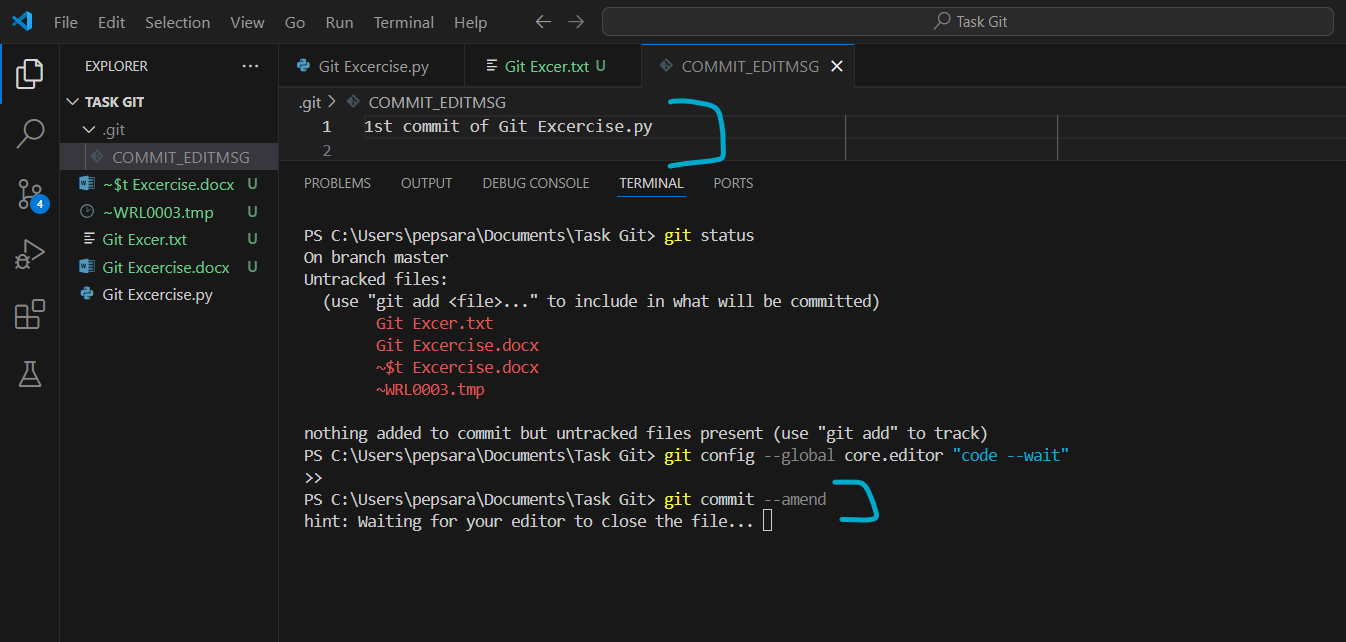


1. Committing Changes

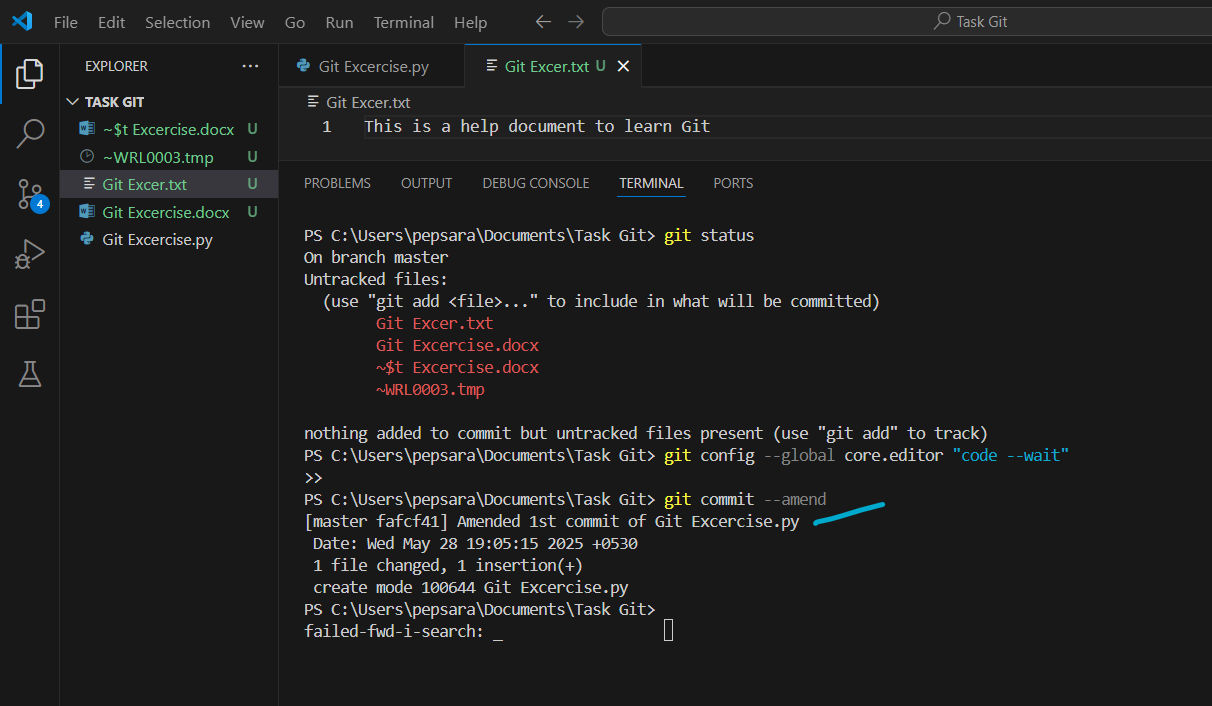
Commit with a message



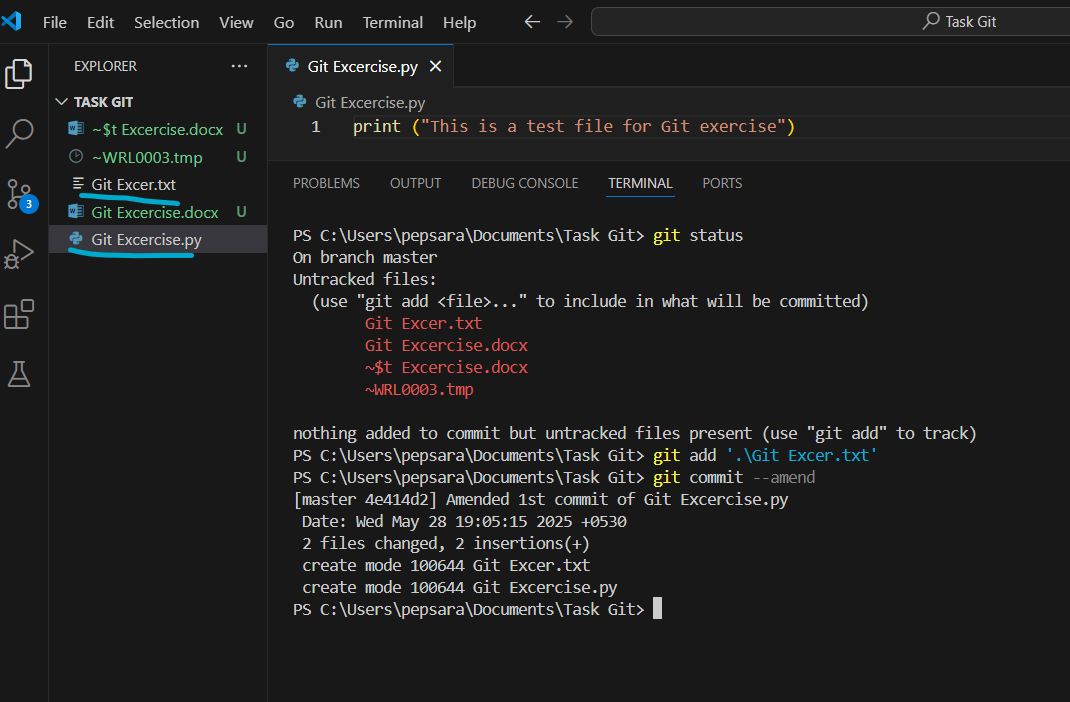
1. Amend a commit (Message)

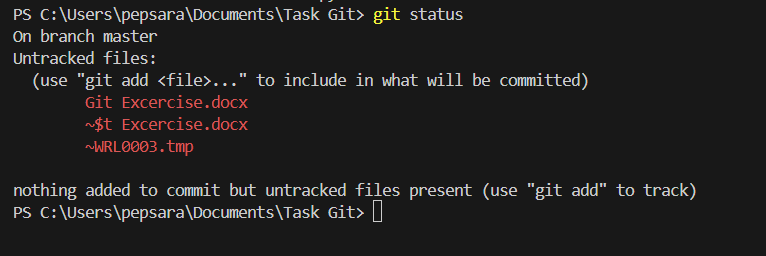


Ctrl+S & Ctrl+W after Changing the Commit message in Editor

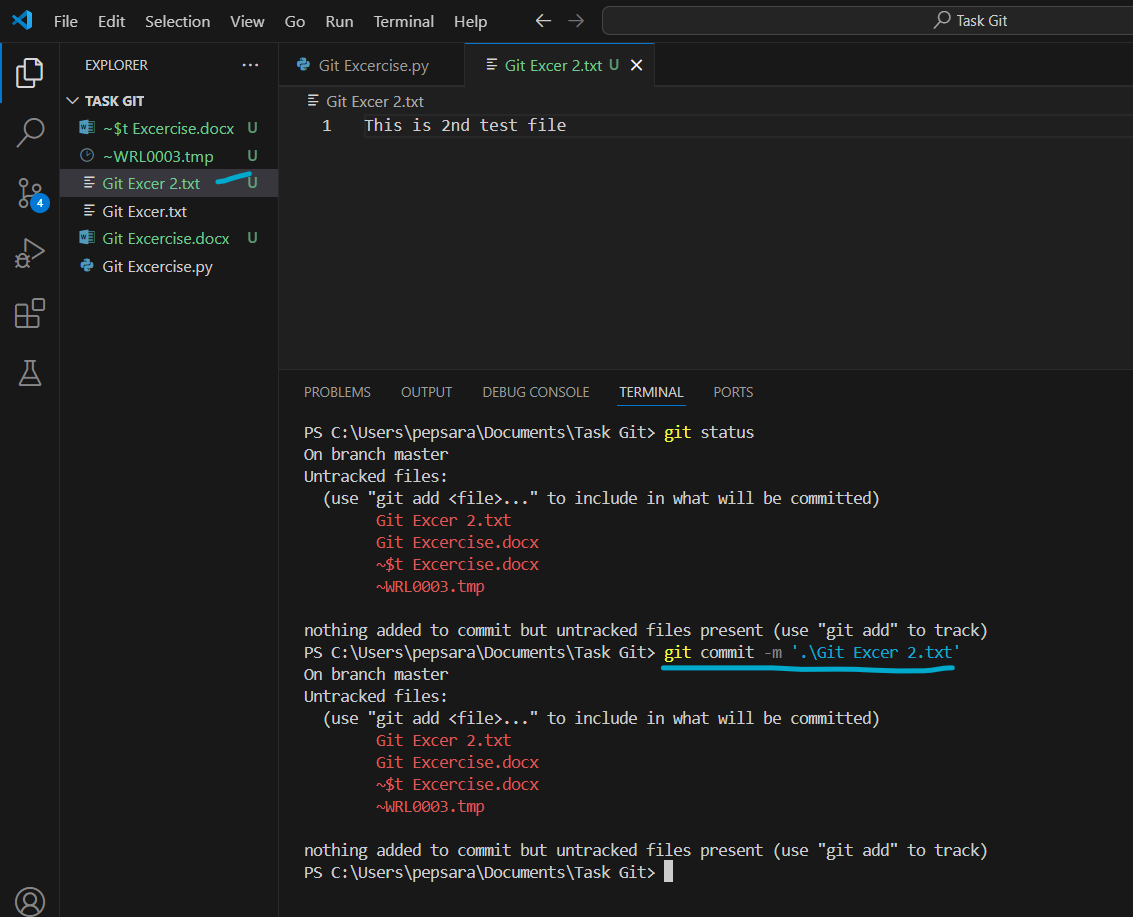


Amend a commit (File)



