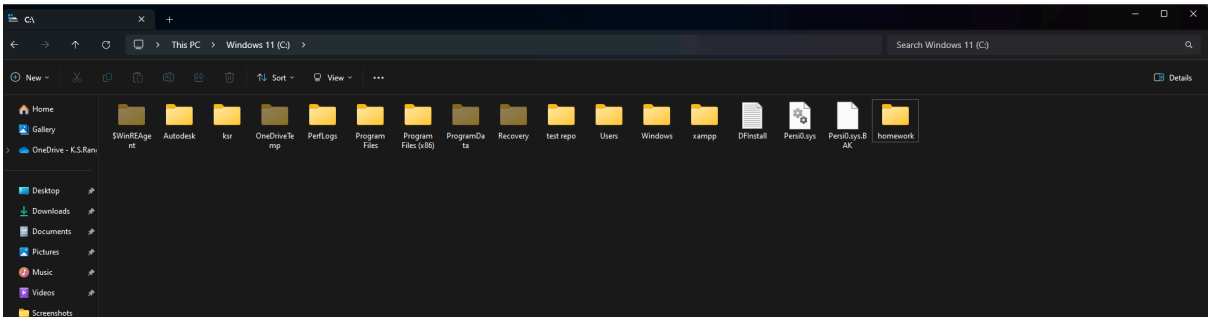
NAME: MURALI NARAYANAN V
REG NO: 73772127134
DEPT: CSBS-IV

12/7/2024

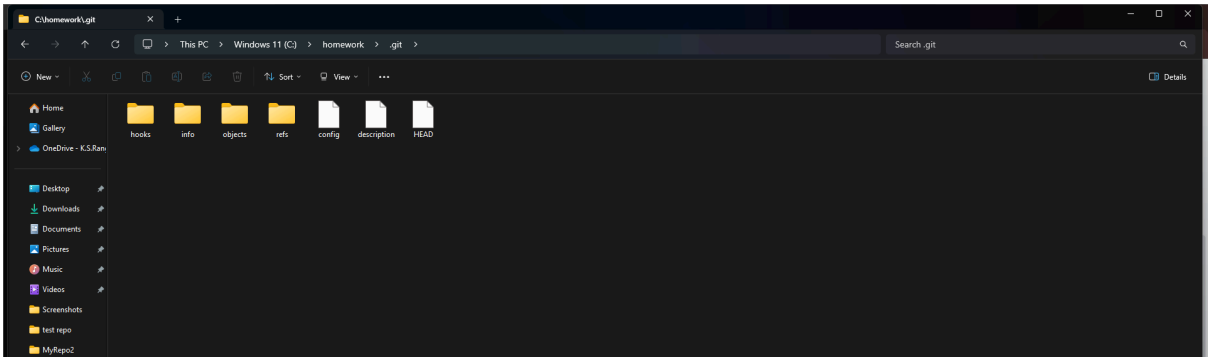
Exercise 1

1. Create a new directory and change into it:

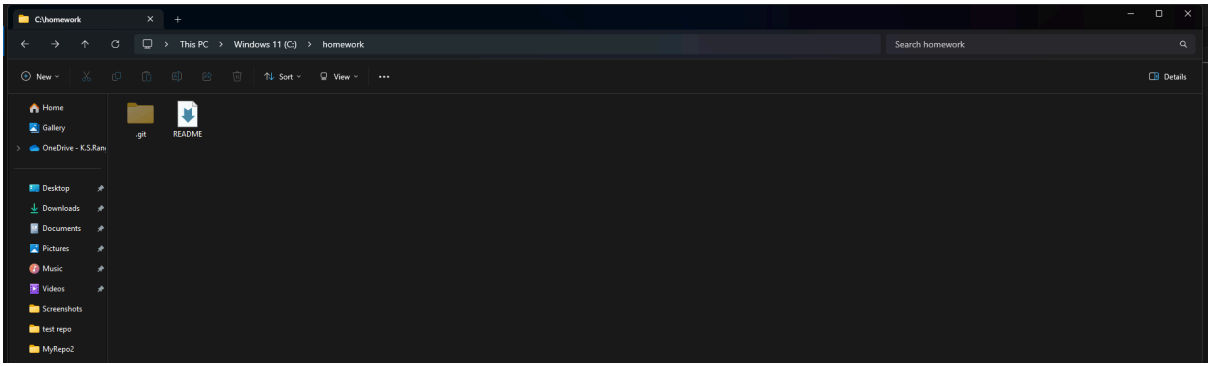


2. Use the init command to create a Git repository in that directory.

3. Observe that there is now a .git directory.



4. Create a README file.



5. Look at the output of the status command; the README you created should appear as an untracked file

```
C:\homework>git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        README.md

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

C:\homework>
```

6. Use the add command to add the new file to the staging area. Again, look at the output of the status command.

```
C:\homework>git add README.md

C:\homework>git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   README.md


C:\homework>
```

7. Now use the commit command to commit the contents of the staging area

```
C:\homework>git commit -m "createed README.md"
[master (root-commit) 86ca6c1] createed README.md
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 README.md

C:\homework>
```

