Hussein Chalhoub – Git Lab

Exercise 1 – Tracking Files : *Main Task :*

1. husse@LAPTOP-993BVQKP MINGW64 /c/Users/husse/OneDrive/Documents/USJ/M2/S3/Machine Learning Operations/Lab/lab1

$ mkdir Software\_Engineer\_Git\_Lab

husse@LAPTOP-993BVQKP MINGW64 /c/Users/husse/OneDrive/Documents/USJ/M2/S3/Machine Learning Oplab1

$ cd Software\_Engineer\_Git\_Lab

1. husse@LAPTOP-993BVQKP MINGW64 /c/Users/husse/OneDrive/Documents/USJ/M2/S3/Machine Learning Oplab1/Software\_Engineer\_Git\_Lab

$ git init

Initialized empty Git repository in C:/Users/husse/OneDrive/Documents/USJ/M2/S3/Machine Learning Operations/Lab/lab1/Software\_Engineer\_Git\_Lab/.git/

1. *Git init creates a new git repository by adding a hidden .git folder in your project. This is usually the first command we run, and it stores all meta data and history for version control.*
2. husse@LAPTOP-993BVQKP MINGW64 /c/Users/husse/OneDrive/Documents/USJ/M2/S3/Machine Learning Operations/Lab/lab1/Software\_Engineer\_Git\_Lab (master)

$ echo "# Git Lab Project " > README

1. husse@LAPTOP-993BVQKP MINGW64 /c/Users/husse/OneDrive/Documents/USJ/M2/S3/Machine Learning Operations/Lab/lab1/Software\_Engineer\_Git\_Lab (master)

$ git status

On branch master

No commits yet

Untracked files:

(use "git add <file>..." to include in what will be committed)

README

nothing added to commit but untracked files present (use "git add" to track)

1. husse@LAPTOP-993BVQKP MINGW64 /c/Users/husse/OneDrive/Documents/USJ/M2/S3/Machine Learning Operations/Lab/lab1/Software\_Engineer\_Git\_Lab (master)

$ git add README

husse@LAPTOP-993BVQKP MINGW64 /c/Users/husse/OneDrive/Documents/USJ/M2/S3/Machine Learning Operations/Lab/lab1/Software\_Engineer\_Git\_Lab (master)

$ git status

On branch master

No commits yet

Changes to be committed:

(use "git rm --cached <file>..." to unstage)

new file: README

1. husse@LAPTOP-993BVQKP MINGW64 /c/Users/husse/OneDrive/Documents/USJ/M2/S3/Machine Learning Operations/Lab/lab1/Software\_Engineer\_Git\_Lab (master)

$ git commit -m "add readme file"

[master (root-commit) f7d02db] add readme file

1 file changed, 1 insertion(+)

create mode 100644 README

1. file1.py and file2.py :

A screenshot of a computer screen

AI-generated content may be incorrect.

1. both files staged:

A screen shot of a computer code

AI-generated content may be incorrect.

1. diffA screen shot of a computer

   AI-generated content may be incorrect.
2. the gif diff looks for changes in un staged area, then we sed – staged to fix it 1A computer screen with text on it

   AI-generated content may be incorrect.
3. here eh result shows the difference , then we added the unstaged and the staged of the same file and combined them , thenended up committing to add them both.A screenshot of a computer program

   AI-generated content may be incorrect.A black screen with yellow and blue text

   AI-generated content may be incorrect.
4. used log to see the commitsA computer screen with text

   AI-generated content may be incorrect.

A computer screen shot of a program

AI-generated content may be incorrect.

1. we can write the first 7 characters of the commit hash instead f the full one:

A screen shot of a computer

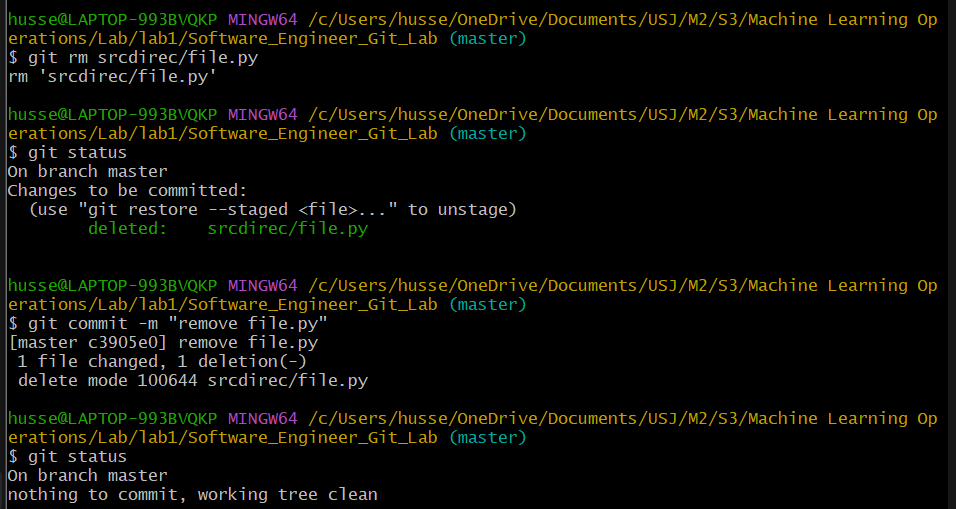
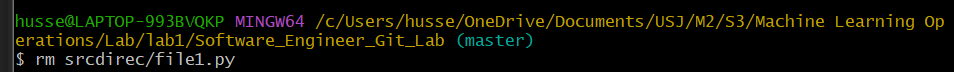
AI-generated content may be incorrect.

1. creating new file named : file.py A computer screen with text on it

   AI-generated content may be incorrect.A black screen with text

   AI-generated content may be incorrect.

Stretch Task:

1. removing file.py 
2. removed file1.py:
3. the file is unstaged so we need to run git to make it staged   
   so git helps in making it staged.

adding again file1.py

A screen shot of a computer

AI-generated content may be incorrect.

A computer screen with text

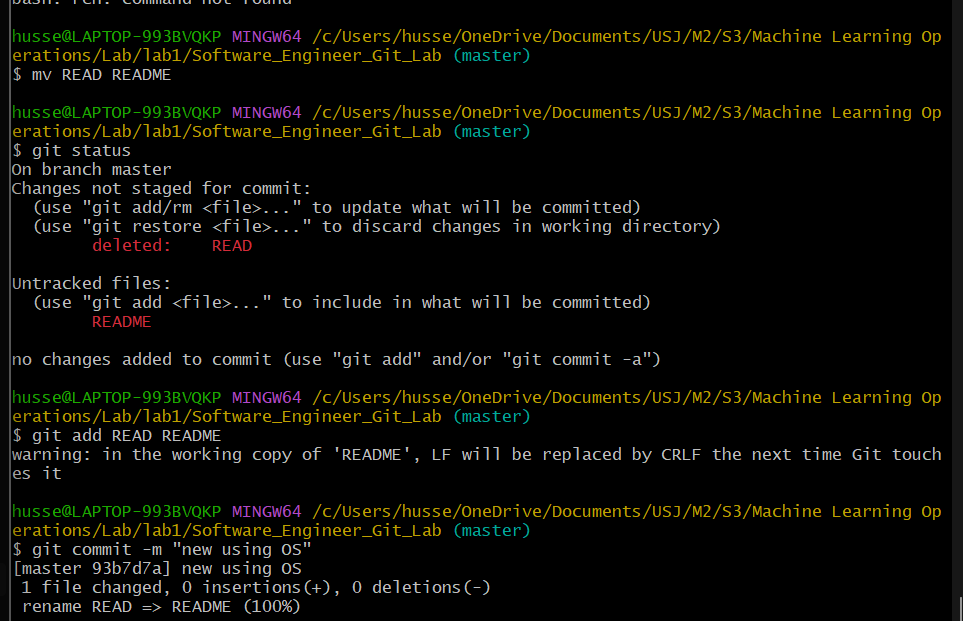
AI-generated content may be incorrect.

1. renaming readme file

A computer screen with text on it

AI-generated content may be incorrect.