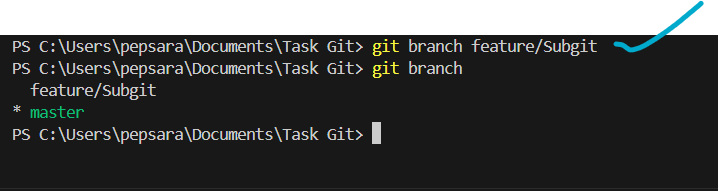


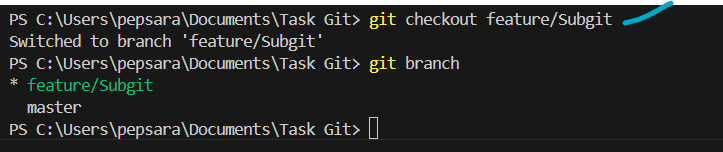
1. Commit without staging



**Branch Management:**

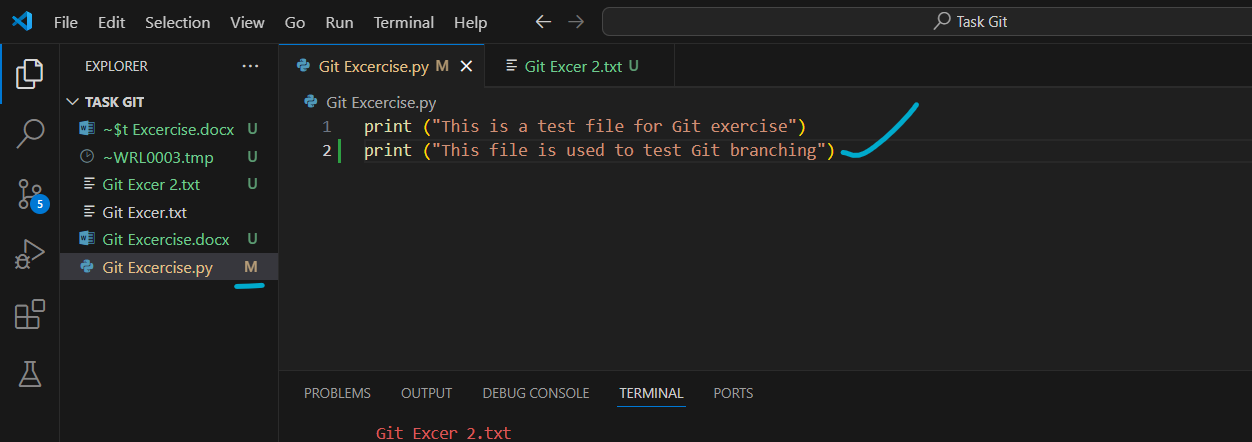
1. Create a Branch





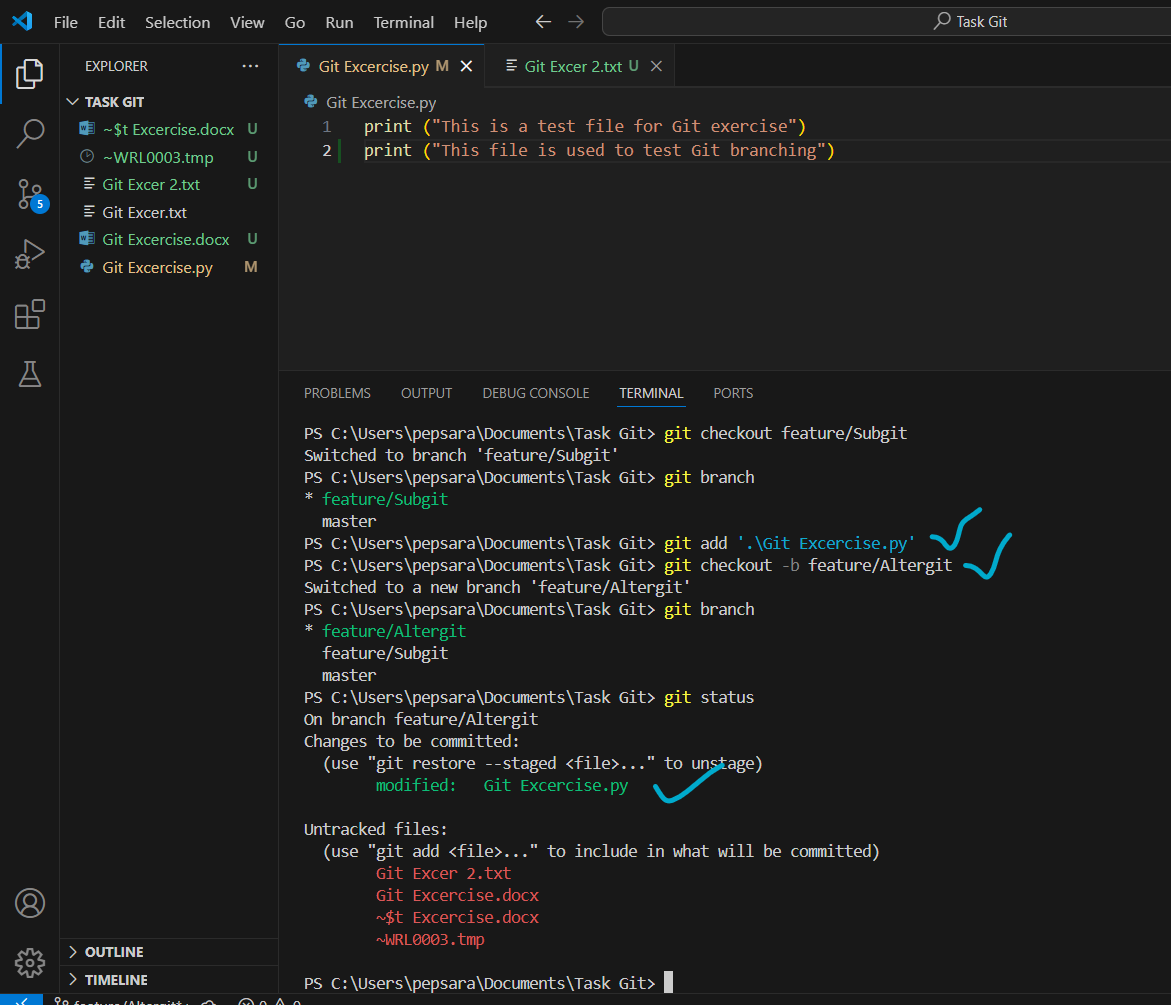
1. Work on this feature branch without affecting Master branch

Problem:

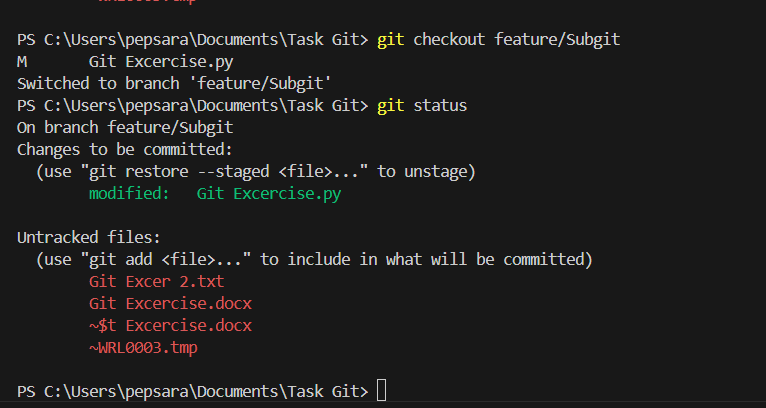
Created a new addition in code on a sub Branch (Feature/Subgit)

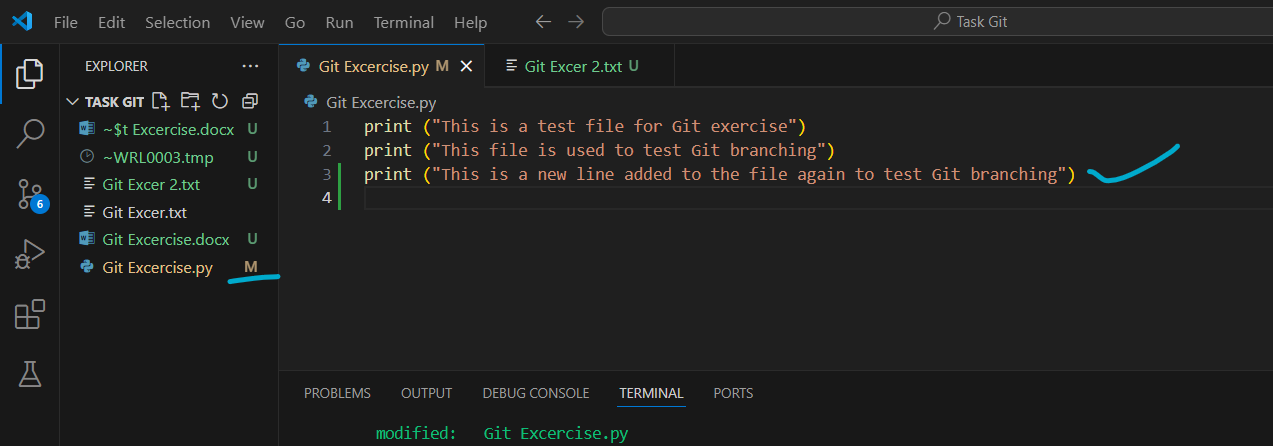
Added the file to Staging. And DIDN’T commit.

Now upon creating a new branch & switching to it (Feature/Altergit), I see the changes specific to Subgit in Altergit as well. This has to be avoided, even incase the code is not committed in 1 branch.

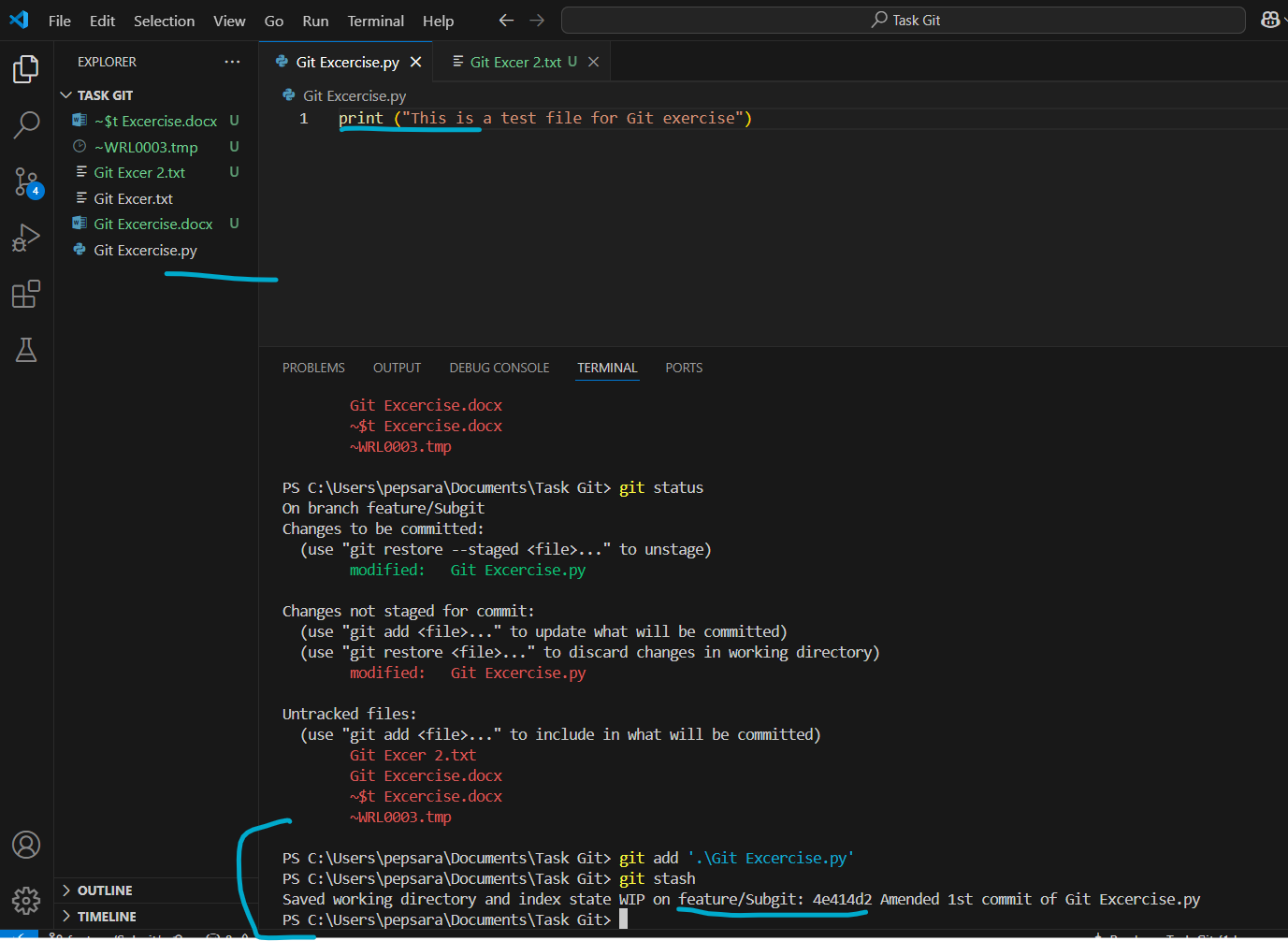


Solution:

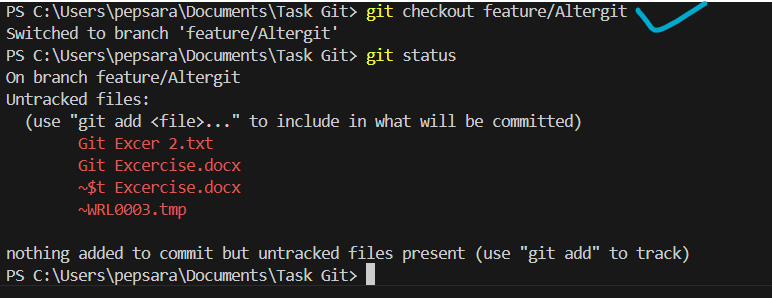


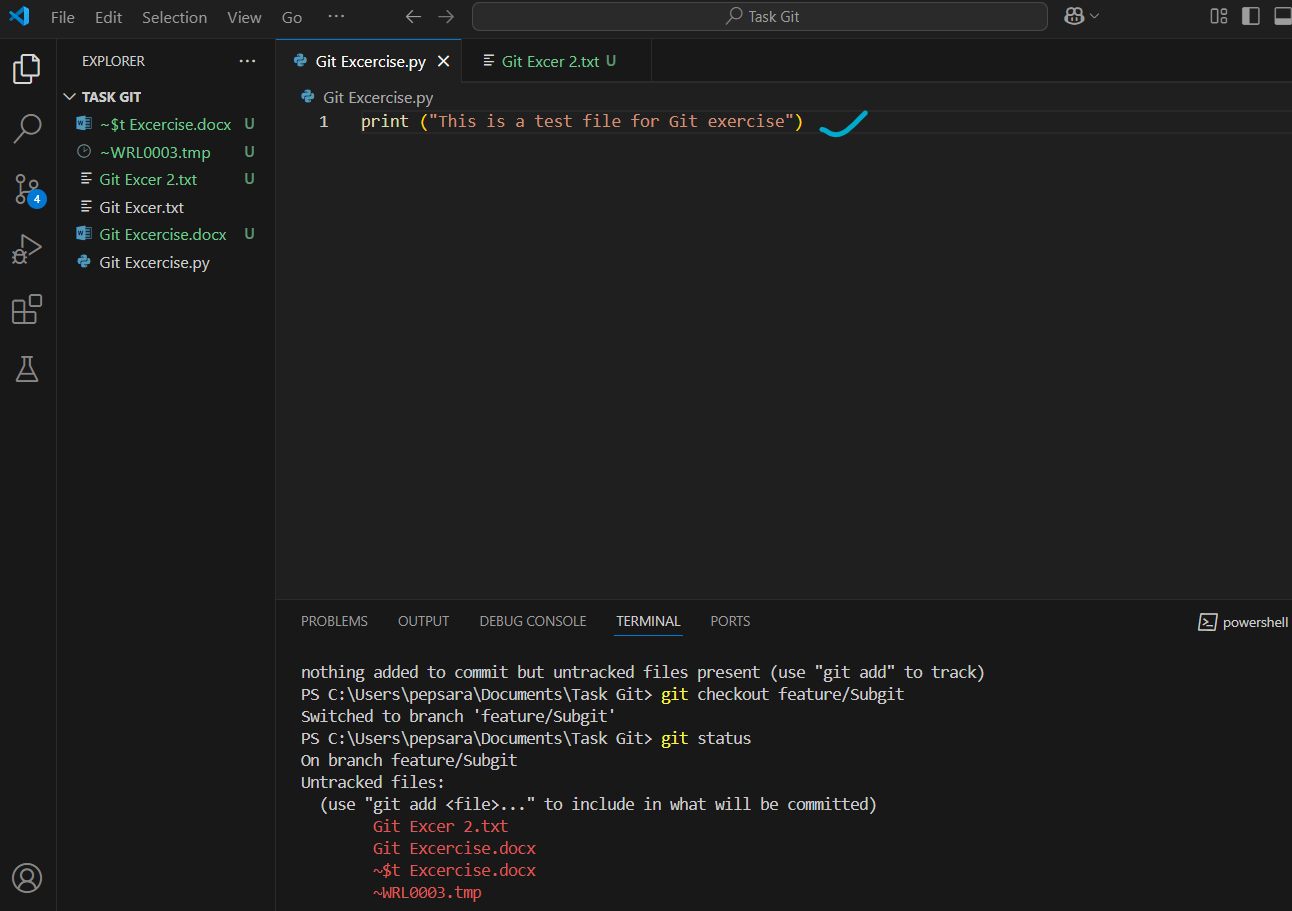


The changes are done in Feature branch 2, but stashed instead of commit

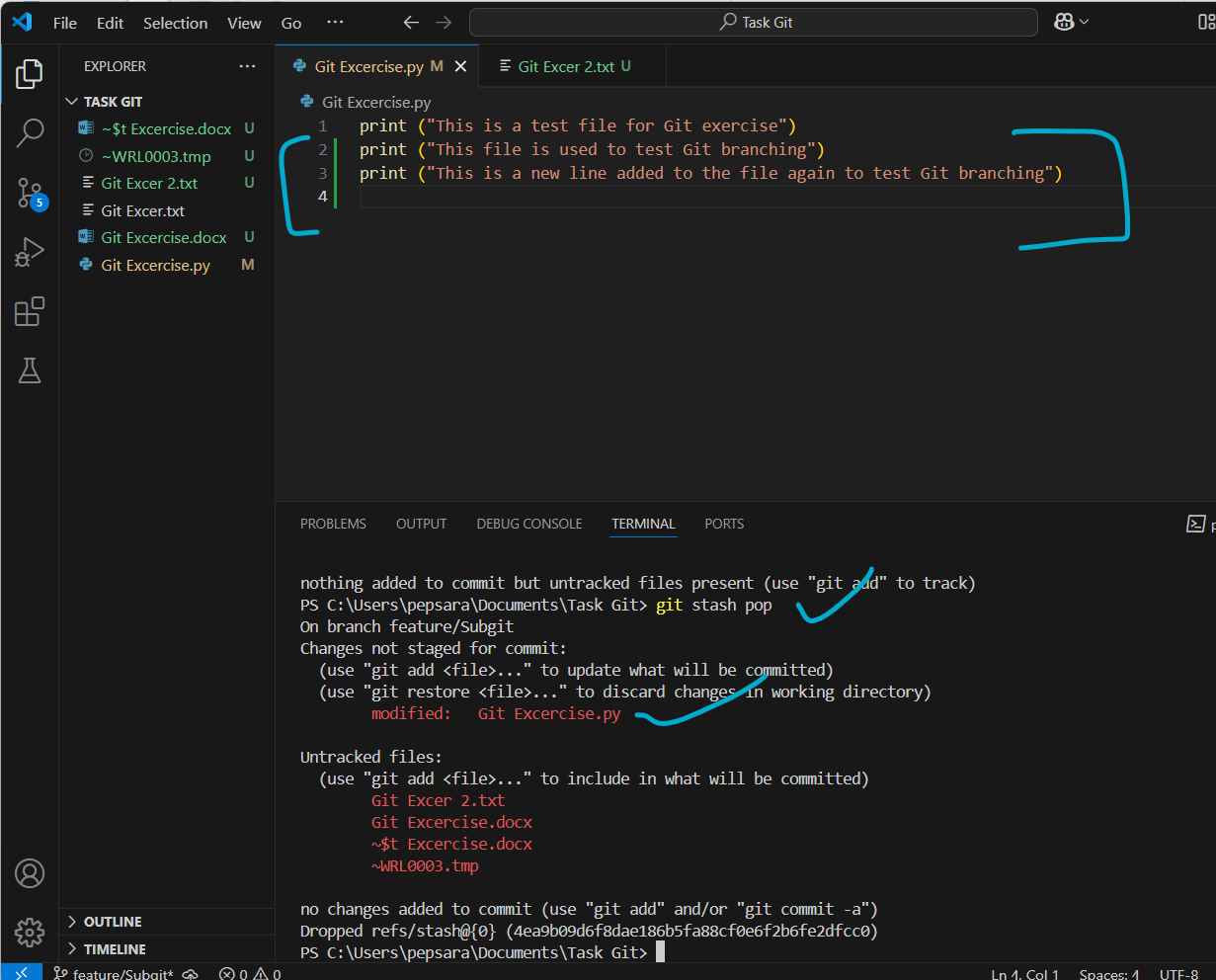


Switch to first branch and you don’t see the staged version here

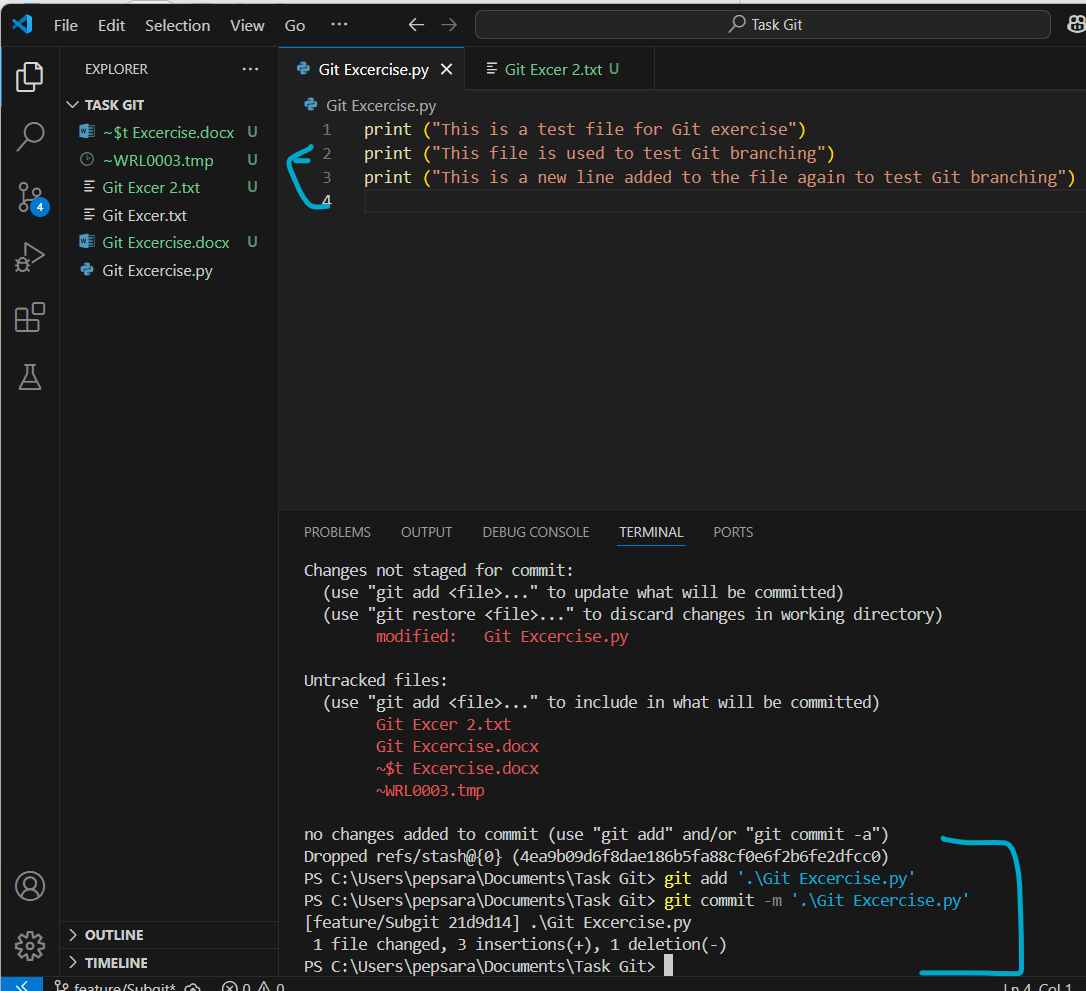


The file also contains the data, which is originally made in that feature branch only.

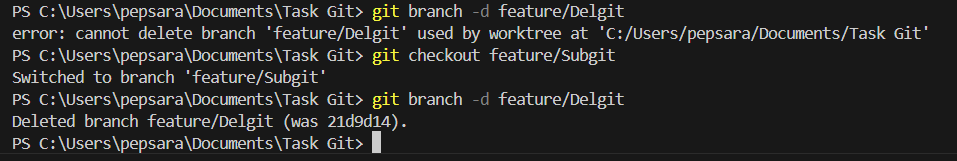
Now switching back to feature branch where u actually made changes & to bring back the changes from Stash. Use ‘Stash Pop’



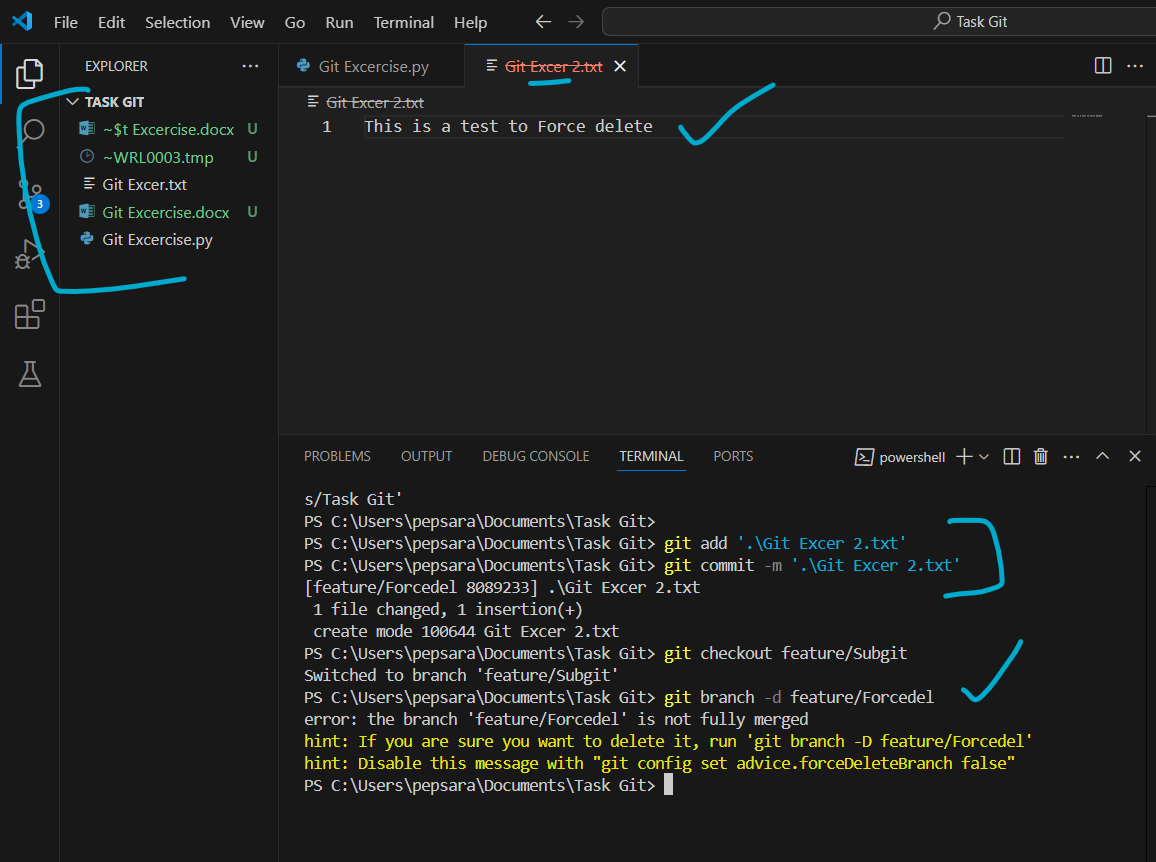
Now you can commit the changes in this Feature branch alone