8. Create a src directory and add a couple of files to it.

```
C:\homework>mkdir src

C:\homework>cat >> cfile.c
'cat' is not recognized as an internal or external command,
operable program or batch file.

C:\homework>cat >> javafile.java
'cat' is not recognized as an internal or external command,
operable program or batch file.

C:\homework>
```

9. Use the add command, but name the directory, not the individual files. Use the status command. See how both files have been staged. Commit them.

```
C:\homework\src>git add .

C:\homework\src>git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   cfile.c
        new file:   javafile.java


C:\homework\src>
```

10. Make a change to one of the files. Use the diff command to view the details of the change.

```
C:\homework\src>git diff
diff --git a/src/cfile.c b/src/cfile.c
deleted file mode 100644
index e69de29..0000000

C:\homework\src>
```

11. Next, add the changed file, and notice how it moves to the staging area in the status output. Also observe that the diff command you did before using add now gives no output. Why not? What do you have to do to see a diff of the things in the staging area? (Hint: review the slides if you can't remember.) git

```
C:\homework\src>git add file1.c

C:\homework\src>git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   cfile.c
        new file:   file1.c
        new file:   javafile.java

Changes not staged for commit:
  (use "git add/rm <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        deleted:    cfile.c


C:\homework\src>git diff
diff --git a/src/cfile.c b/src/cfile.c
deleted file mode 100644
index e69de29..0000000

C:\homework\src>
```

- **Only after the commit comment only diff comment will be able to work or access**

12. Now – without committing – make another change to the same file you changed in step 10. Look at the status output, and the diff output. Notice how you can have both staged and unstaged changes, even when you're talking about a single file. Observe the difference when you use the add command to stage the latest round of changes. Finally, commit them. You should now have started to get a feel for the staging area.

```
C:\homework\src>git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   cfile.c
        new file:   file1.c
        new file:   javafile.java

Changes not staged for commit:
  (use "git add/rm <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        deleted:    cfile.c
        modified:   file1.c


C:\homework\src>git diff
diff --git a/src/cfile.c b/src/cfile.c
deleted file mode 100644
index e69de29..0000000
diff --git a/src/file1.c b/src/file1.c
index e69de29..cc55f90 100644
--- a/src/file1.c
+++ b/src/file1.c
@@ -0,0 +1,2 @@
+mural;i narayanan v
+deprt cbs-ibv
\ No newline at end of file

C:\homework\src>git commit -m "step 12"
[master d410d14] step 12
 3 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 src/cfile.c
 create mode 100644 src/file1.c
 create mode 100644 src/javafile.java

C:\homework\src>
```

13. Use the log command in order to see all of the commits you made so far.

```
C:\homework\src>git log
commit d410d145f5afa02cfa563f5b519b09f0daeea65b (HEAD -> master)
Author: MuraliNarayanan01 <narayananmurali01@gmail.com>
Date:   Fri Jul 12 18:19:45 2024 +0530

    step 12

commit 86ca6c1a4b3f0b19e9173879db4b65e32df2a4f3
Author: MuraliNarayanan01 <narayananmurali01@gmail.com>
Date:   Fri Jul 12 18:00:25 2024 +0530

    createed README.md

C:\homework\src>
```

14. Use the show command to look at an individual commit. How many characters of the commit identifier can you get away with typing at a minimum?

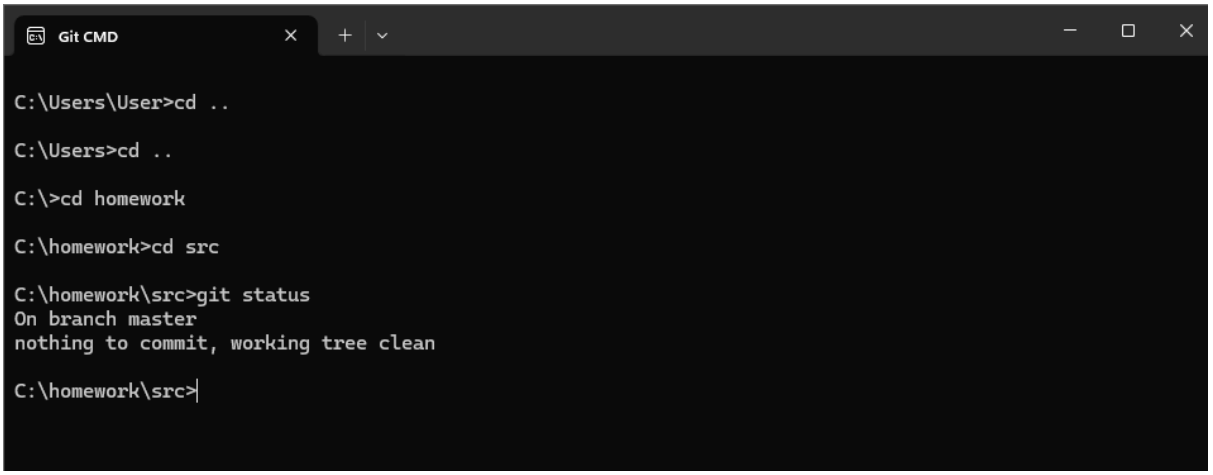
```
C:\homework\src>git show d410
commit d410d145f5afa02cfa563f5b519b09f0daeea65b (HEAD -> master)
Author: MuraliNarayanan01 <narayananmurali01@gmail.com>
Date:   Fri Jul 12 18:19:45 2024 +0530

    step 12

diff --git a/src/cfile.c b/src/cfile.c
new file mode 100644
index 0000000..e69de29
diff --git a/src/file1.c b/src/file1.c
new file mode 100644
index 0000000..e69de29
diff --git a/src/javafile.java b/src/javafile.java
new file mode 100644
index 0000000..e69de29

C:\homework\src>
```