1. using the os mv command git sees the file as deleted and the new one as untracked  
   we need to use git add to stage the changes and let git know the relation  
   after commit git records it as a rename even if the os treated it as delete and create



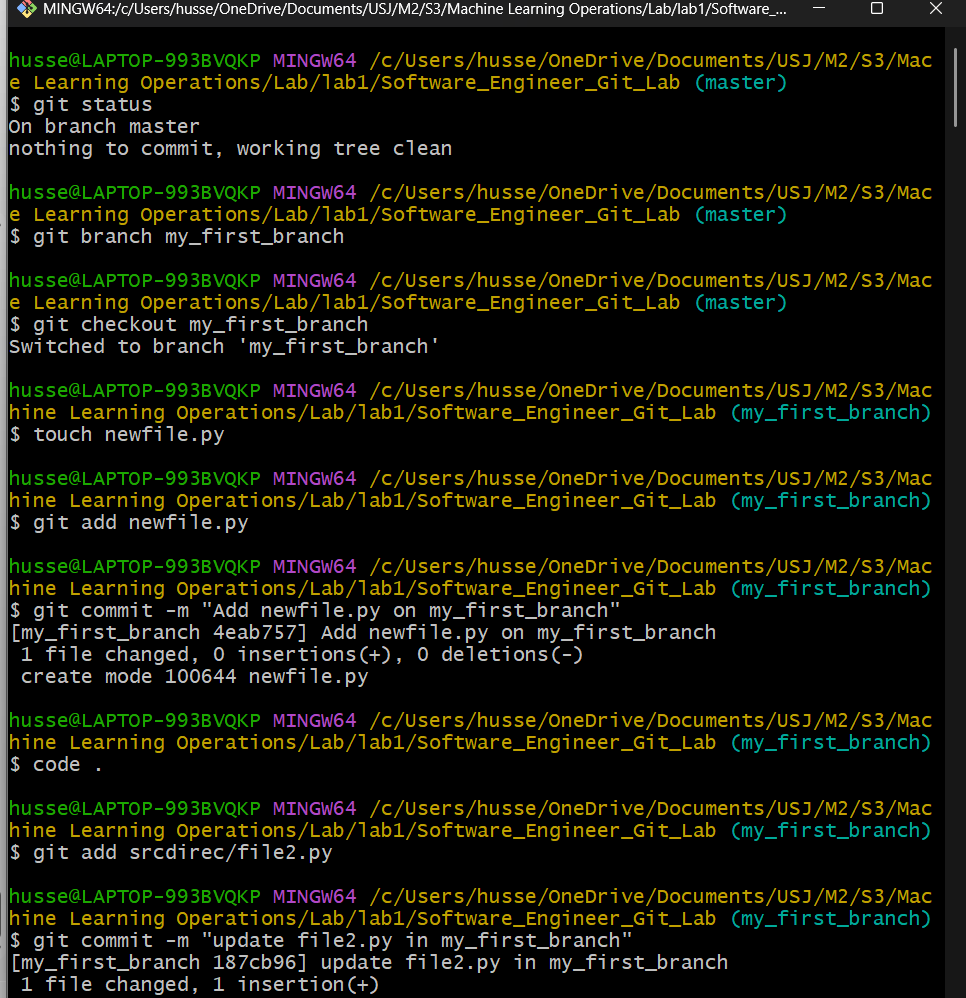
1. display the most recent 3 commit :  
   A computer screen shot of a program code

   AI-generated content may be incorrect.
2. git –stat shows which files are changed, staged, or untracked in your repository, it helps to see what will be committed next. A screen shot of a computer program

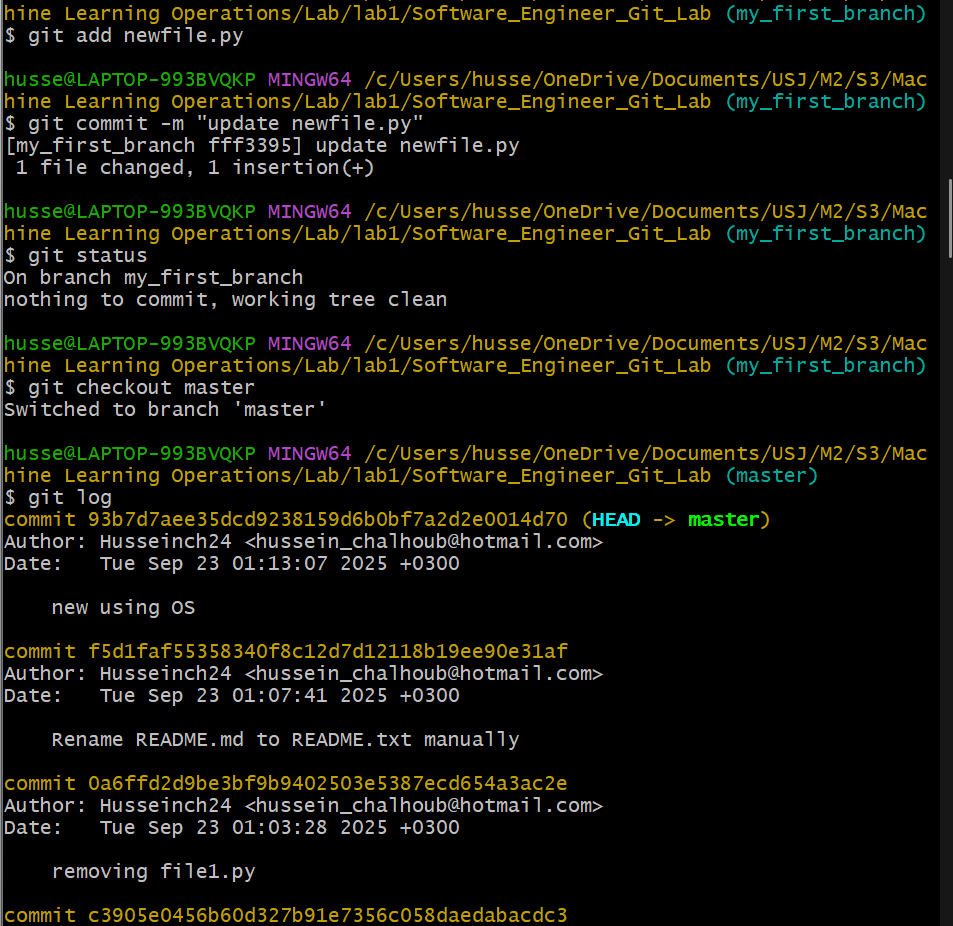
   AI-generated content may be incorrect.A computer screen with text and numbers

   AI-generated content may be incorrect.