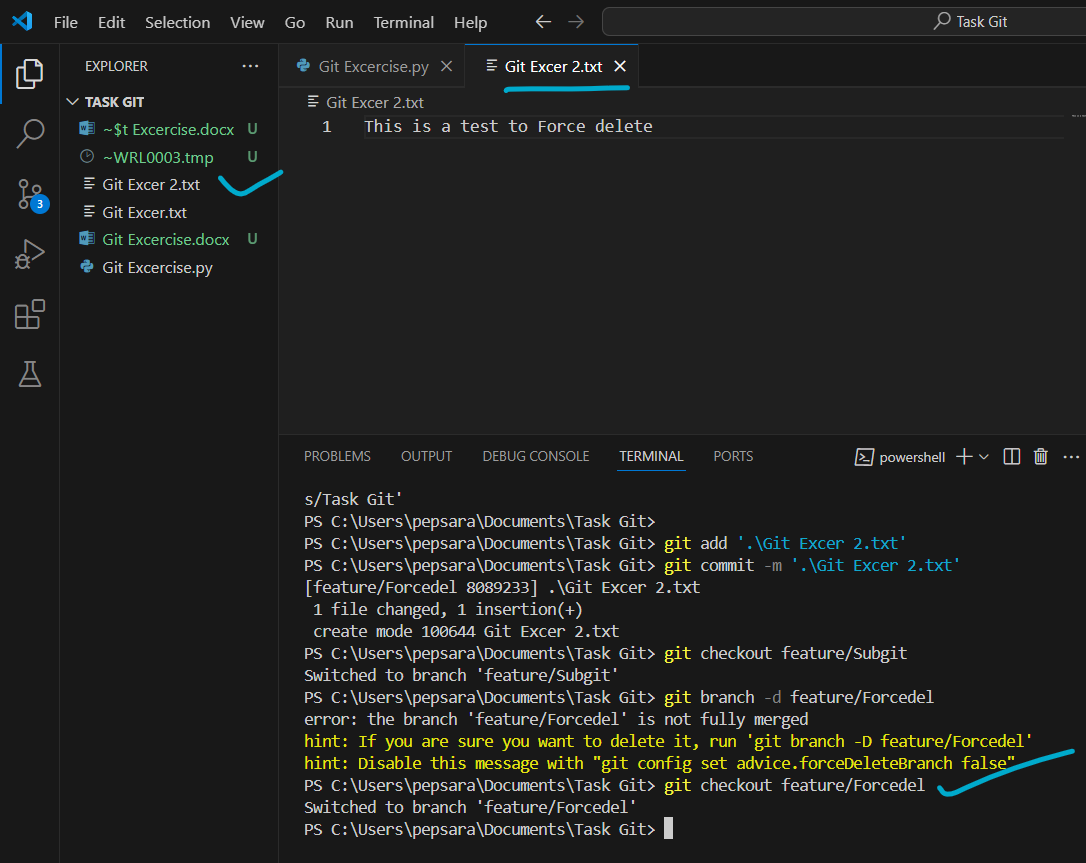
1. Delete a branch

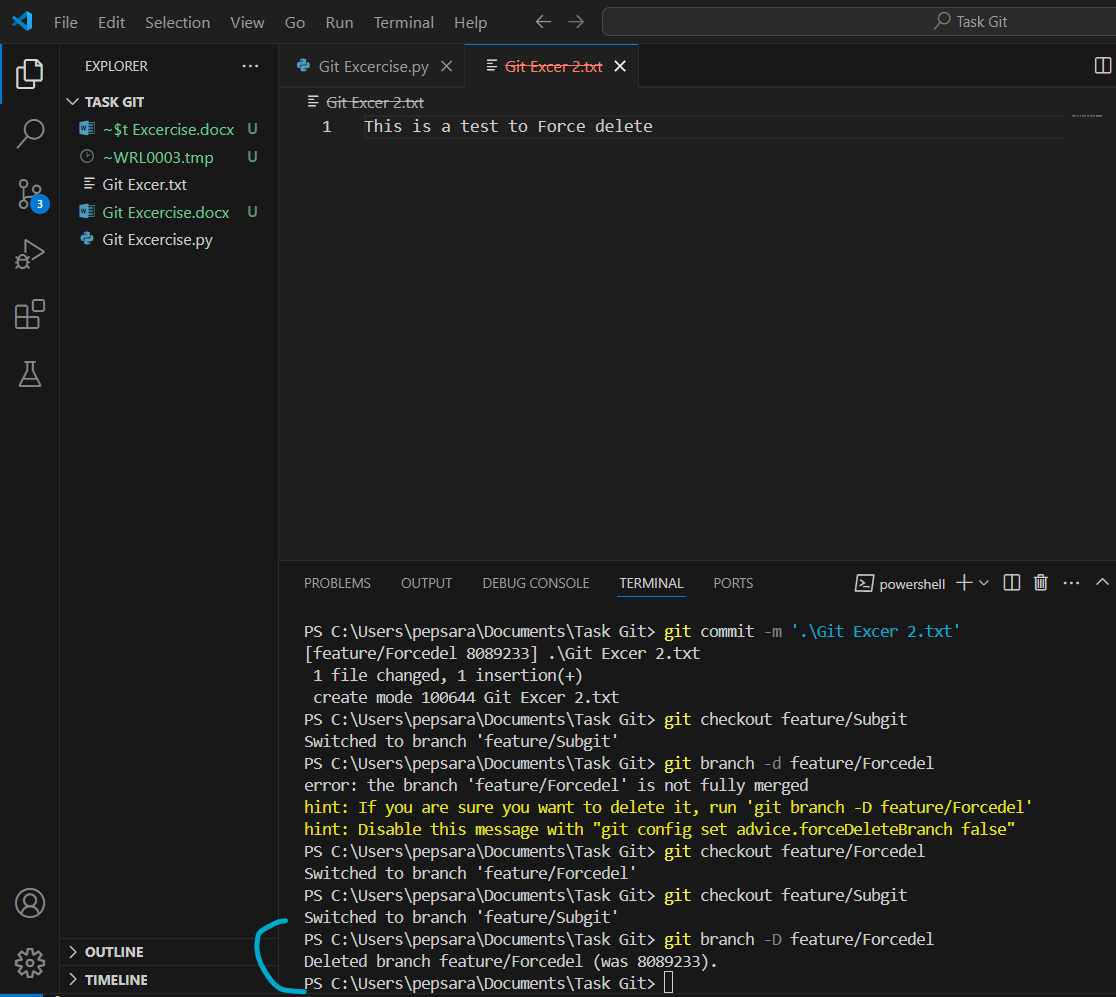


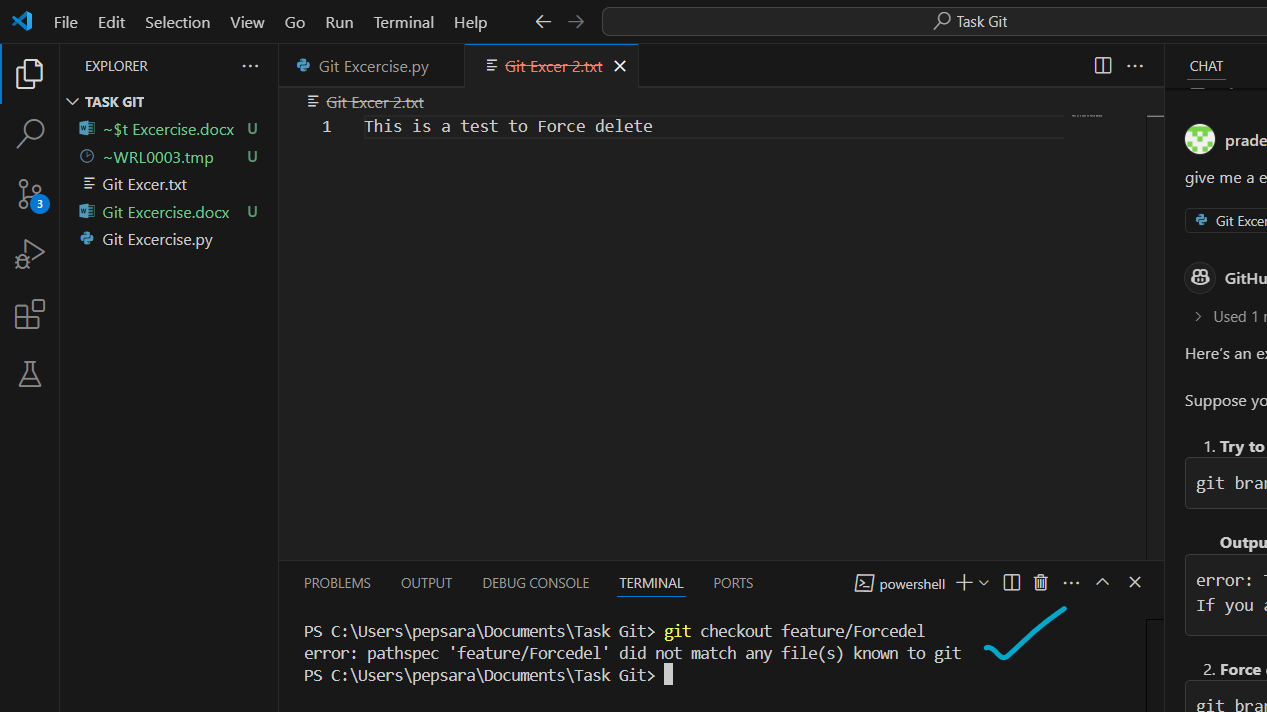
1. Force deletes a branch



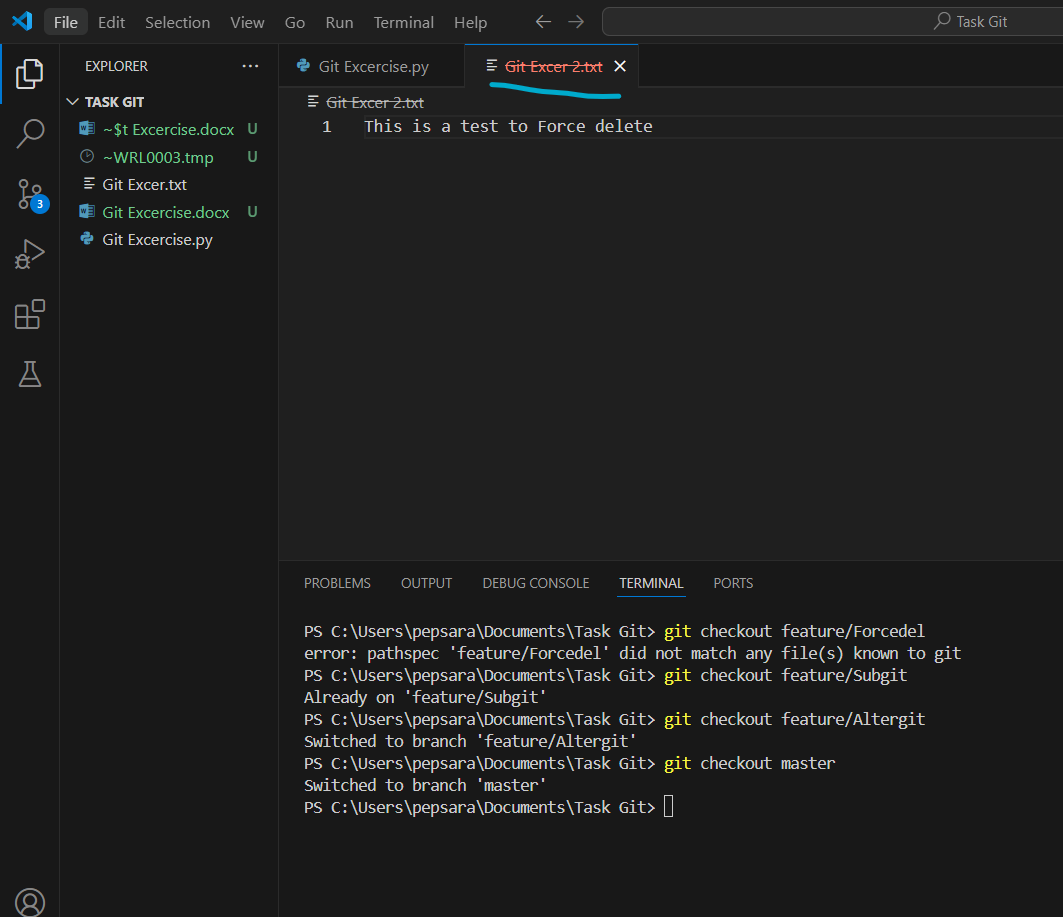
Delete the branch regardless of merge status, possibly losing changes.





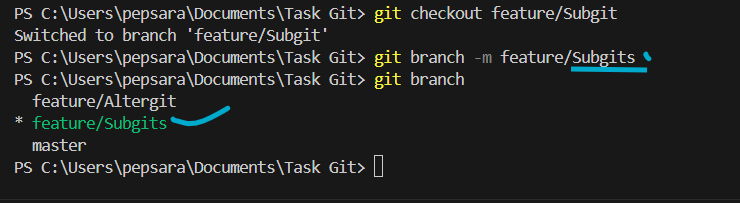


This will still be cancelled as “ The change is done in Deleted Branch” ?

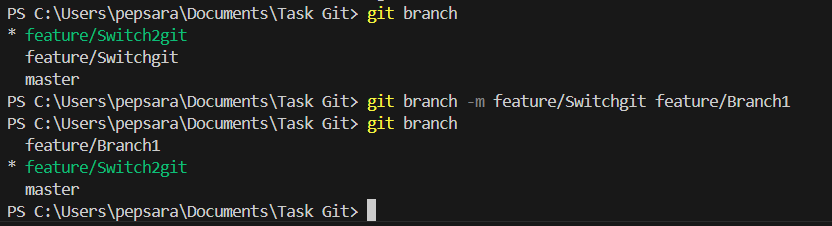


1. Rename a branch

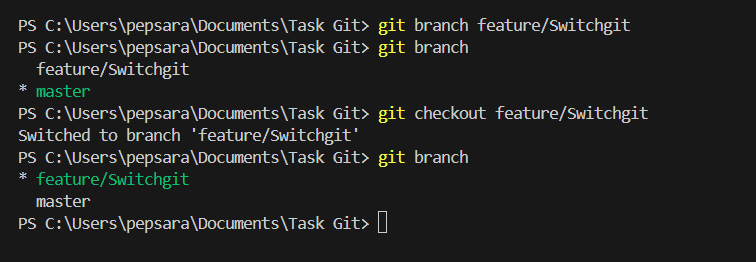
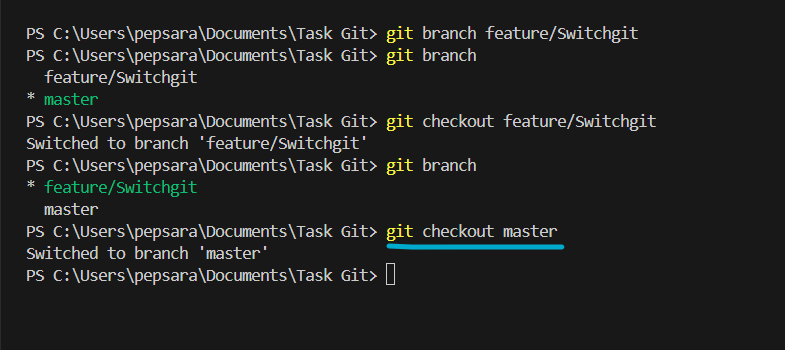
To rename a checked-out branch