15. Make a couple more commits, at least one of which should add an extra file

```
C:\homework\src>git add .

C:\homework\src>git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   file1.c
        renamed:    cfile.c -> txt.txt


C:\homework\src>git commit -m "final commit"
[master 8c8b7b7] final commit
 2 files changed, 2 insertions(+)
 rename src/{cfile.c => txt.txt} (100%)

C:\homework\src>
```

Exercise 2

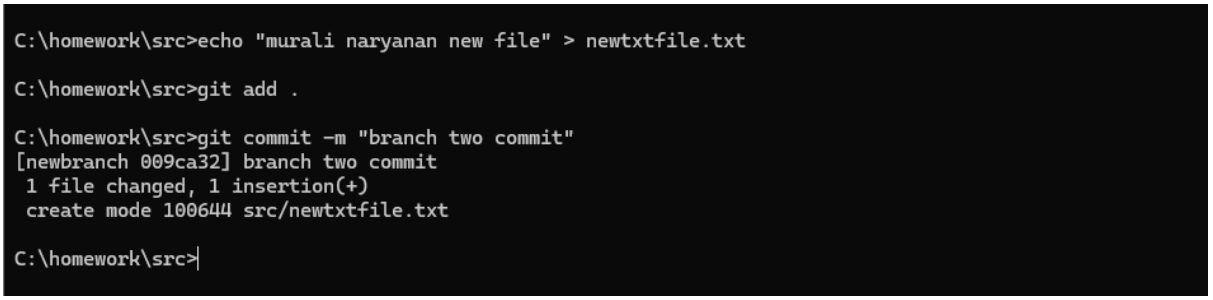
1. Run the status command. Notice how it tells you what branch you are in



2. Use the branch command to create a new branch.

3. Use the checkout command to switch to it.

4. Make a couple of commits in the branch – perhaps adding a new file and/or editing existing ones.



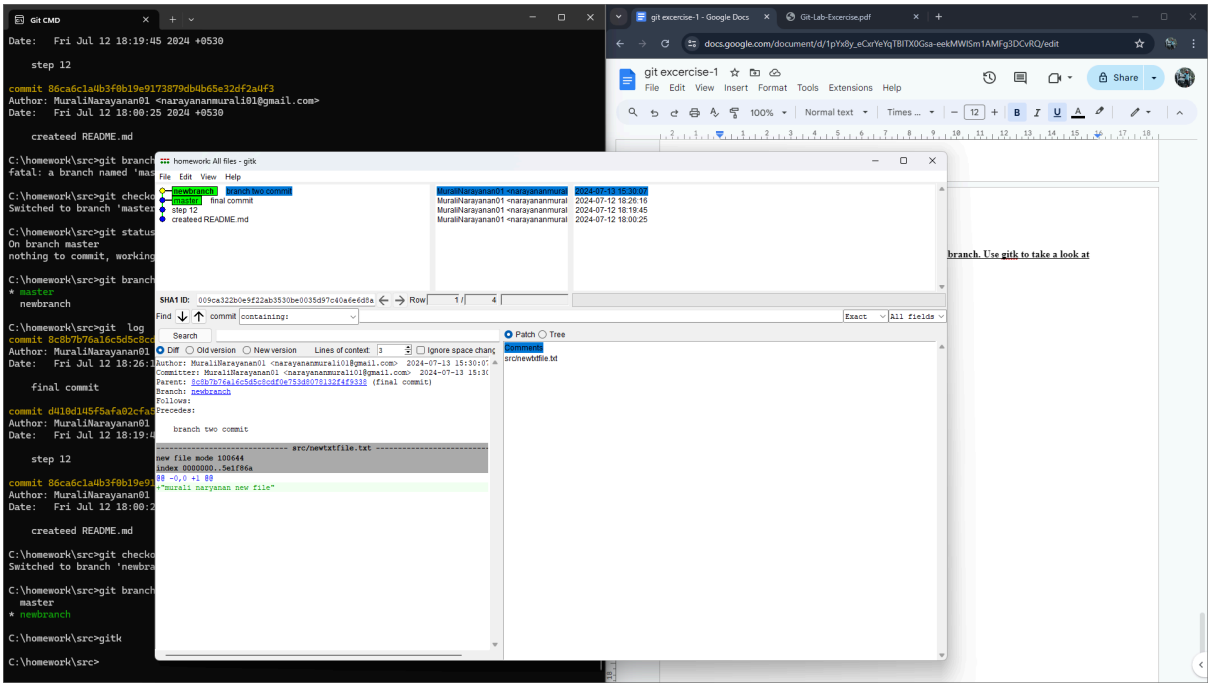
5. Use the log command to see the latest commits. The two you just made should be at the top of the list.

```
C:\homework\src>git log
commit 009ca322b0e9f22ab3530be0035d97c40a6e6d8a (HEAD -> newbranch)
Author: MuraliNarayanan01 <narayananmurali01@gmail.com>
Date:   Sat Jul 13 15:30:07 2024 +0530

    branch two commit

commit 8c8b7b76a16c5d5c8cdf0e753d8078132f4f9338 (master)
Author: MuraliNarayanan01 <narayananmurali01@gmail.com>
Date:   Fri Jul 12 18:26:16 2024 +0530

    final commit

commit d410d145f5afa02cfa563f5b519b09f0daeea65b
Author: MuraliNarayanan01 <narayananmurali01@gmail.com>
Date:   Fri Jul 12 18:19:45 2024 +0530

    step 12

commit 86ca6c1a4b3f0b19e9173879db4b65e32df2a4f3
Author: MuraliNarayanan01 <narayananmurali01@gmail.com>
Date:   Fri Jul 12 18:00:25 2024 +0530

    createed README.md
```