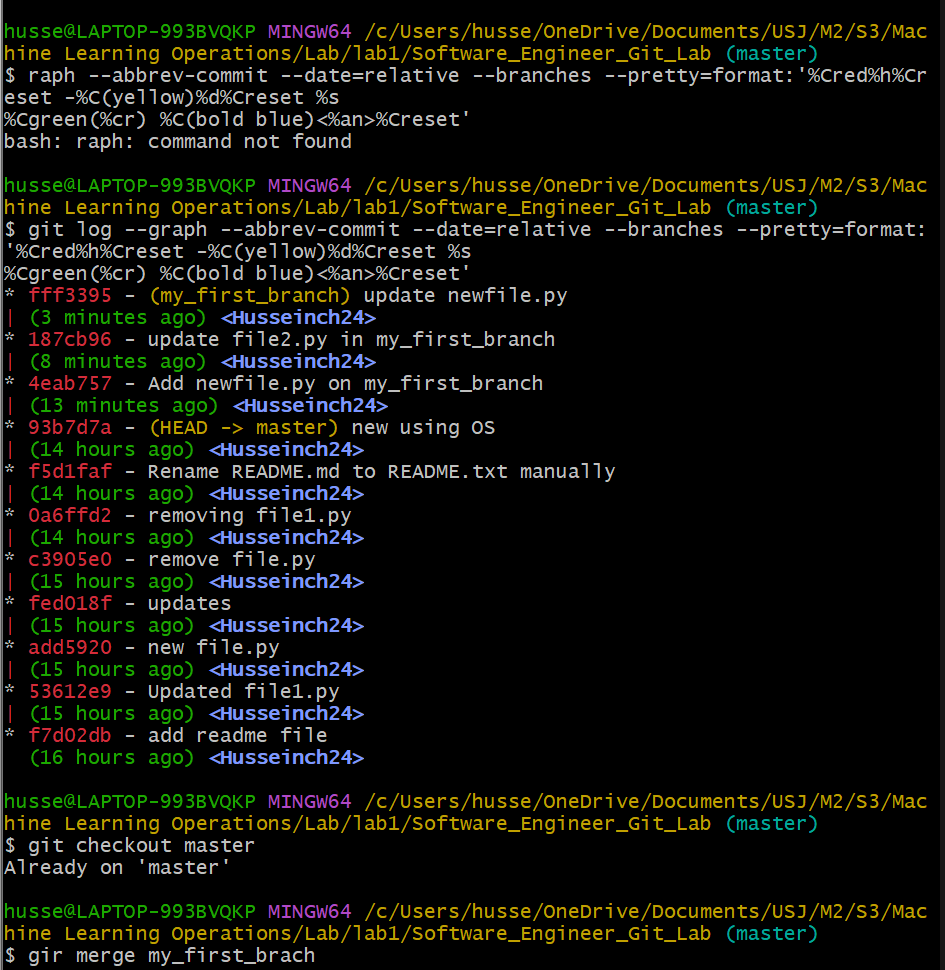
Exercise 2 - Main Task - Git Branches:

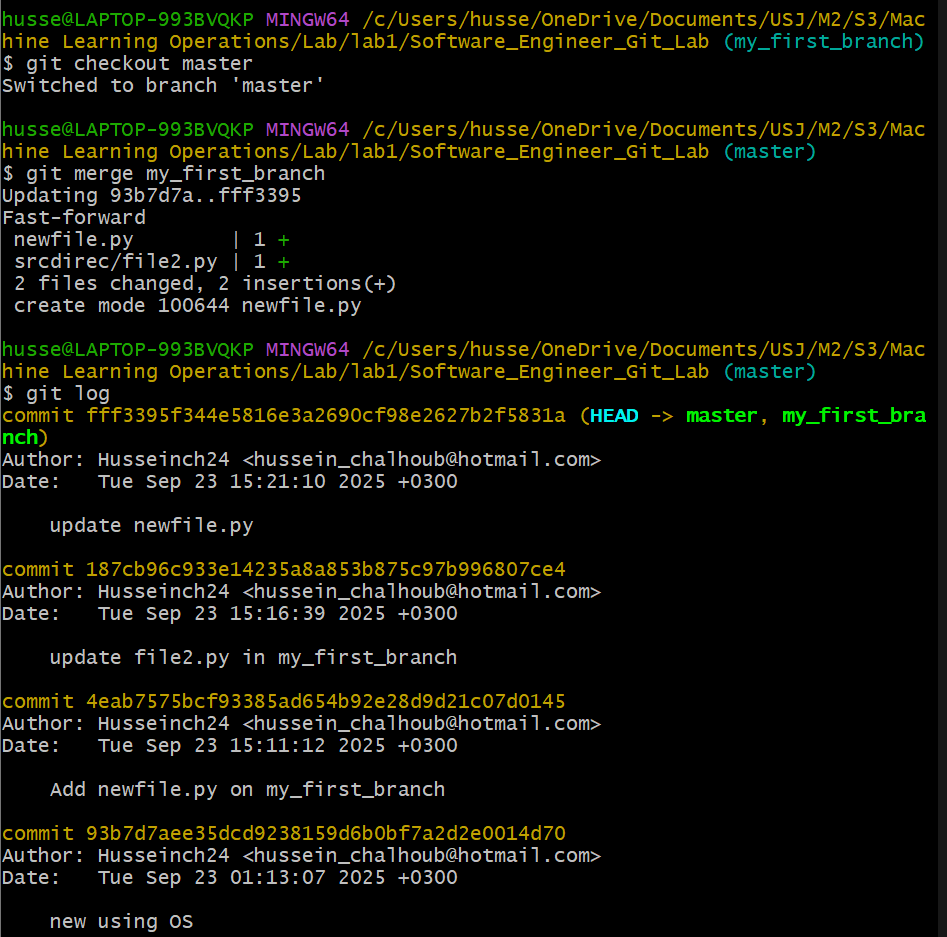
1. 2. 3. 4. A screenshot of a computer program

   AI-generated content may be incorrect.

5. 

6. 7.



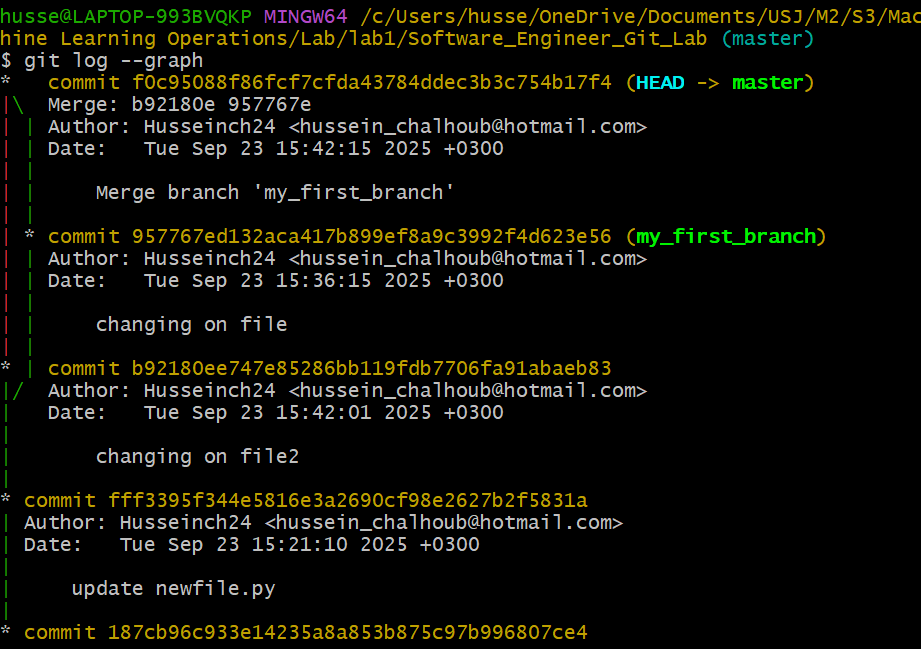
8.A computer screen shot of a program

AI-generated content may be incorrect.

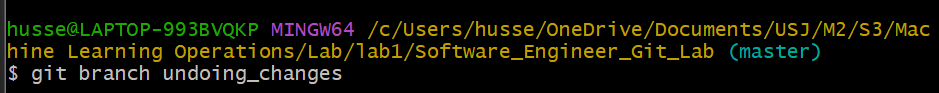
1. 10. THE OUTPUT WAS HIDDEN I DON’T KNOW WHY AFTER I MADE THE MERGE OF PART 11

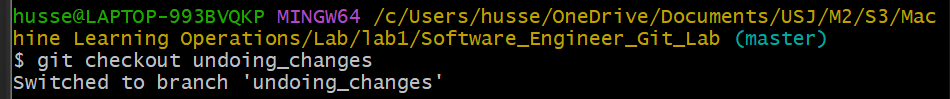
11 - A black background with white text

AI-generated content may be incorrect.