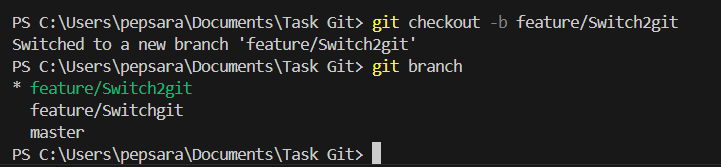


To rename a Non checked-out branch

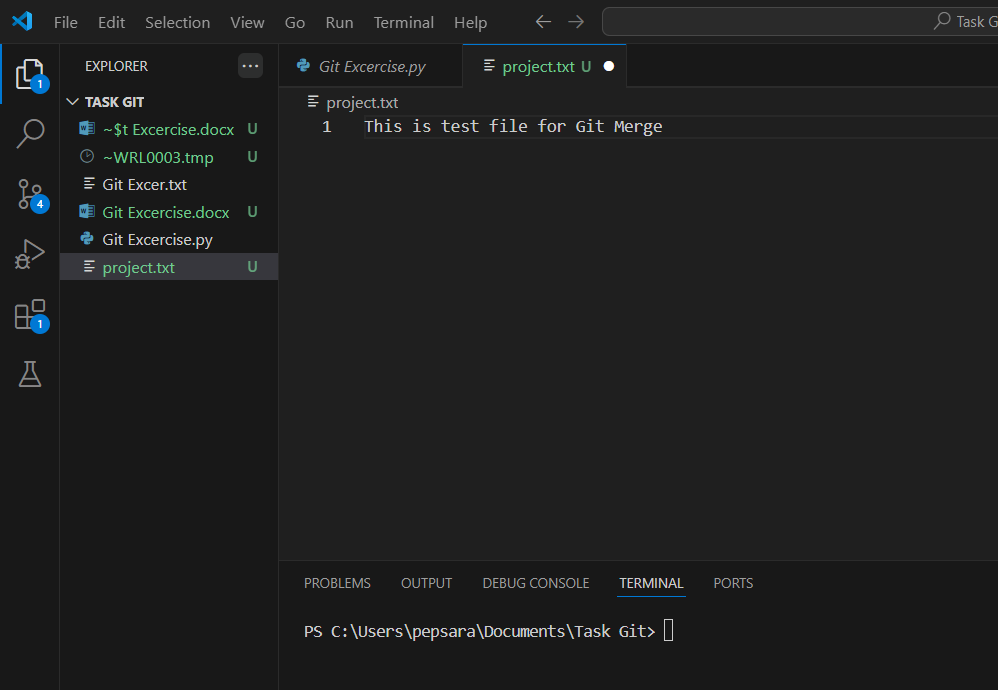


**Checking Out:**

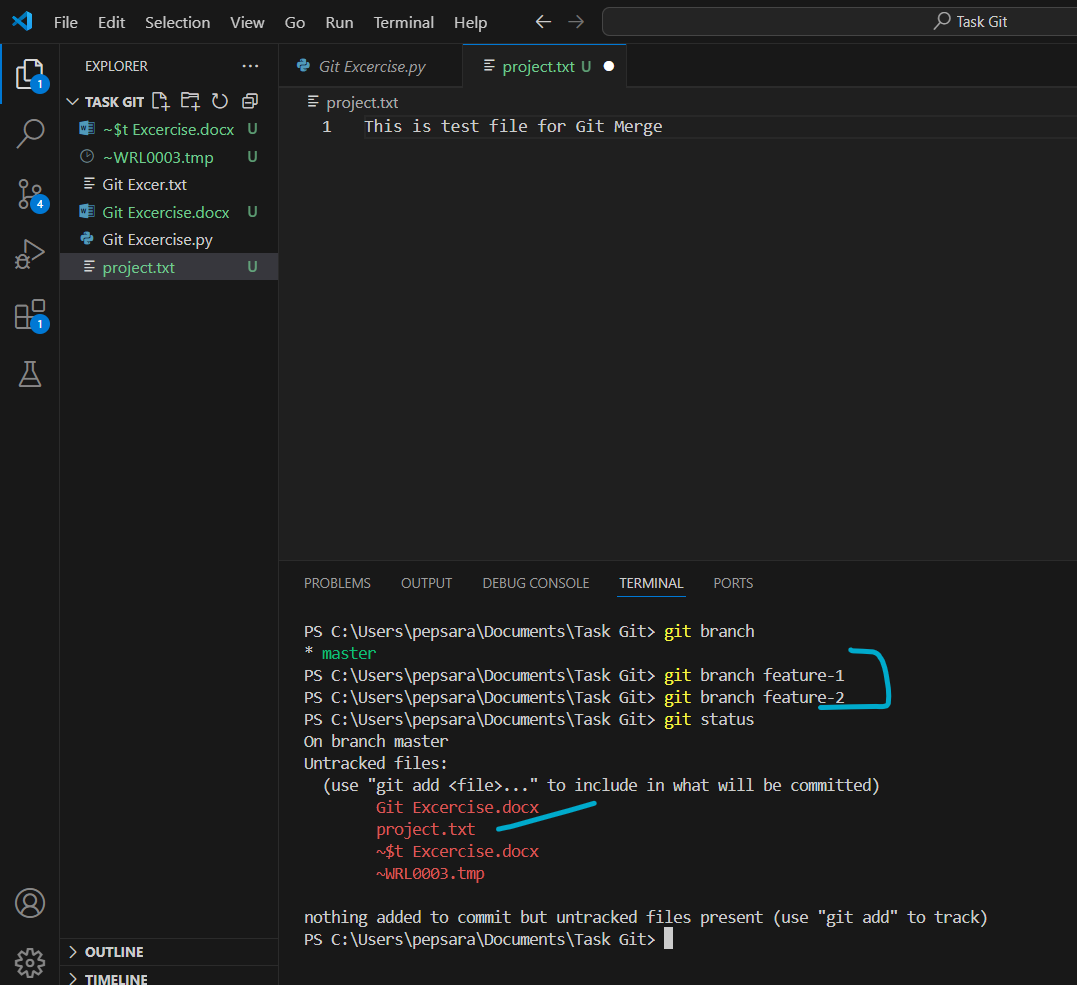
1. Switch to a branch
2. Switch to main branch using git checkout
3. Create and switch to a branch &
4. Create and switch to new branch using single command



1. Start with a repository containing a file name project.txt

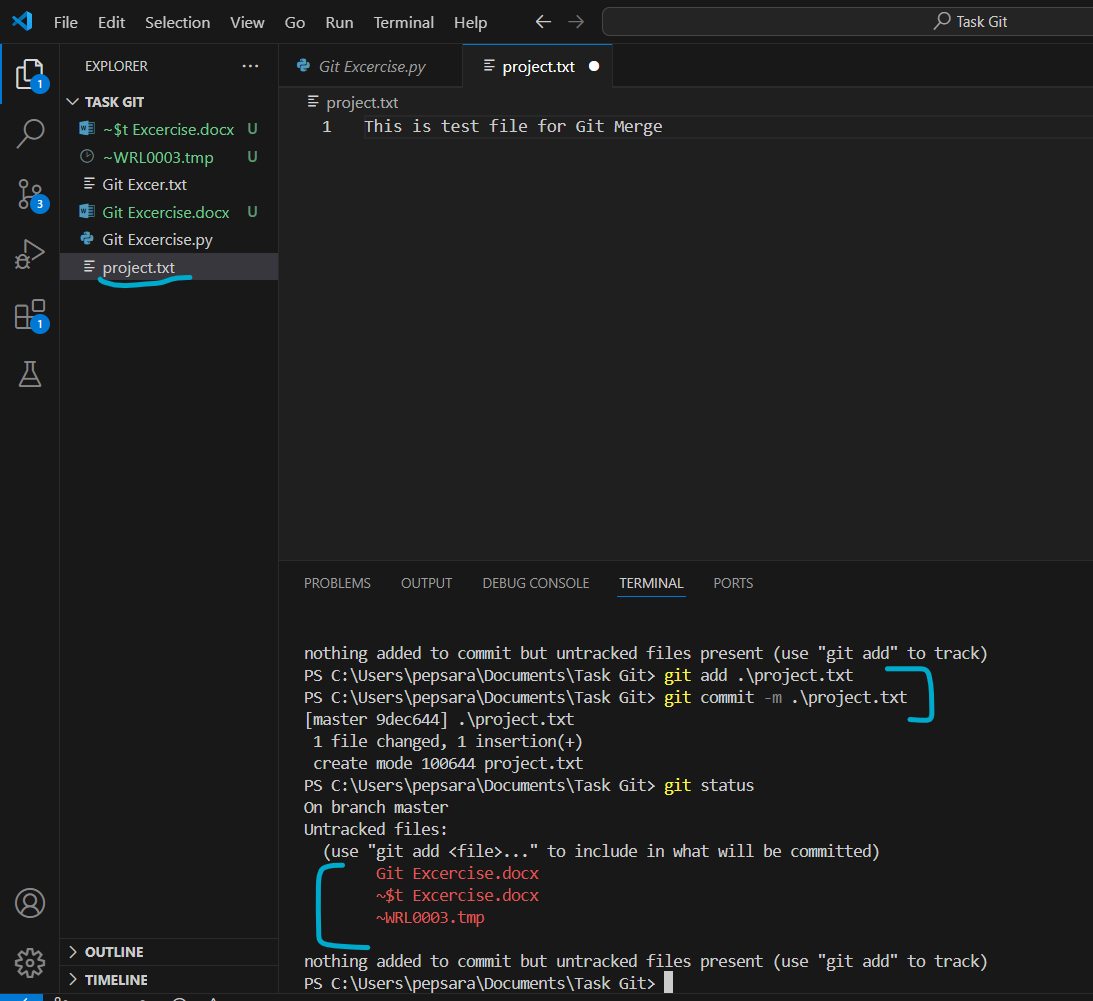


1. Create 2 branches feature 1 & feature 2 from the main branch

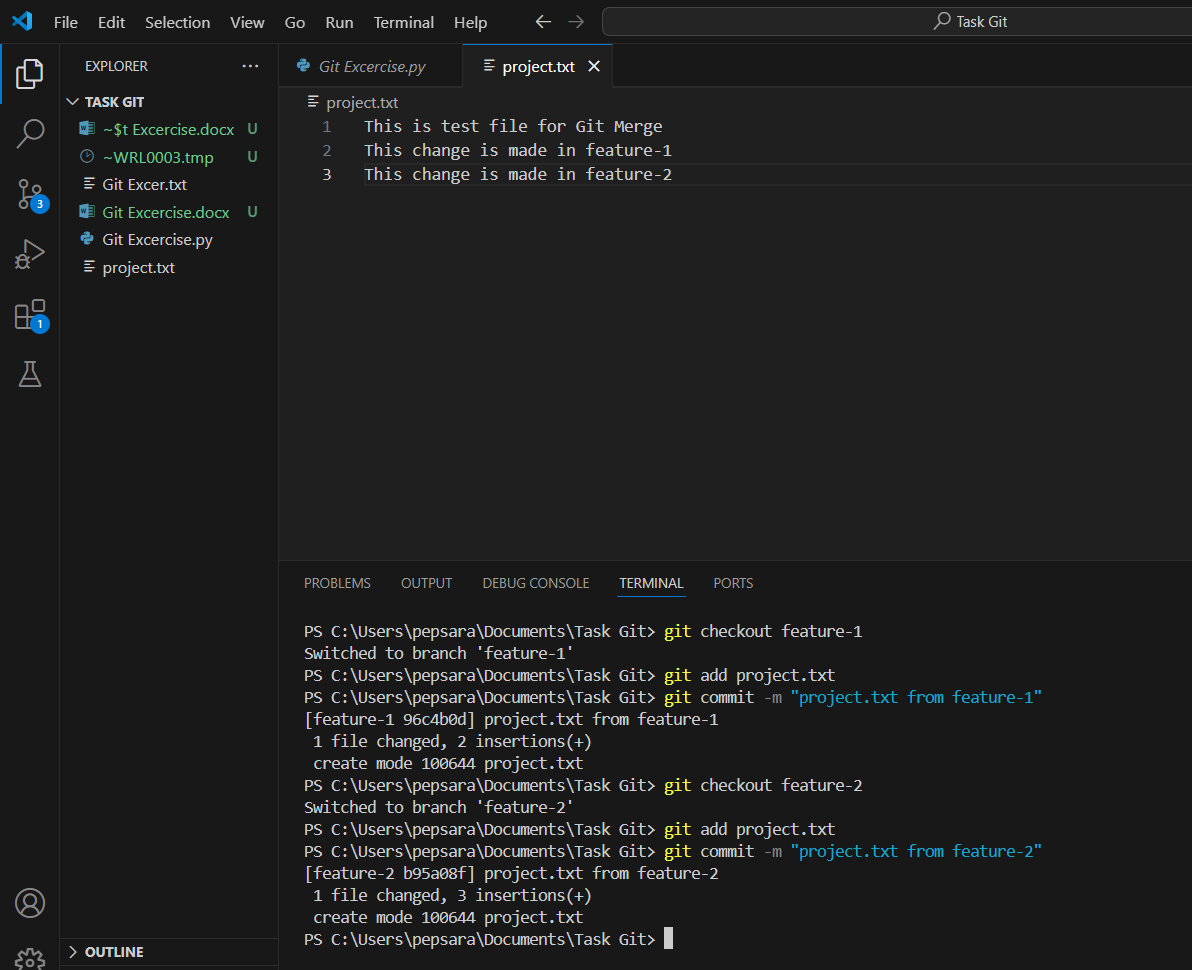


1. Make changes to project.txt in both feature branches

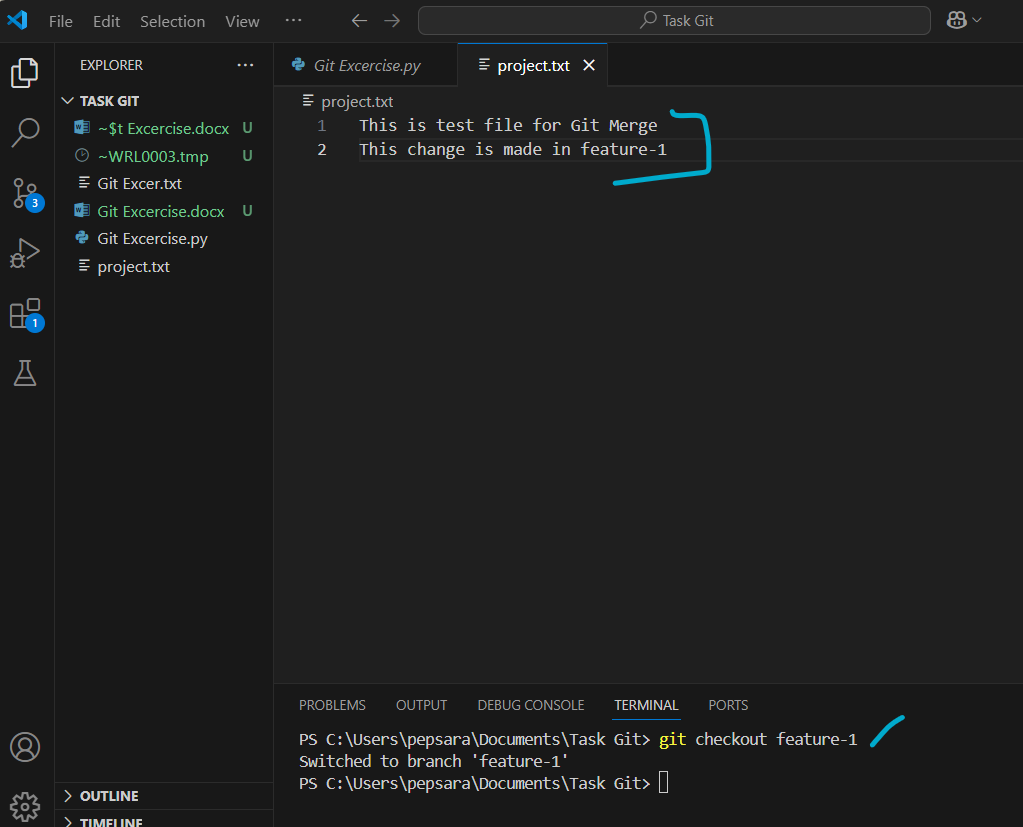
Created a project.txt with first line text and is committed in master