6. Use the checkout command to switch back to the master branch. Run log again. Notice your commits don't show up now. Check the files also – they should have their original contents

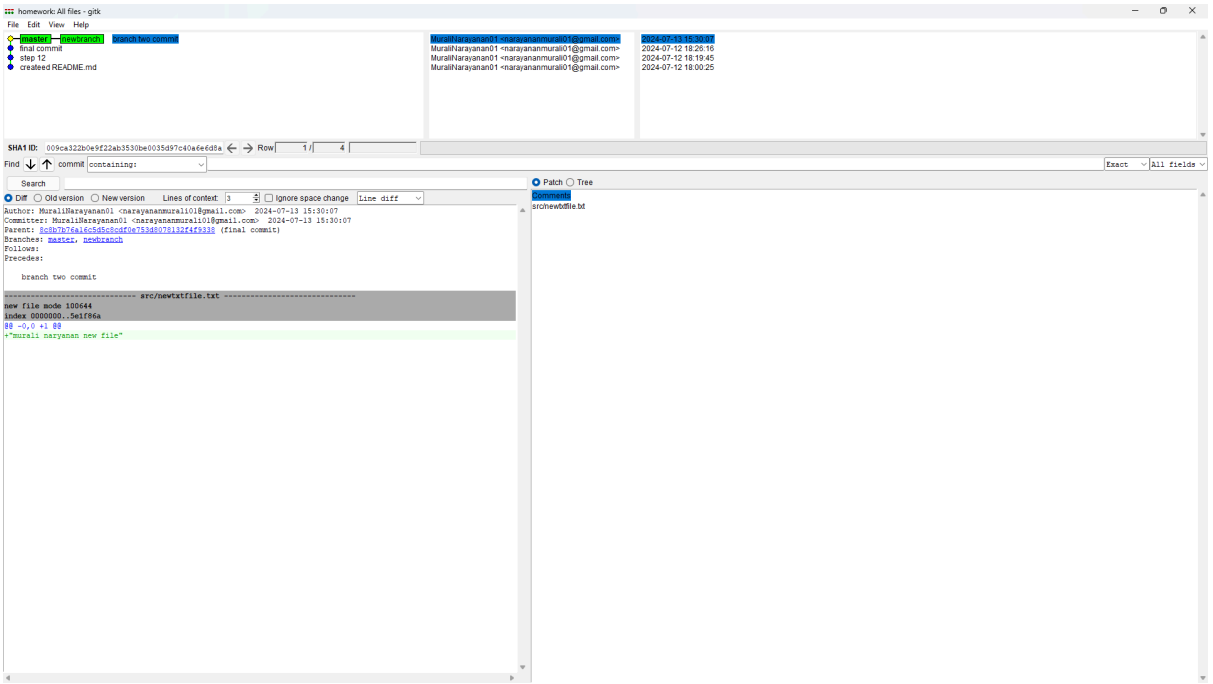
```
C:\homework\src>git checkout master
Switched to branch 'master'

C:\homework\src>git status
On branch master
nothing to commit, working tree clean

C:\homework\src>git branch
* master
  newbranch

C:\homework\src>git  log
commit 8c8b7b76a16c5d5c8cdf0e753d8078132f4f9338 (HEAD -> master)
Author: MuraliNarayanan01 <narayananmurali01@gmail.com>
Date:   Fri Jul 12 18:26:16 2024 +0530

    final commit

commit d410d145f5afa02cfa563f5b519b09f0daeea65b
Author: MuraliNarayanan01 <narayananmurali01@gmail.com>
Date:   Fri Jul 12 18:19:45 2024 +0530

    step 12

commit 86ca6c1a4b3f0b19e9173879db4b65e32df2a4f3
Author: MuraliNarayanan01 <narayananmurali01@gmail.com>
Date:   Fri Jul 12 18:00:25 2024 +0530

    createed README.md
```

7. Use the checkout command to switch back to your branch. Use gitk to take a look at the commit graph; notice it's linear.