12 - 

Exercise 3 - Main Task: Undoing Changes in Git :

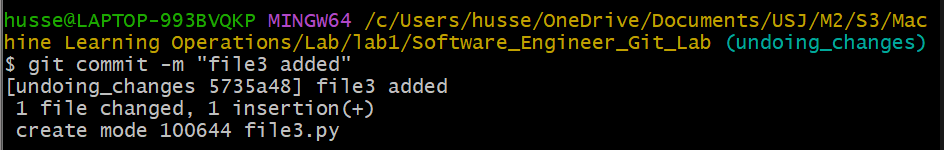
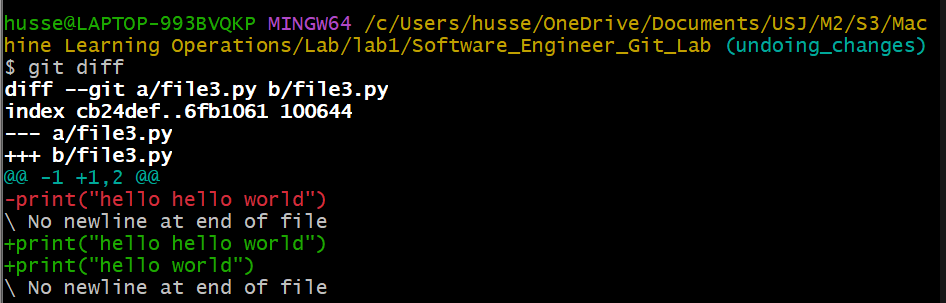
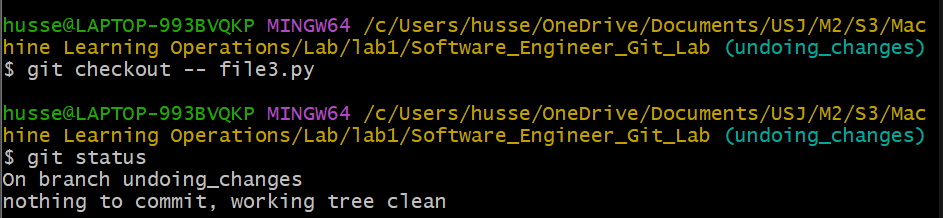
1. Undoing\_changes
2. Checkout:



1. File.py created

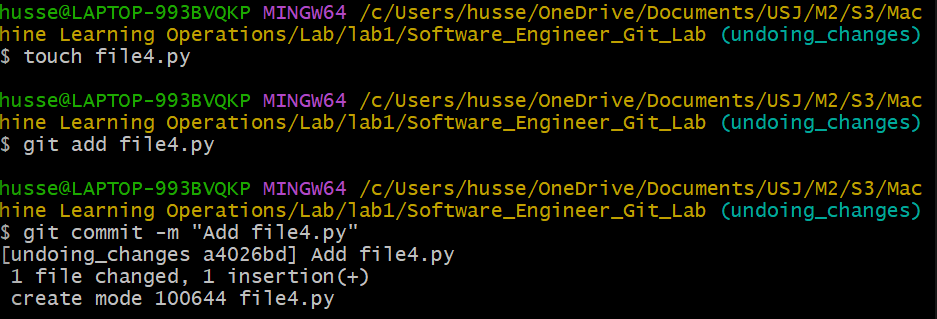
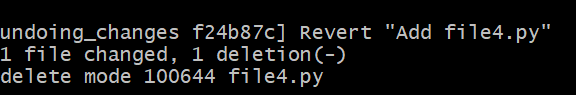
A black screen with yellow text

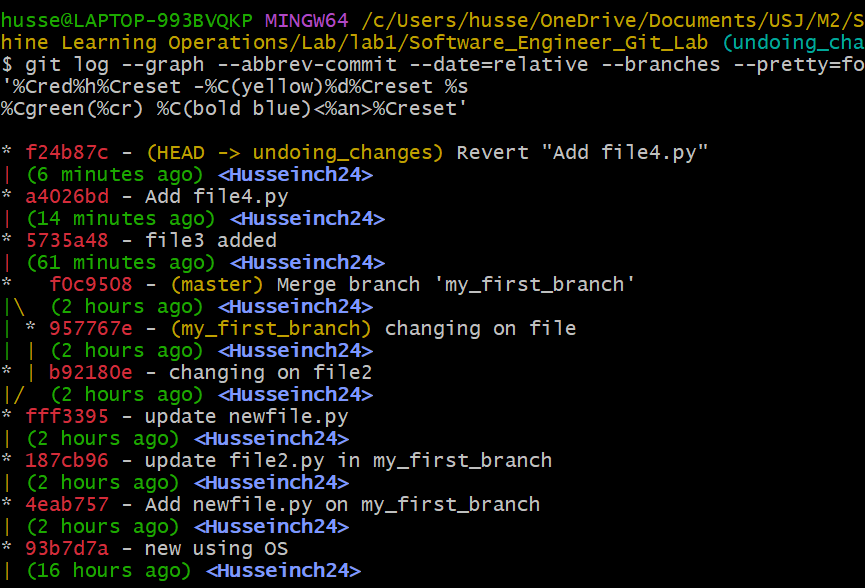
AI-generated content may be incorrect.

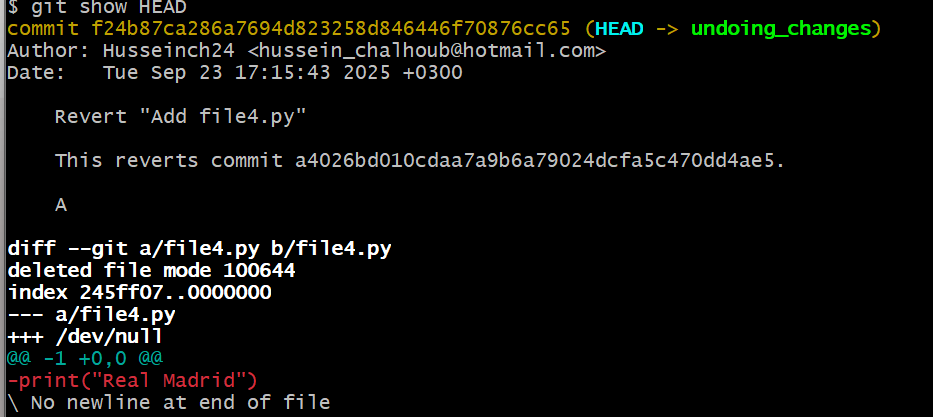
1. Committed the file 
2. Used diff 
3. A. Status :  
   

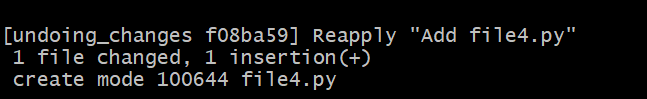
B. the changes discarded and file3 is restored to the committed state it was before.

C. yes using git restore file3.py

1. Created file4.py
2. A- the revert creates a new commit and undo the previous one thus the checkout let the files in the directory without changing the commit.

B- log : 

C- new commit is reverted and on the top with head: 

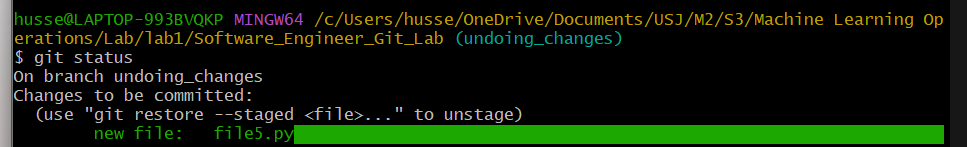
1. Committed file3 and file4

A computer screen with text and images

AI-generated content may be incorrect.

1. a. Execute the required commands step by step.  
   b. Changes are staged and ready to commit.  
   c. Previous commit is removed from history.  
   d. Use backup branch or revert for safety.