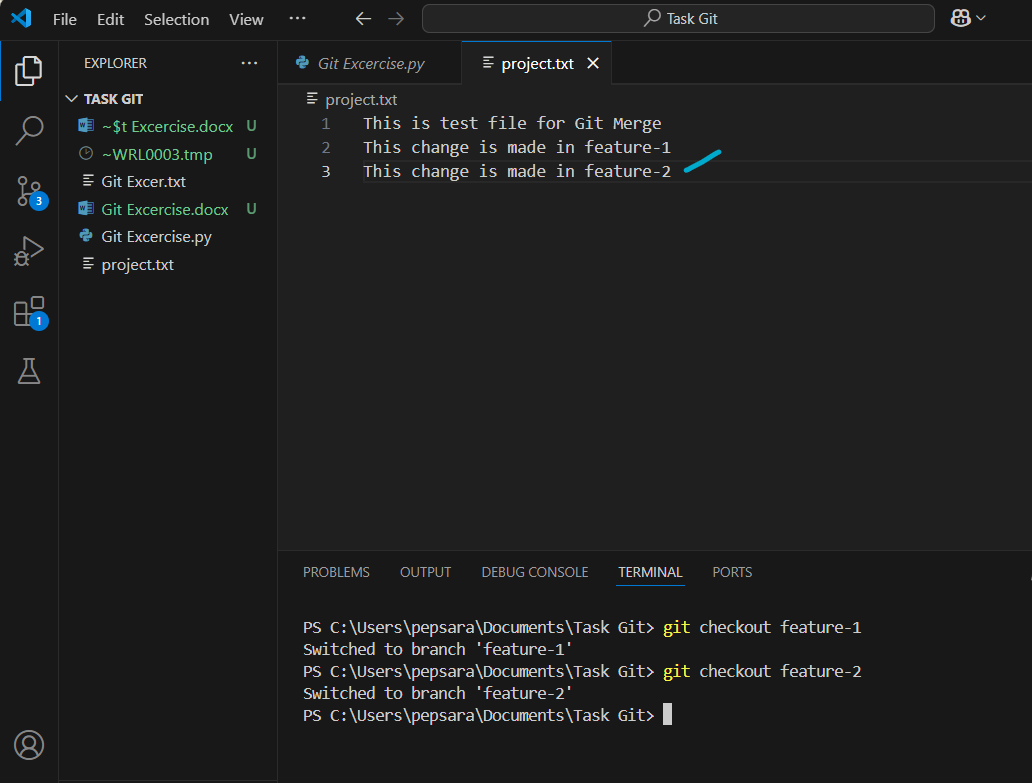
Two changes are made, each line in each feature branch & committed