8. Now checkout the master branch again. Use the merge command to merge your branch in to it. Look for information about it having been a fast-forward merge. Look at git log, and see that there is no merge commit. Take a look in gitk and see how the DAG is linear.

```
C:\homework\src>git checkout master
Switched to branch 'master'

C:\homework\src>git branch
* master
  newbranch

C:\homework\src>git merge newbranch
Updating 8c8b7b7..009ca32
Fast-forward
 src/newtxtfile.txt | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 src/newtxtfile.txt

C:\homework\src>git log
commit 009ca322b0e9f22ab3530be0035d97c40a6e6d8a (HEAD -> master, newbranch)
Author: MuraliNarayanan01 <narayananmurali01@gmail.com>
Date:   Sat Jul 13 15:30:07 2024 +0530

    branch two commit

commit 8c8b7b76a16c5d5c8cdf0e753d8078132f4f9338
Author: MuraliNarayanan01 <narayananmurali01@gmail.com>
Date:   Fri Jul 12 18:26:16 2024 +0530

    final commit

commit d410d145f5afa02cfa563f5b519b09f0daeea65b
Author: MuraliNarayanan01 <narayananmurali01@gmail.com>
Date:   Fri Jul 12 18:19:45 2024 +0530

    step 12

commit 86ca6c1a4b3f0b19e9173879db4b65e32df2a4f3
Author: MuraliNarayanan01 <narayananmurali01@gmail.com>
Date:   Fri Jul 12 18:00:25 2024 +0530

    createed README.md
```



9. Switch back to your branch. Make a couple more commits

```
C:\homework\src>git checkout newbranch
Switched to branch 'newbranch'

C:\homework\src>echo "first commit of couple" > newjava1.java

C:\homework\src>git add .

C:\homework\src>git commit -m "commit 1 off 2"
[newbranch 2e0902c] commit 1 off 2
 1 file changed, 1 insertion(+)
 create mode 100644 src/newjava1.java

C:\homework\src>echo "second commit of couple" > newjava2.java

C:\homework\src>git add .

C:\homework\src>git commit -m "commit 2 off 2"
[newbranch cf5249e] commit 2 off 2
 1 file changed, 1 insertion(+)
 create mode 100644 src/newjava2.java
```

10. Switch back to master. Make a commit there, which should edit a different file from the ones you touched in your branch – to be sure there is no conflict.

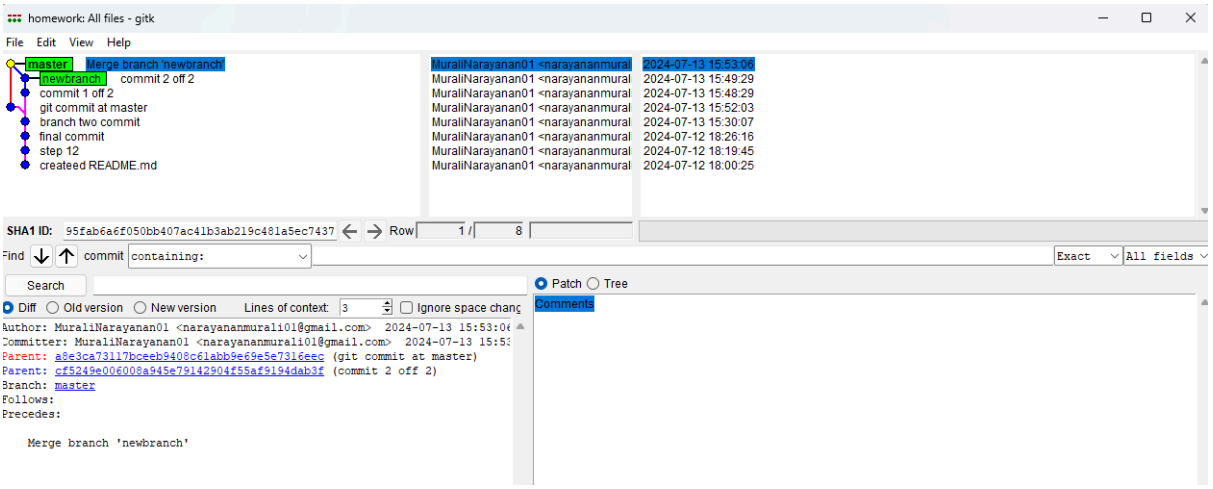
```
C:\homework\src>git checkout master
Switched to branch 'master'

C:\homework\src>echo "this is master commit" > masterjava.java

C:\homework\src>git commit -m "git commit at master"
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        masterjava.java

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

C:\homework\src>git add .

C:\homework\src>git commit -m "git commit at master"
[master a8e3ca7] git commit at master
 1 file changed, 1 insertion(+)
 create mode 100644 src/masterjava.java
```

11. Now merge your branch again. (Aside: you don't need to do anything to inform Git that you only want to merge things added since your previous merge. Due to the way Git works, that kind of issue simply does not come up, unlike in early versions of Subversion.)

```
C:\homework\src>git merge newbranch
Merge made by the 'ort' strategy.
 src/newjava1.java | 1 +
 src/newjava2.java | 1 +
 2 files changed, 2 insertions(+)
 create mode 100644 src/newjava1.java
 create mode 100644 src/newjava2.java
```