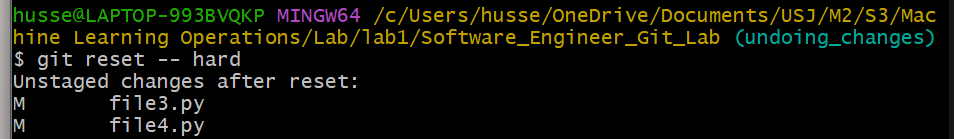
A computer screen shot of text

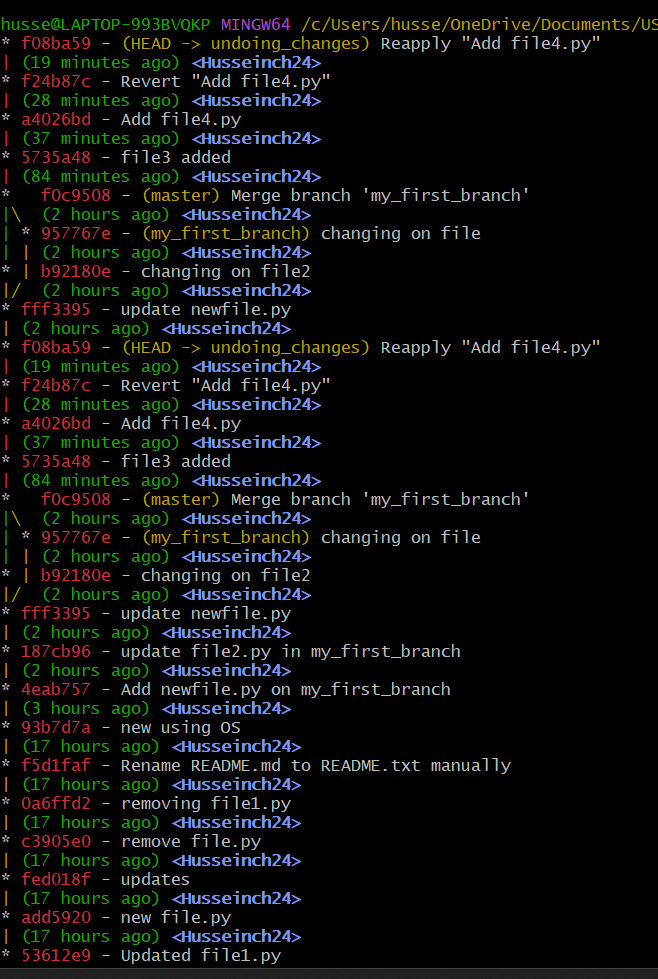
AI-generated content may be incorrect.

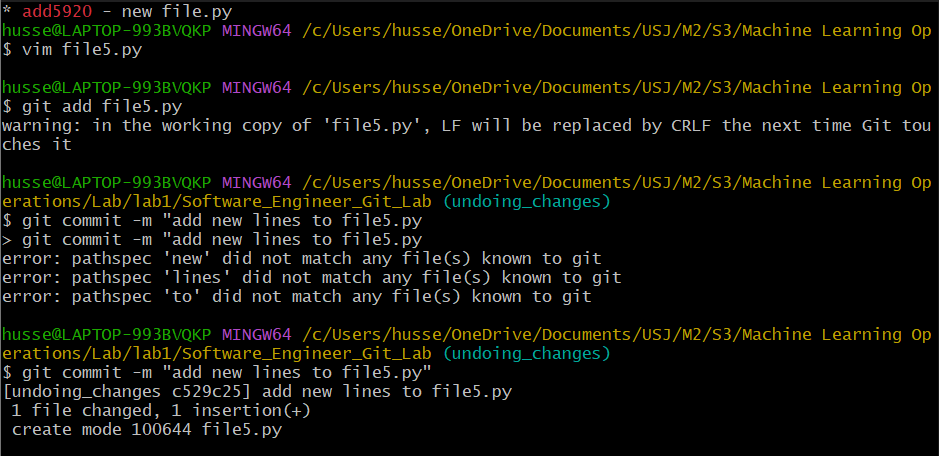
1. a- nothing is staged

b- HEAD refers to the current commit, not moving commits while the HEAD^ is moving to the previous commit.

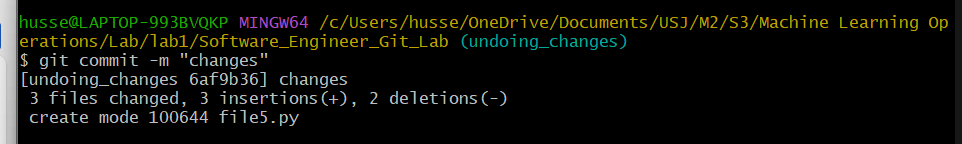
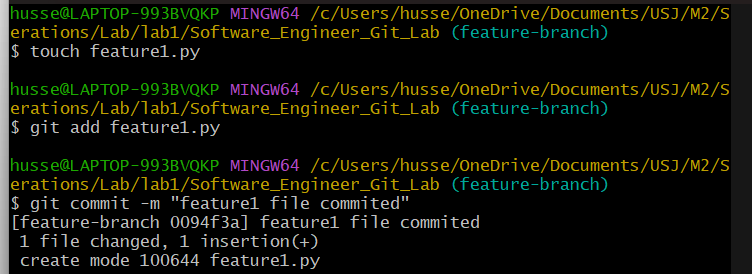
c- Index/staging area

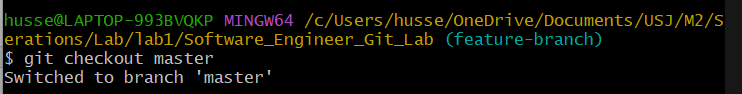
1. a**.** The commit history stays the same, but the working directory is reset to match the last commit.  
   b**.** Git uses HEAD (the most recent commit) as the default value.  
   c**.** If done right after step 9, all uncommitted changes would be lost and files would go back to the last committed state.
2. log:



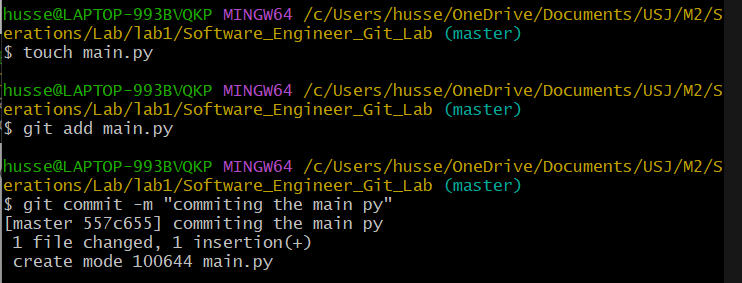
1. file5 created and committed 

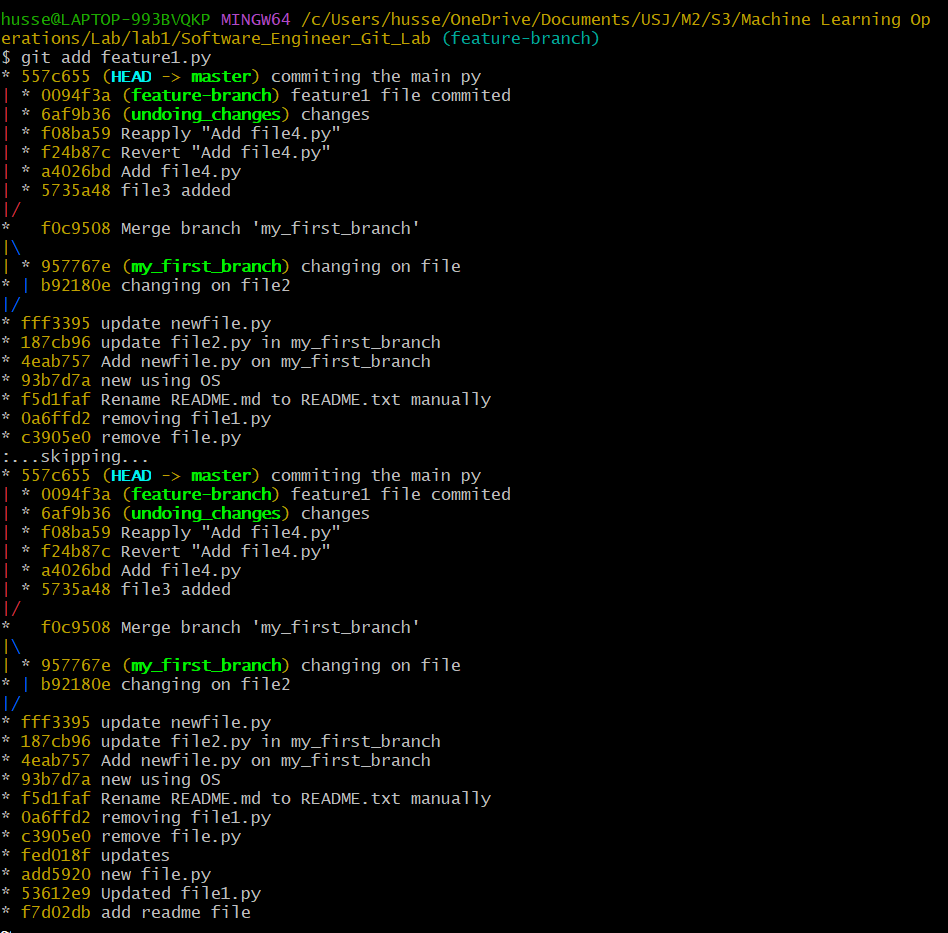
Exercise 4 : Git rebase :