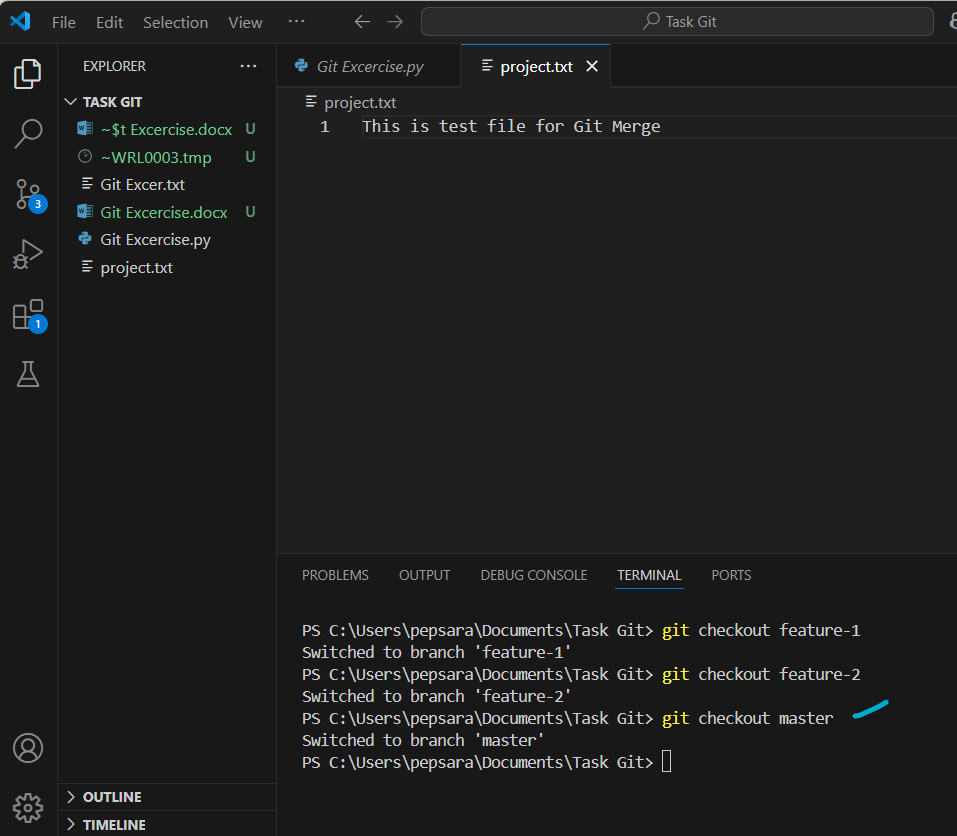
Seen from feature-1



Seen from feature-2

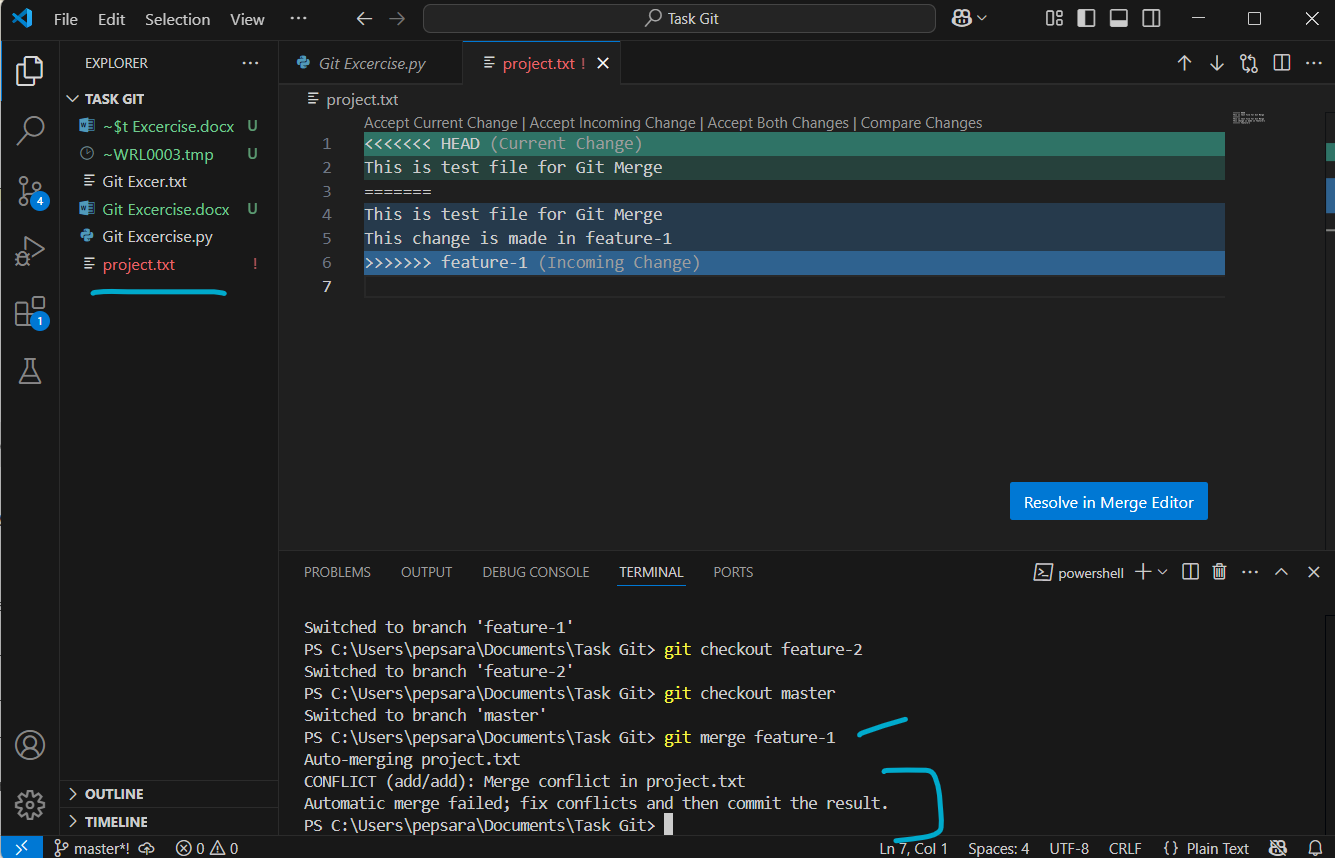


Checkout to master

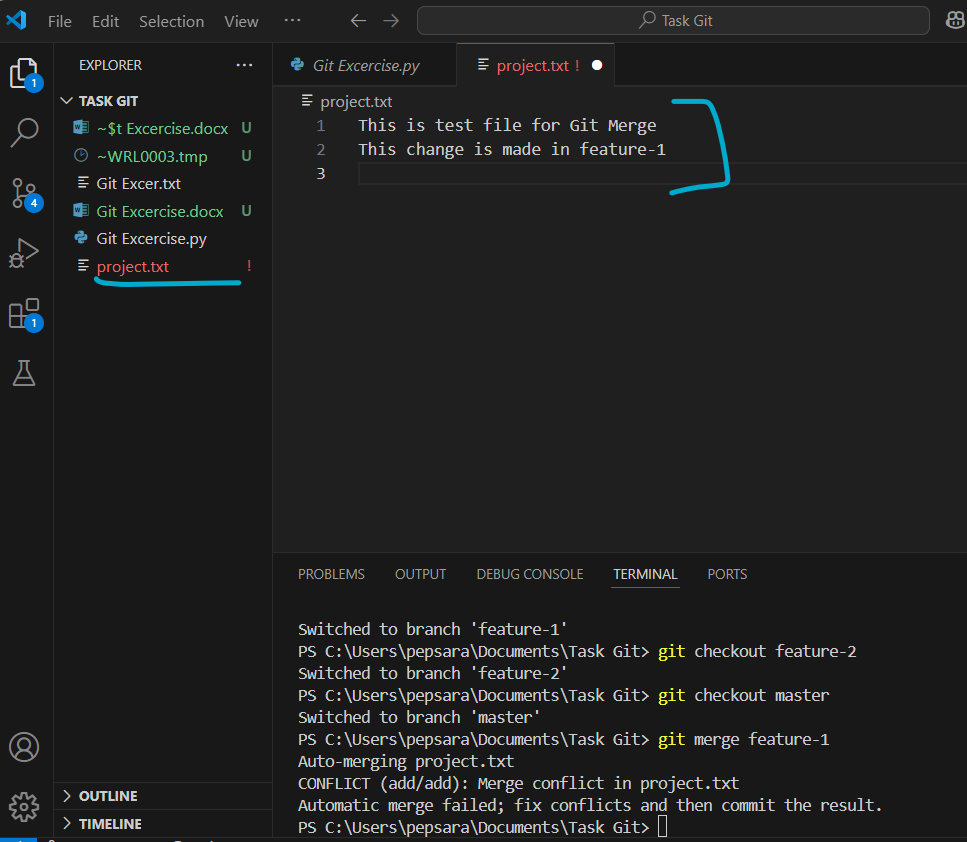


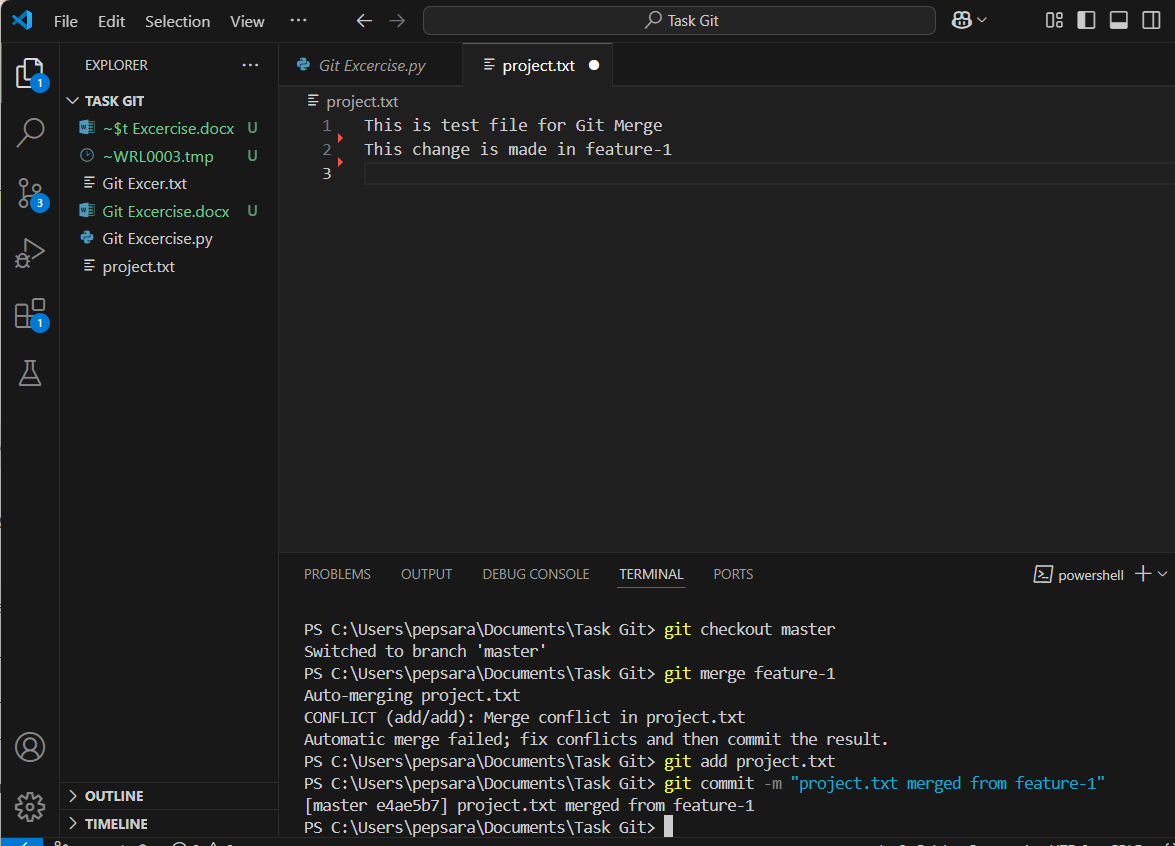
1. Resolve any merge conflicts

Merge feature-1

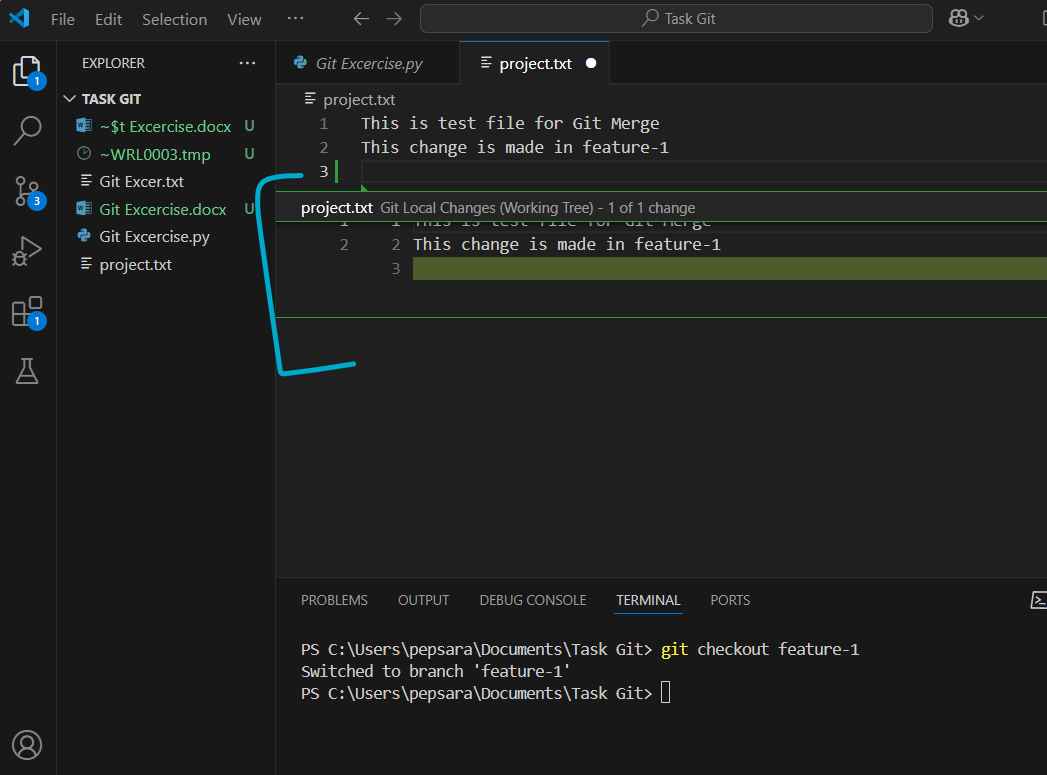


Resolved manually

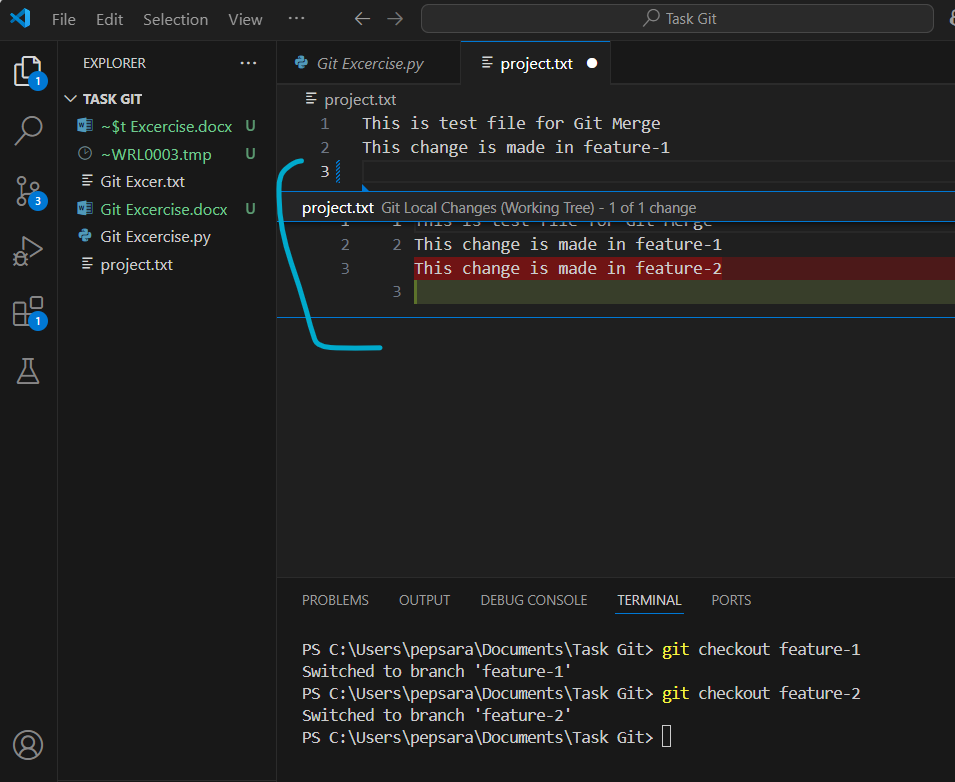




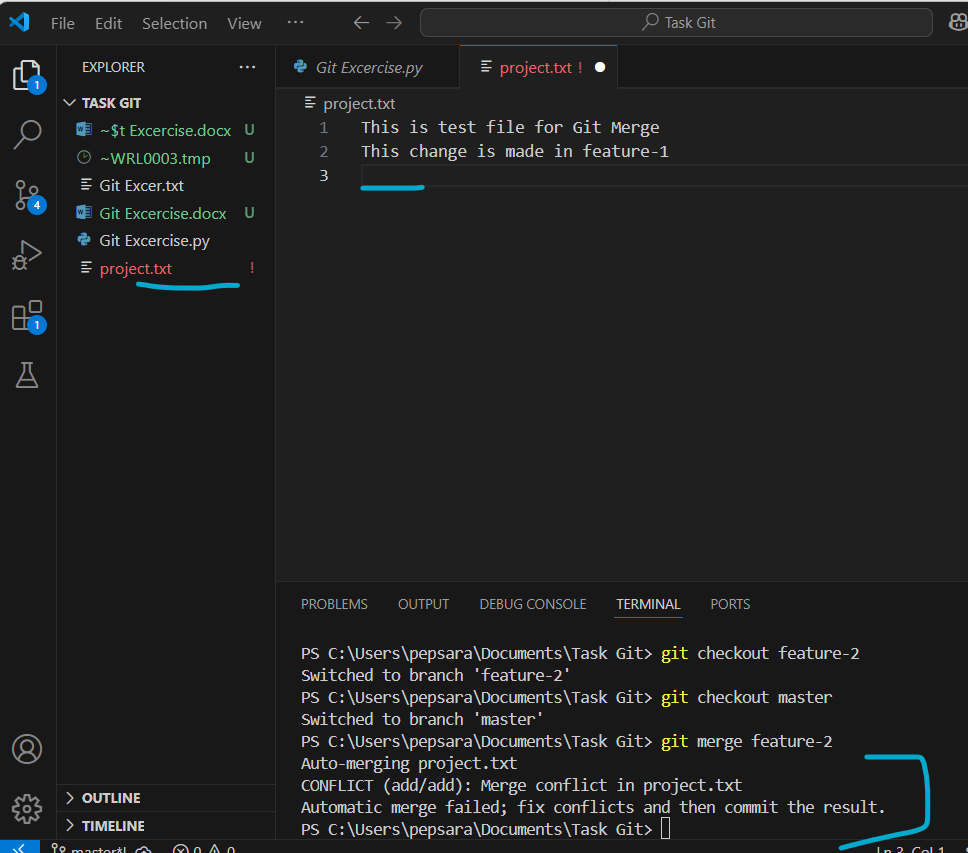
Now view from feature-1



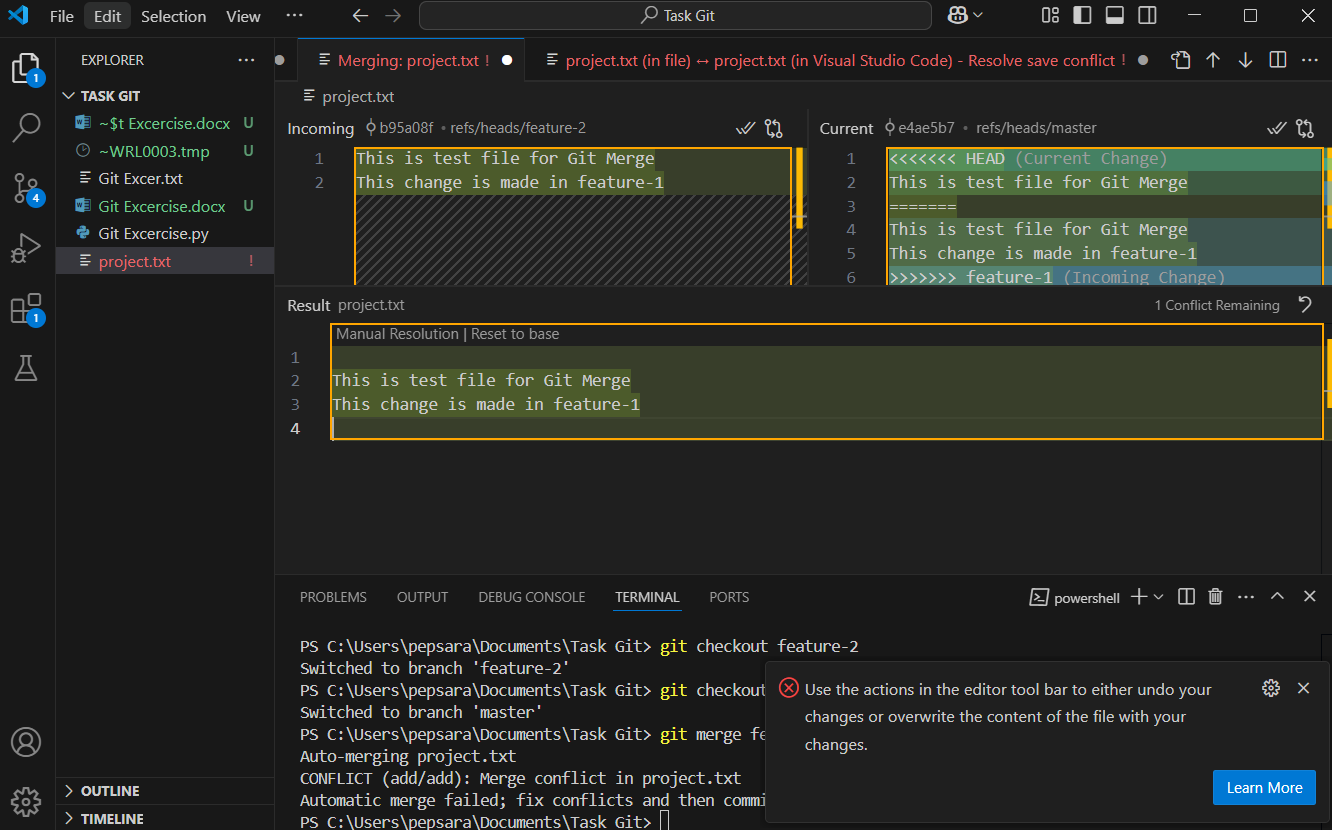