12. Look at git log. Notice that there is a merge commit. Also look in gitk. Notice the DAG now shows how things forked, and then were joined up again by a merge commit.

```
C:\homework\src>git log
commit 95fab6a6f050bb407ac41b3ab219c481a5ec7437 (HEAD -> master)
Merge: a8e3ca7 cf5249e
Author: MuraliNarayanan01 <narayananmurali01@gmail.com>
Date:   Sat Jul 13 15:53:06 2024 +0530

    Merge branch 'newbranch'

commit a8e3ca73117bceeb9408c61abb9e69e5e7316eec
Author: MuraliNarayanan01 <narayananmurali01@gmail.com>
Date:   Sat Jul 13 15:52:03 2024 +0530

    git commit at master

commit cf5249e006008a945e79142904f55af9194dab3f (newbranch)
Author: MuraliNarayanan01 <narayananmurali01@gmail.com>
Date:   Sat Jul 13 15:49:29 2024 +0530

    commit 2 off 2

commit 2e0902cfdac2ec2be48bbcc43db3b11211faeeda
Author: MuraliNarayanan01 <narayananmurali01@gmail.com>
Date:   Sat Jul 13 15:48:29 2024 +0530

    commit 1 off 2

commit 009ca322b0e9f22ab3530be0035d97c40a6e6d8a
Author: MuraliNarayanan01 <narayananmurali01@gmail.com>
Date:   Sat Jul 13 15:30:07 2024 +0530

    branch two commit

commit 8c8b7b76a16c5d5c8cdf0e753d8078132f4f9338
Author: MuraliNarayanan01 <narayananmurali01@gmail.com>
Date:   Fri Jul 12 18:26:16 2024 +0530

    final commit

commit d410d145f5afa02cfa563f5b519b09f0daeea65b
Author: MuraliNarayanan01 <narayananmurali01@gmail.com>
Date:   Fri Jul 12 18:19:45 2024 +0530

    step 12

commit 86ca6c1a4b3f0b19e9173879db4b65e32df2a4f3
Author: MuraliNarayanan01 <narayananmurali01@gmail.com>
Date:   Fri Jul 12 18:00:25 2024 +0530

    createed README.md
```