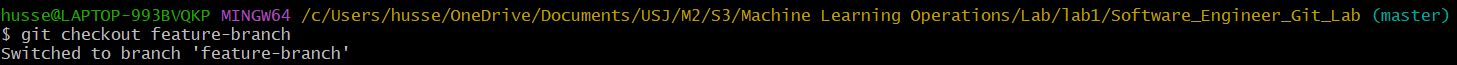
1. yes created using git commit –m“changes”
2. feature1.py created : 
3. back to master branch :

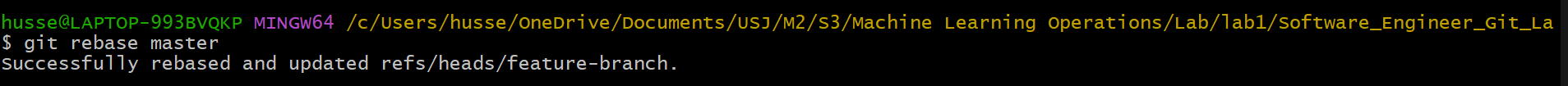
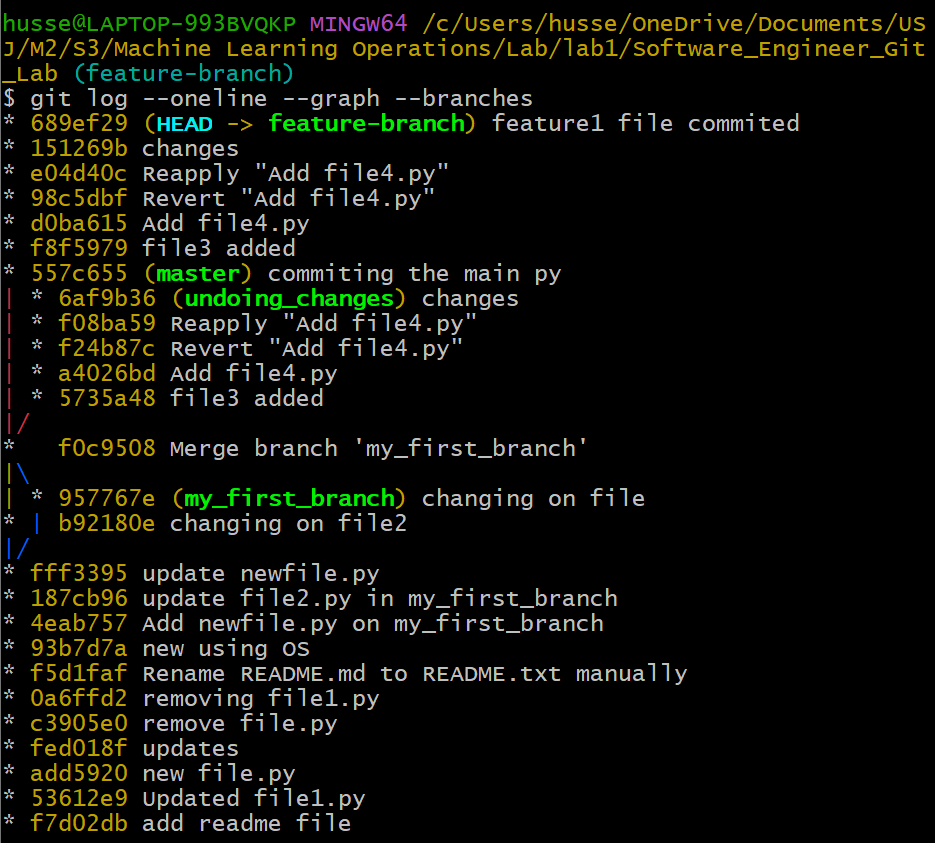


1. main file created :



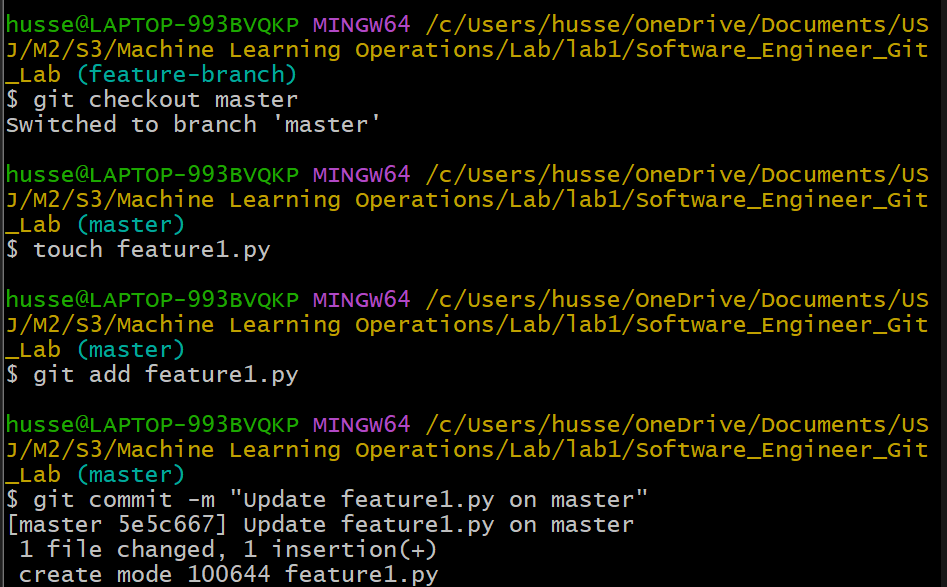
1. check the commit history : 
2. a- rebase is moving the commits from feature – branch on top of master, so that feature is shown as it is created after the last commit on master.

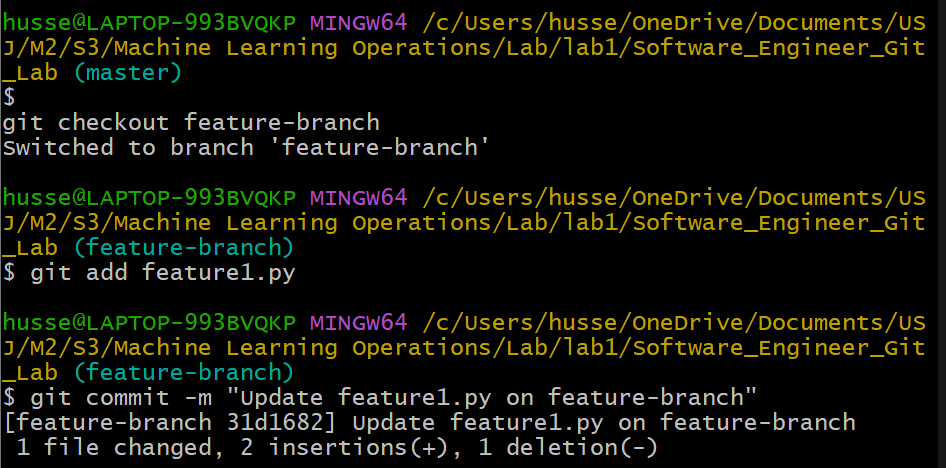


b- check the status and commit history: 

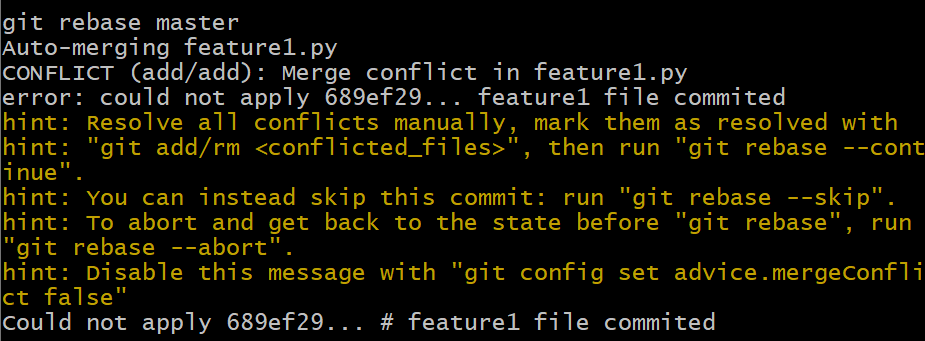
c- Commits from feature-branch now follow master, making history linear without divergence.

d- Rebase rewrites history for a clean linear flow, while merge keeps divergence and adds a merge commit.