Now view from feature-2

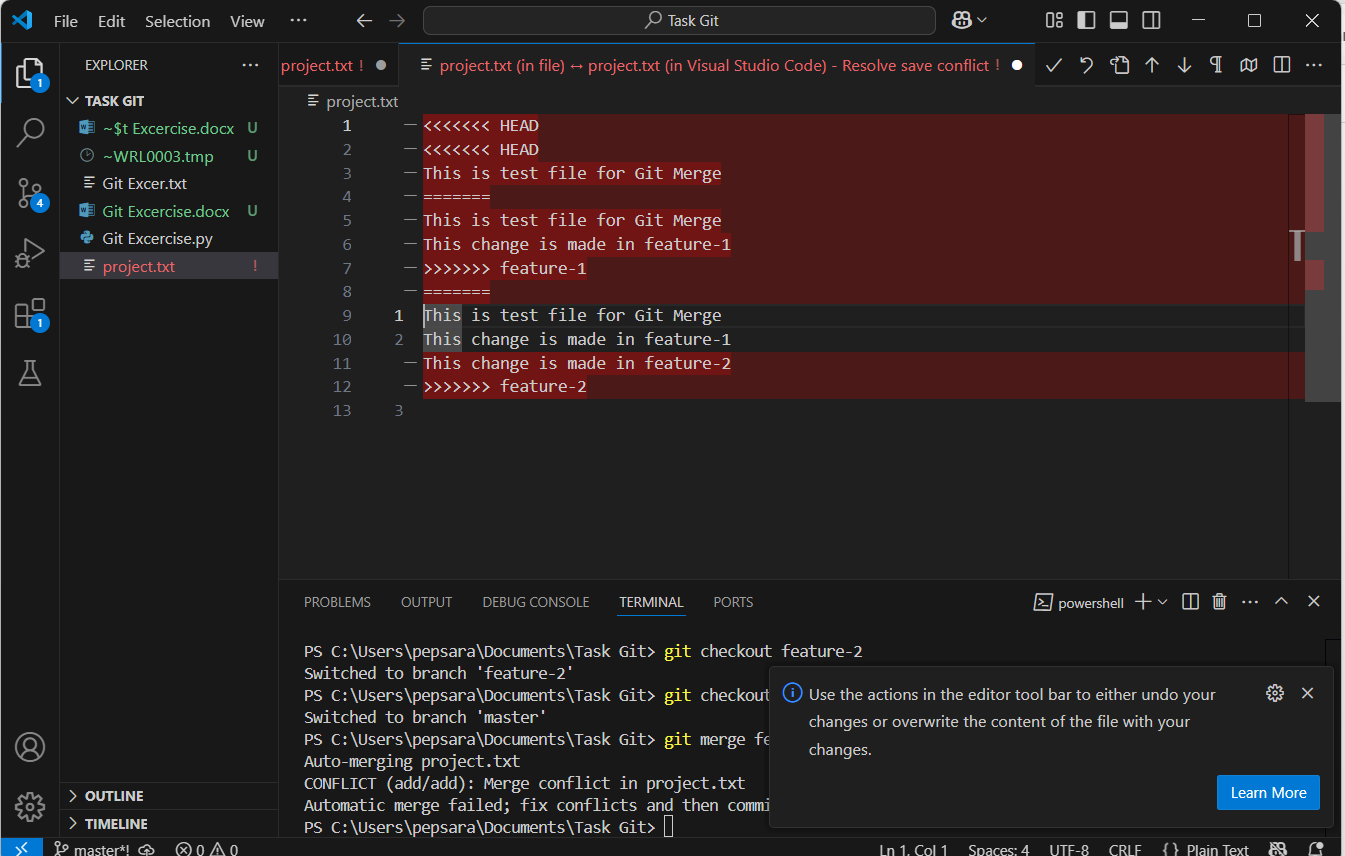


Checkout to Master & Merge the feature-2

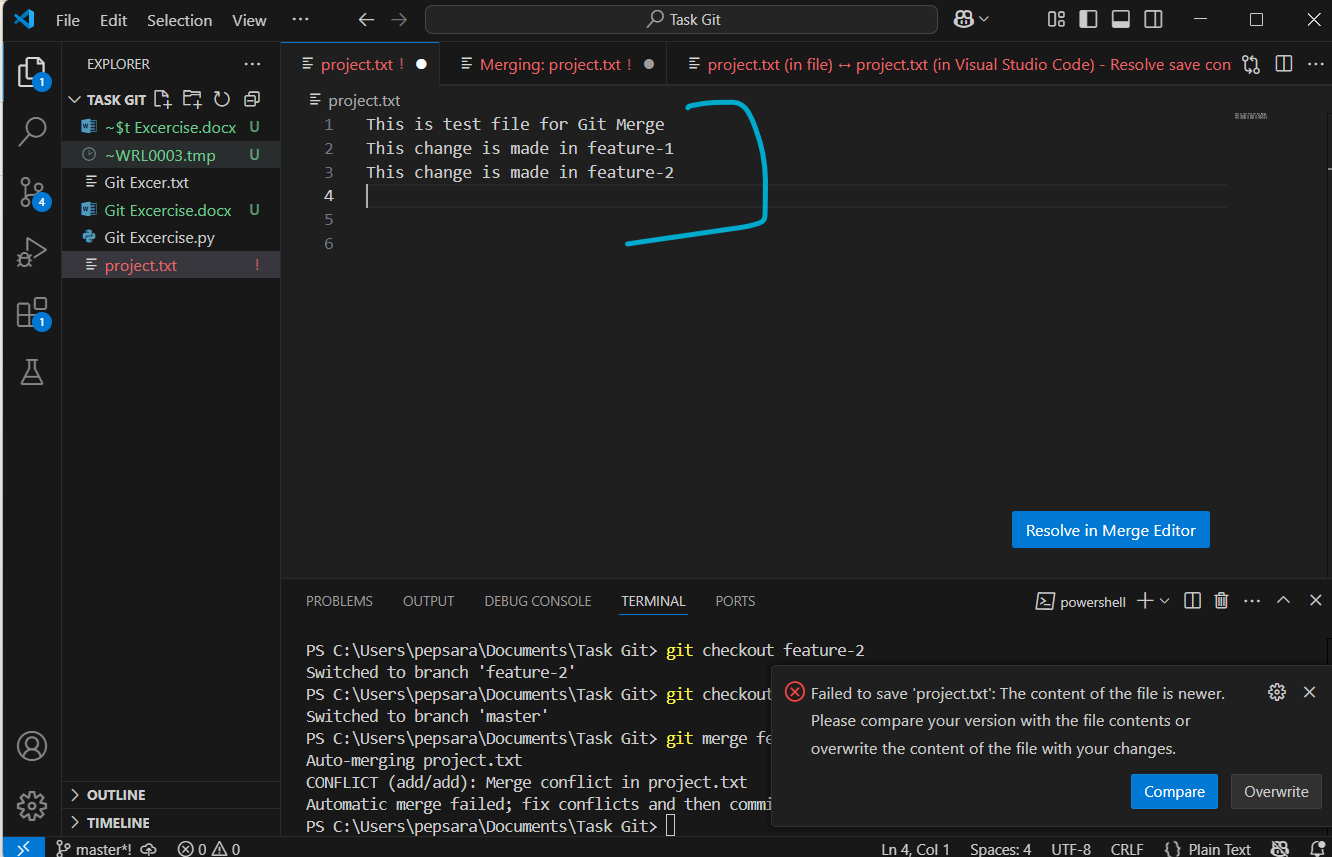


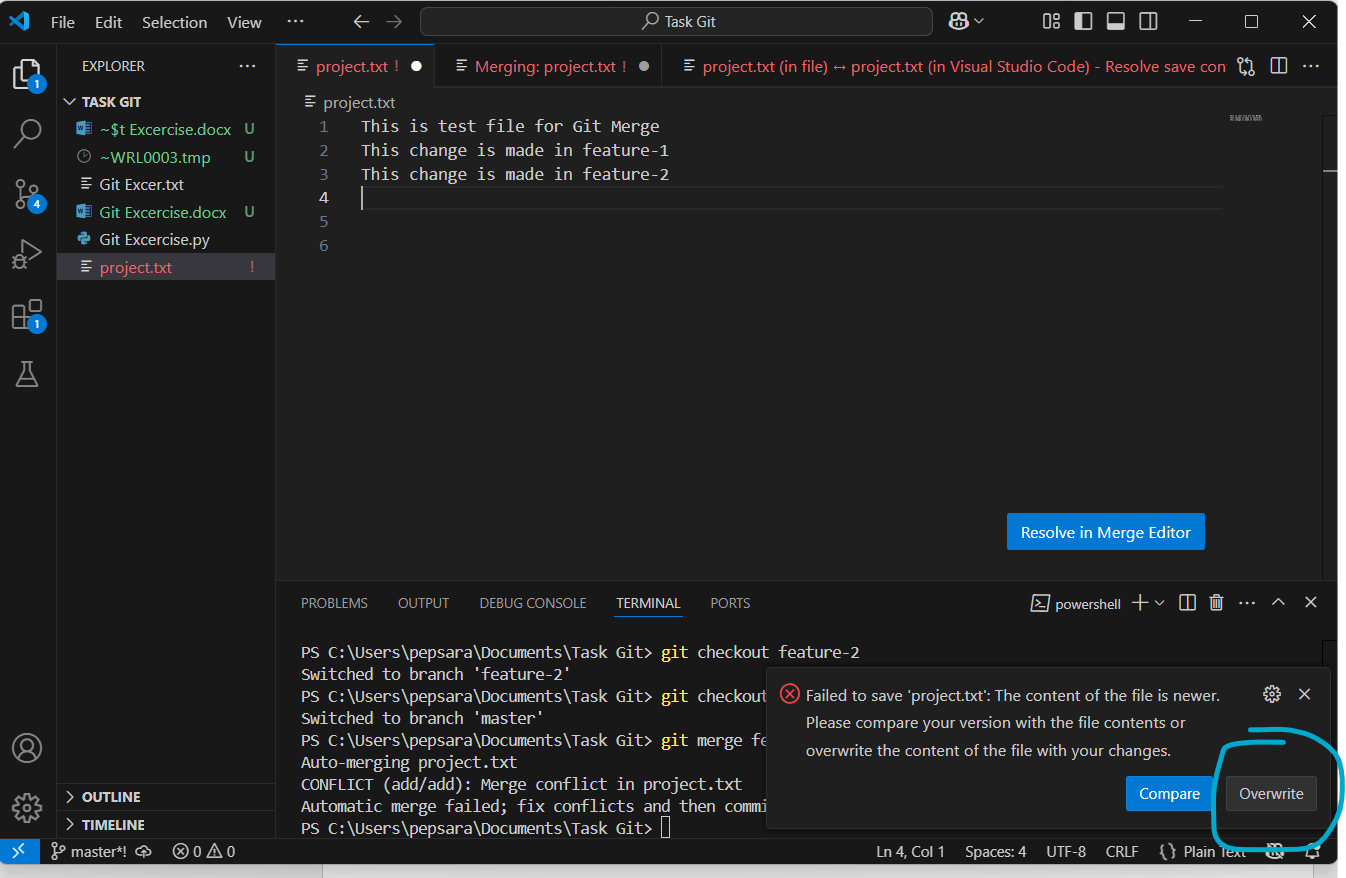
Compare button: to view & change in Merge editor

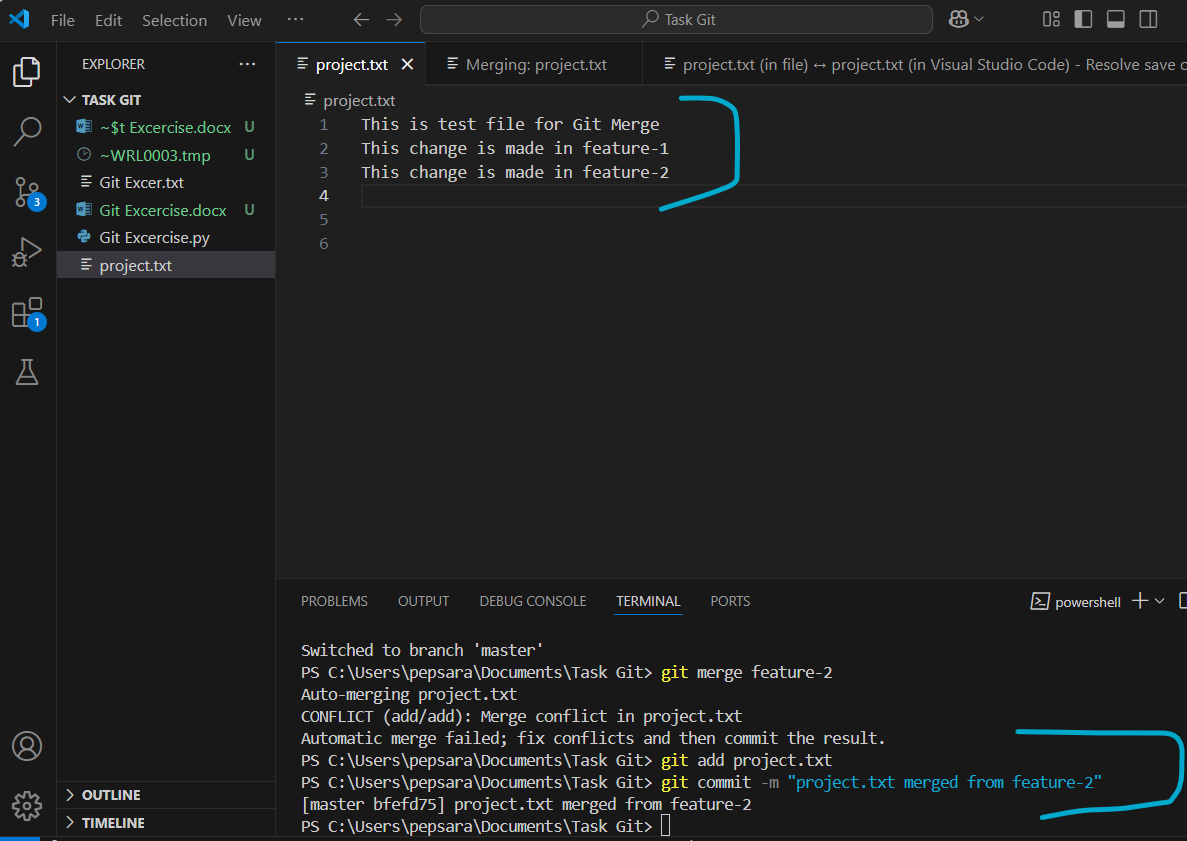


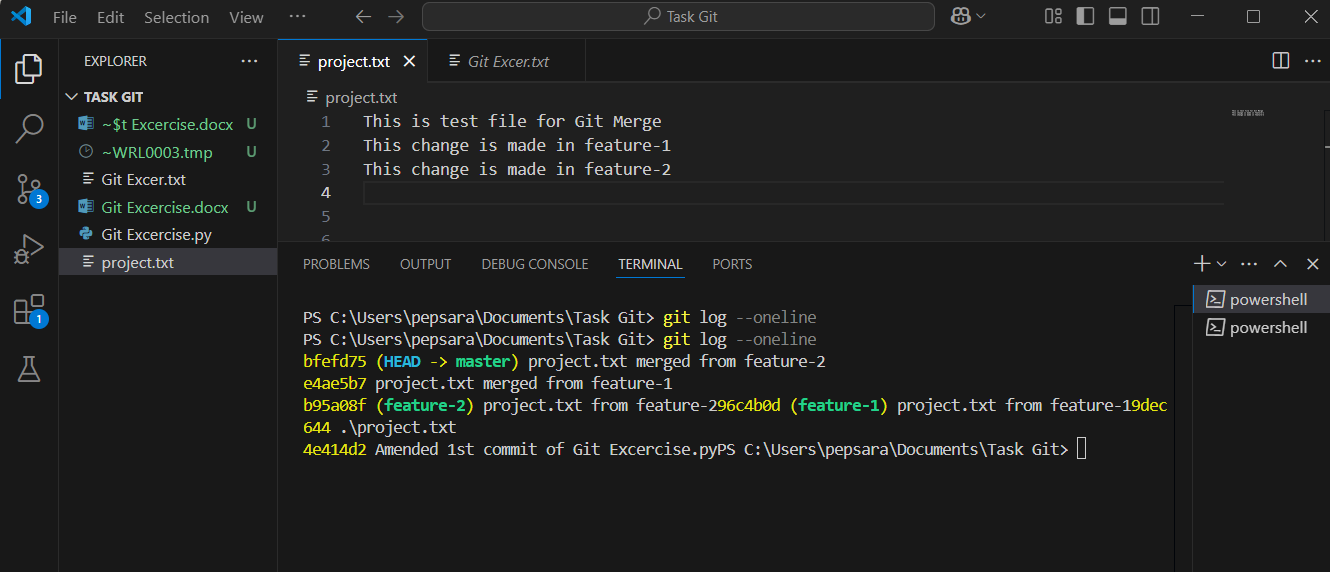
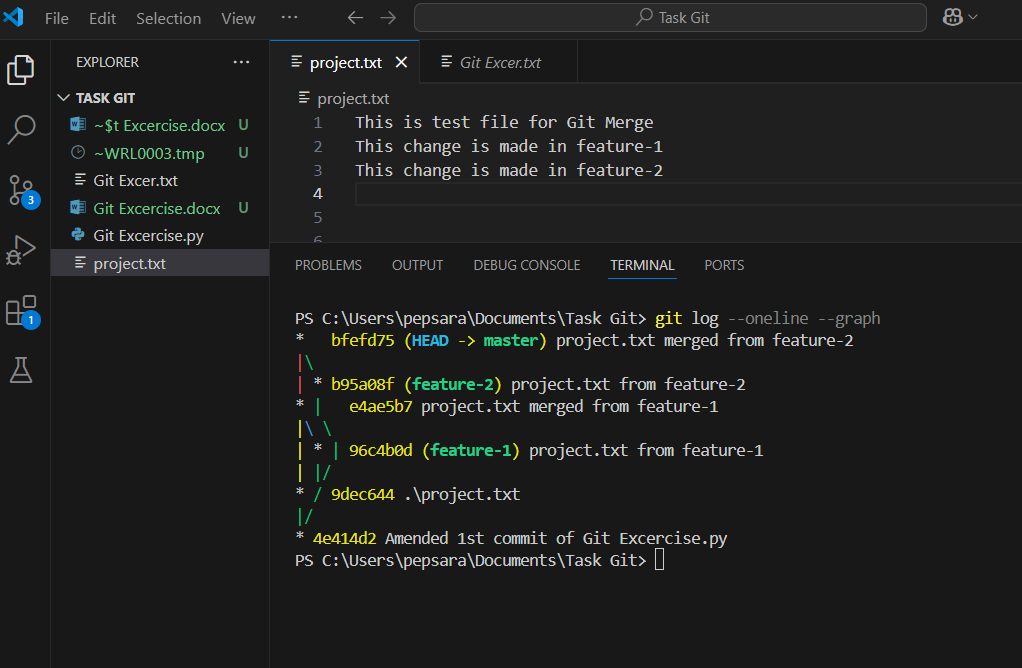


Manually copied & pasted







1. Git history in one line format
2. History in graph format
3. Filter commits by author