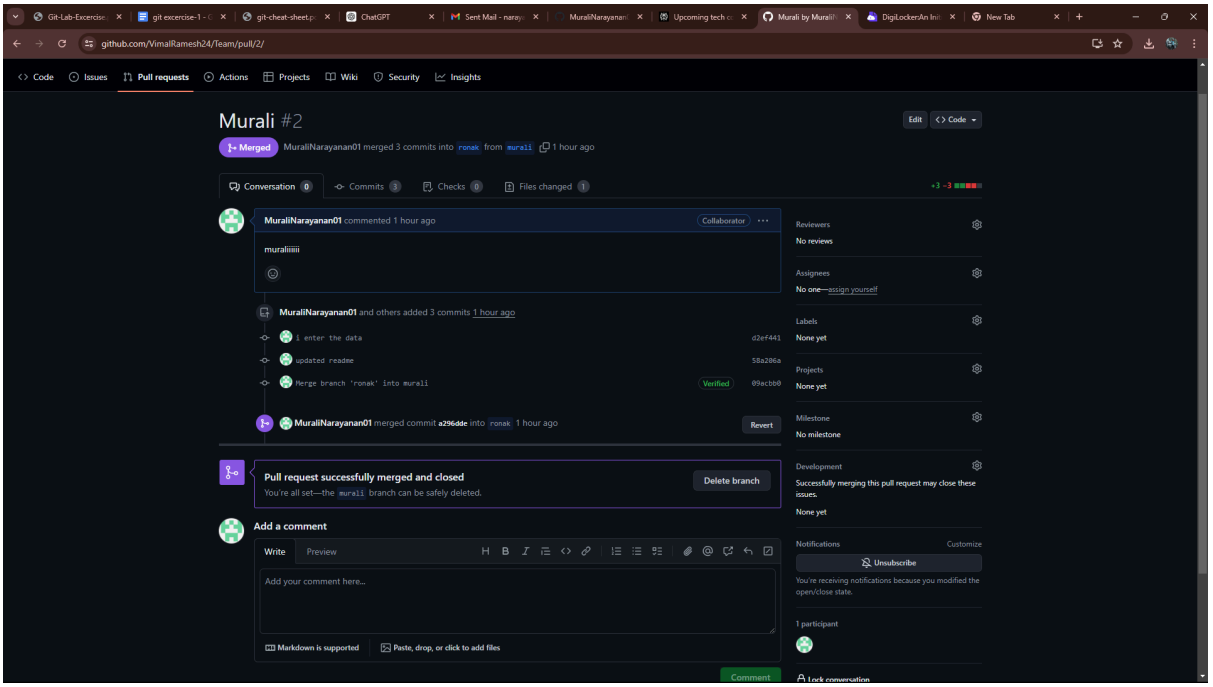


Exercise 3

1. First, one person in the group should create a public repository using their GitHub account.

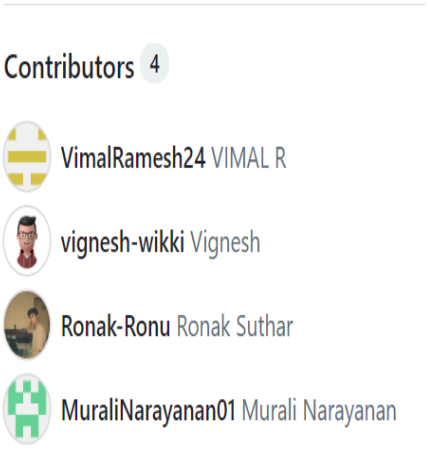
```
C:\homework>git clone https://github.com/VimalRamesh24/Team.git
Cloning into 'Team'...
remote: Enumerating objects: 25, done.
remote: Counting objects: 100% (25/25), done.
remote: Compressing objects: 100% (17/17), done.
remote: Total 25 (delta 1), reused 16 (delta 0), pack-reused 0
Receiving objects: 100% (25/25), 4.32 KiB | 4.32 MiB/s, done.
Resolving deltas: 100% (1/1), done.
```

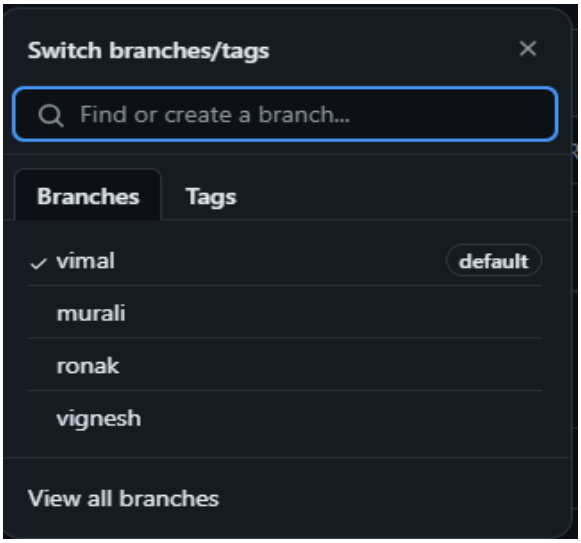
```
C:\homework>git push murali
fatal: The current branch newbranch has no upstream branch.
To push the current branch and set the remote as upstream, use

    git push --set-upstream murali newbranch

To have this happen automatically for branches without a tracking
upstream, see 'push.autoSetupRemote' in 'git help config'.


C:\homework>git push newbranch
fatal: The current branch newbranch has no upstream branch.
To push the current branch and set the remote as upstream, use

    git push --set-upstream newbranch newbranch

To have this happen automatically for branches without a tracking
upstream, see 'push.autoSetupRemote' in 'git help config'.


C:\homework>git branch murali

C:\homework>git branch
  master
  murali
```



2. This same person should then follow the instructions from GitHub to add a remote, and then push their repository. Do not forget the –u flag, as suggested by GitHub!

3. All of the other members of the group should then be added as collaborators, so they can commit to the repository also.





4. Next, everyone else in the group should clone the repository from GitHub. Verify that the context of the repository is what is expected.

```
C:\>cd teamrepo

C:\teamrepo>git clone https://github.com/VimalRamesh24/TeamRepo.git
Cloning into 'TeamRepo'...
remote: Enumerating objects: 6, done.
remote: Counting objects: 100% (6/6), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 6 (delta 0), reused 3 (delta 0), pack-reused 0
Receiving objects: 100% (6/6), done.
```

5. One of the group members who just cloned should now make a local commit, then push it. Everyone should verify that when they pull, that commit is added to their local repository (use git log to check for it).

```
C:\teamrepo\TeamRepo>git pull
error: Pulling is not possible because you have unmerged files.
hint: Fix them up in the work tree, and then use 'git add/rm <file>'
hint: as appropriate to mark resolution and make a commit.
fatal: Exiting because of an unresolved conflict.

C:\teamrepo\TeamRepo>git log
commit 74ff755c28aad5d3f8585bfe4e14472af84cd566 (HEAD -> vimal, origin/murali, newmurali)
Author: MuraliNarayanan01 <narayananmurali01@gmail.com>
Date:   Tue Jul 16 10:08:35 2024 +0530

    murali updated file

commit d50b7b4f728825eaaa5c0c8b99ac297655604e3d
Author: VIMAL R <123486203+VimalRamesh24@users.noreply.github.com>
Date:   Tue Jul 16 09:56:05 2024 +0530

    Update README.md

commit cc73fa2c32d6ee6d3ec1dcf3e96cd8bf5d7fefa4
Author: VimalRamesh24 <vimalsaritha2004@gmail.com>
Date:   Tue Jul 16 09:52:06 2024 +0530

    first commit

C:\teamrepo\TeamRepo>
```

Exercise 4

1. Make a commit, and make a silly typo in the commit message.

```
C:\homework>git commit -m "silly typo "
[master 1c9b675] silly typo
 1 file changed, 1 insertion(+)
 create mode 100644 newday.txt

C:\homework>git log
commit 1c9b6755d711ecd6ebf9b98ae45e0b7f8895efa5 (HEAD -> master)
Author: MuraliNarayanan01 <narayananmurali01@gmail.com>
Date:   Mon Jul 15 09:56:11 2024 +0530

    silly typo
```