1. Back to master and created feature1.py

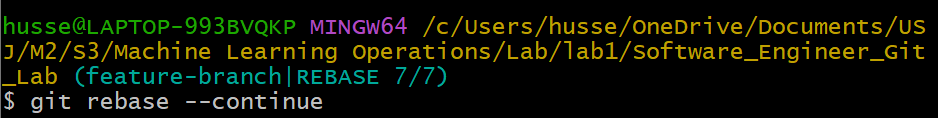
b- back to feature branch

c- Attempt to rebase the feature-branch:

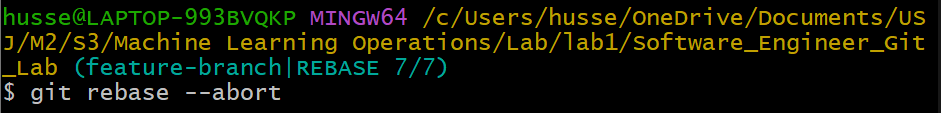


d- status to see the conflict : A screen shot of a computer screen

AI-generated content may be incorrect.

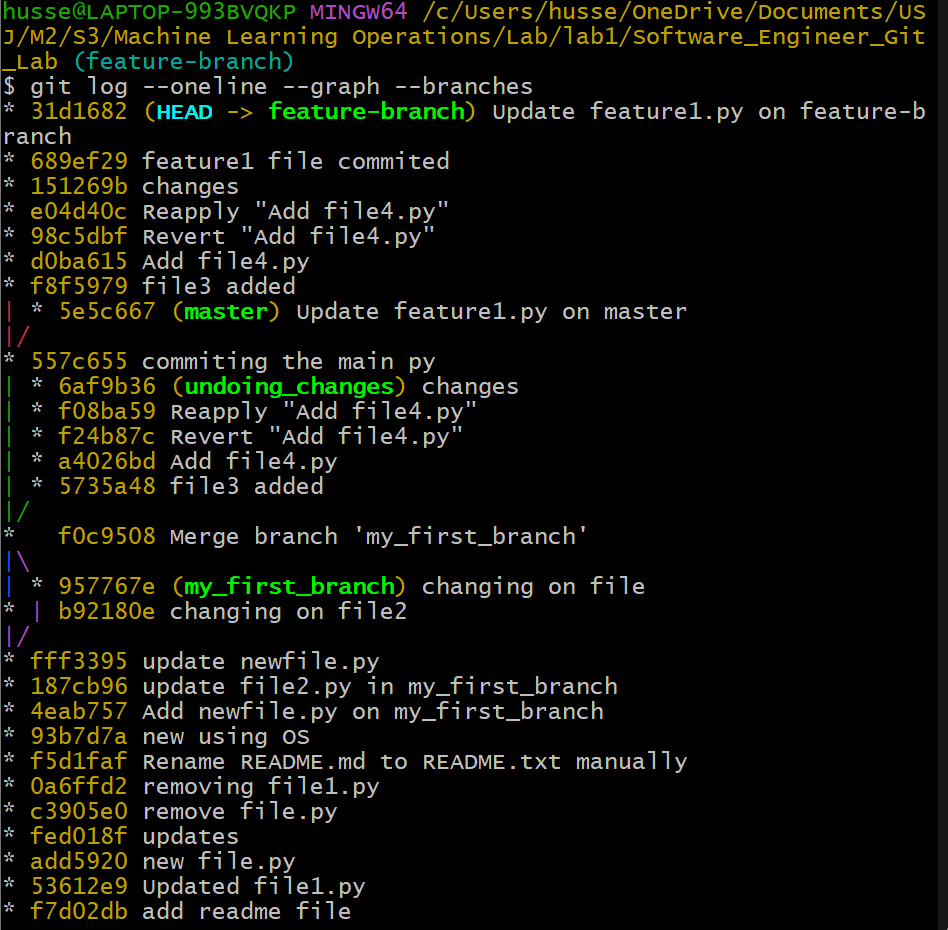
1. A- deleted the conflicted from vs code   
   b- A screenshot of a computer screen

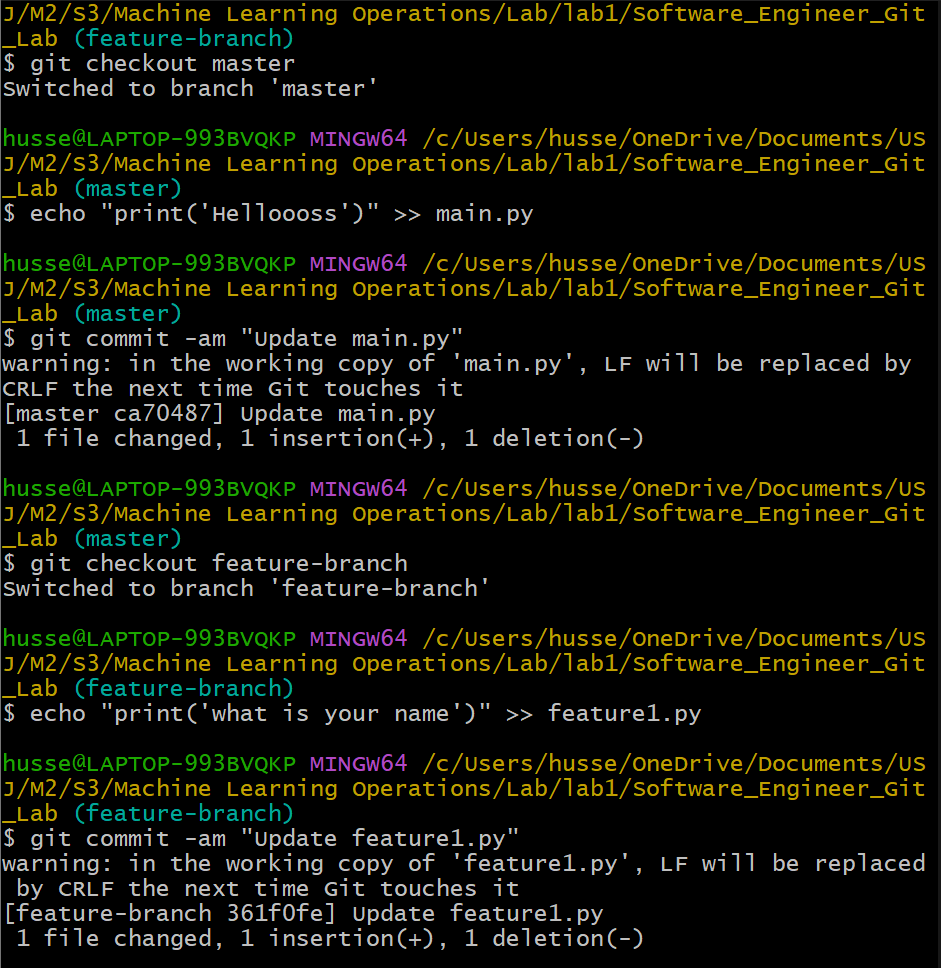
   AI-generated content may be incorrect.

D- abort 

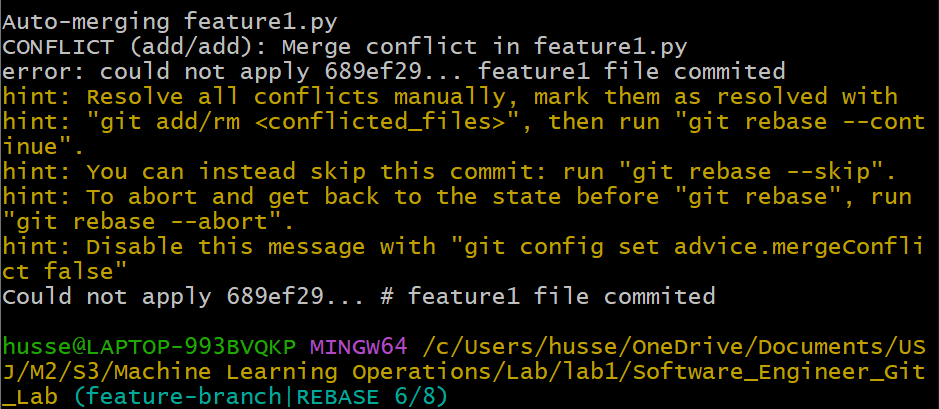
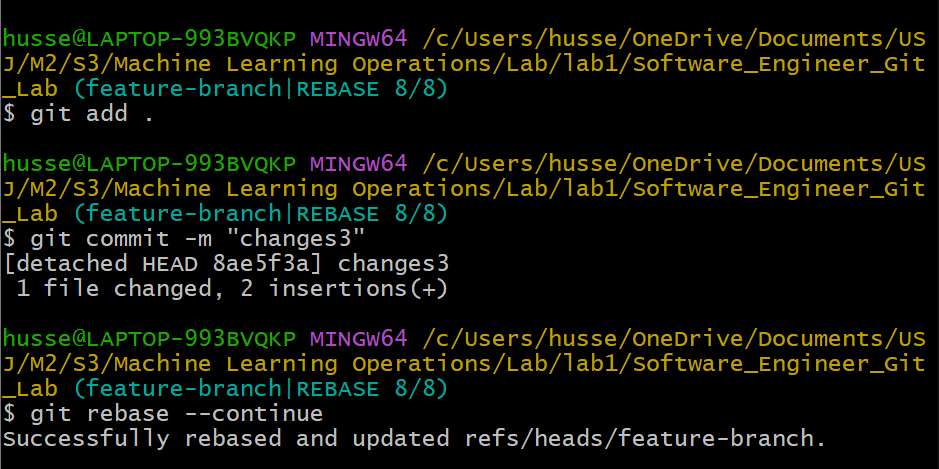
E- i- After aborting, the branch resets to its pre-rebase state with all commits preserved.

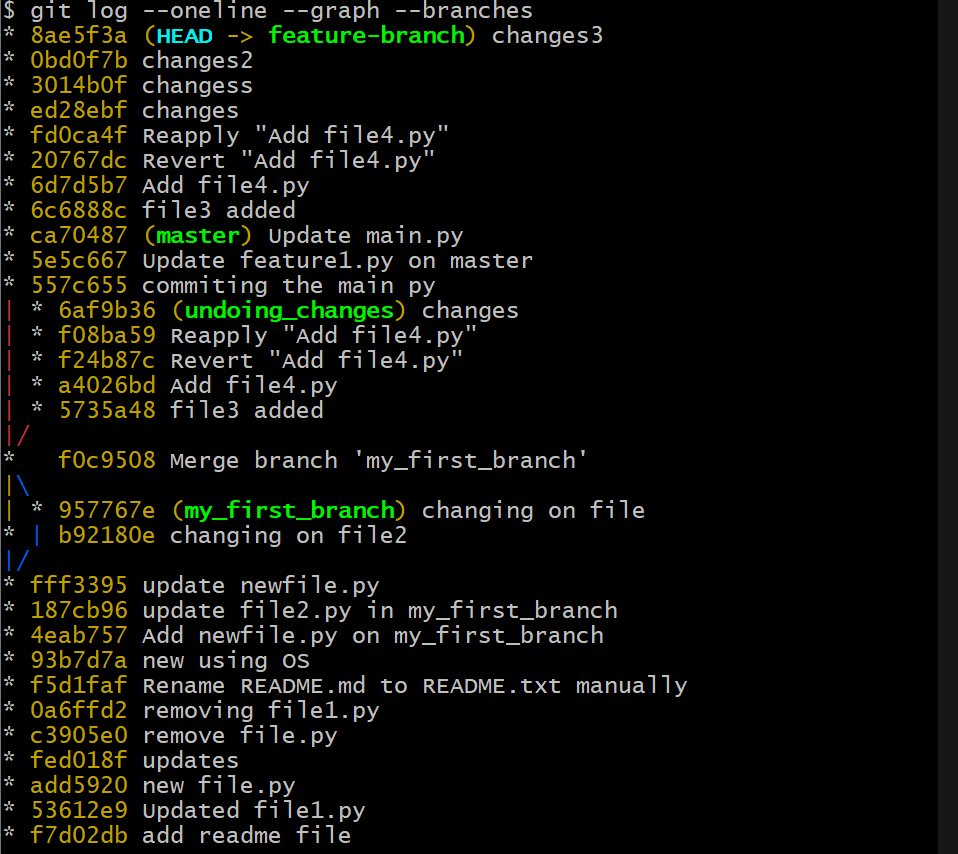
ii- Yes, by repeating the rebase and resolving conflicts properly, it can be finished correctly.

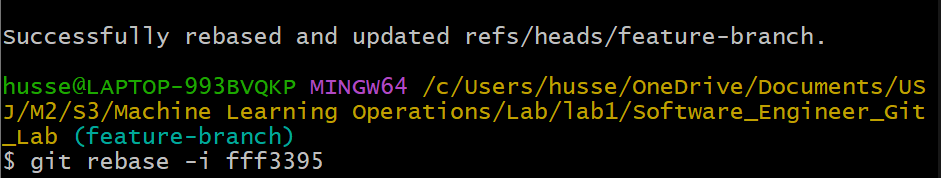
1. Git log : 
2. Feature-branch commits are stacked directly on top of main.
3. Both branches’ changes apply in order, keeping all edits.
4. No merge commit is created, leaving a cleaner linear history.
5. Created some commits

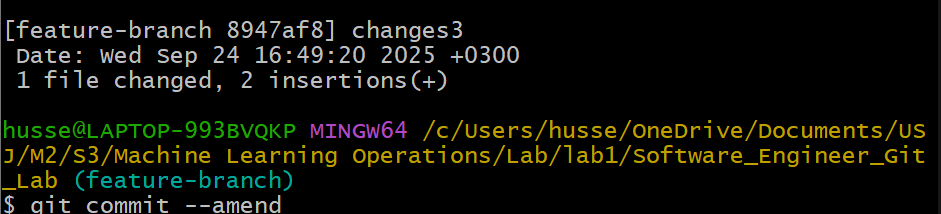
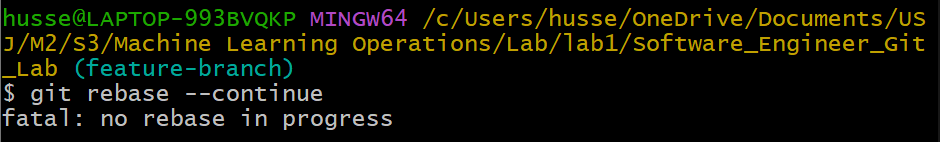


1. A-

B- Squashing merges several commits into one, simplifying history. 

C- modifying a commit during an interactive rebase:

D-

1 - Start with git rebase -i <base-branch> and mark the commit as *edit*.

2- Apply the necessary file changes.

3- Stage them using git add <files>.

4- Update the commit with git commit --amend.

5- Resume with git rebase --continue.

1. Use git log --oneline --graph --branches to inspect the commit history and see how the rebase has altered it.