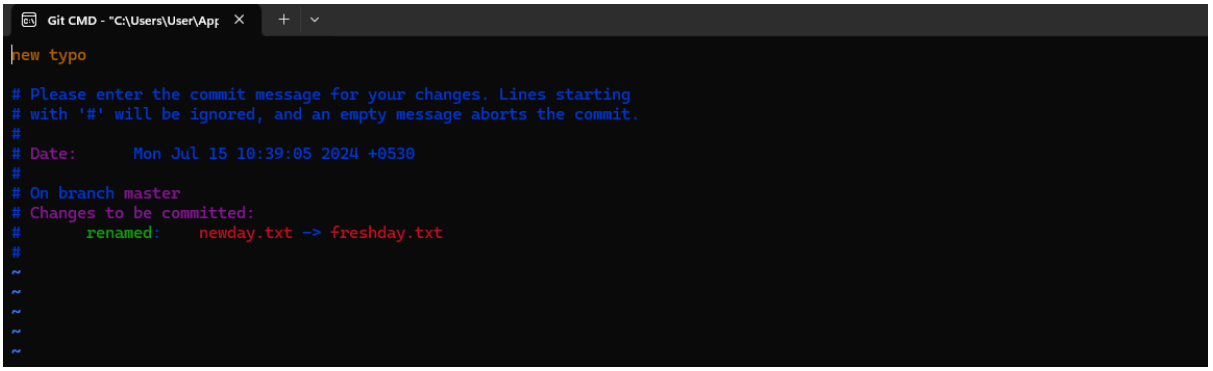
2. Use the --amend flag to enable you to fix the commit message.

```
Git CMD - "C:\Users\User\Apf
silly typo

# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
#
# Date:      Mon Jul 15 09:56:11 2024 +0530
#
# On branch master
# Changes to be committed:
#       new file:   newday.txt
#
~
~
~
```

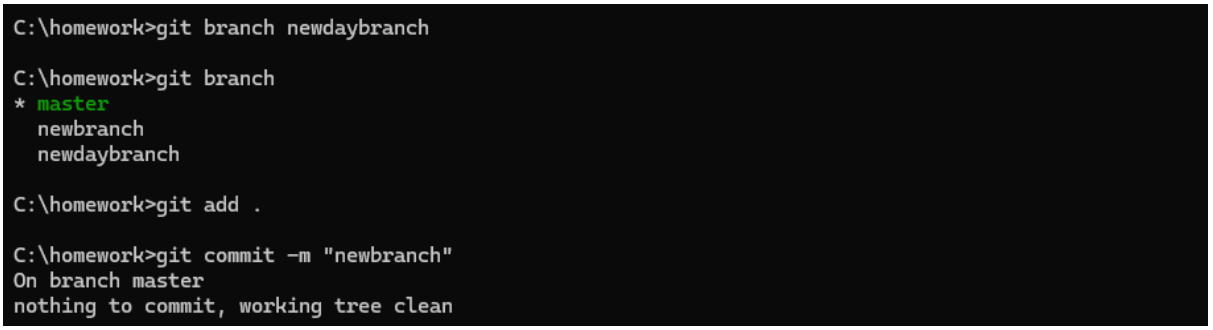
3. Look at the log and notice how the mistake is magically gone.

```
Git CMD - "C:\Users\User\Apf
C:\>cd homework

C:\homework>git log
commit ade4397fe5d608f437c2eae4345372a94828777d (HEAD -> master)
Author: MuraliNarayanan01 <narayananmurali01@gmail.com>
Date:   Mon Jul 15 09:56:11 2024 +0530

    silly typo

commit 95fab6a6f050bb407ac41b3ab219c481a5ec7437
Merge: a8e3ca7 cf5249e
Author: MuraliNarayanan01 <narayananmurali01@gmail.com>
Date:   Sat Jul 13 15:53:06 2024 +0530

    Merge branch 'newbranch'

commit a8e3ca73117bceeb9408c61abb9e69e5e7316eec
Author: MuraliNarayanan01 <narayananmurali01@gmail.com>
Date:   Sat Jul 13 15:52:03 2024 +0530

    git commit at master

commit cf5249e006008a945e79142904f55af9194dab3f (newbranch)
Author: MuraliNarayanan01 <narayananmurali01@gmail.com>
Date:   Sat Jul 13 15:49:29 2024 +0530

    commit 2 off 2

commit 2e0902cfdac2ec2be48bbcc43db3b11211faeeda
Author: MuraliNarayanan01 <narayananmurali01@gmail.com>
Date:   Sat Jul 13 15:48:29 2024 +0530

:...skipping...
commit ade4397fe5d608f437c2eae4345372a94828777d (HEAD -> master)
Author: MuraliNarayanan01 <narayananmurali01@gmail.com>
Date:   Mon Jul 15 09:56:11 2024 +0530

    silly typo

commit 95fab6a6f050bb407ac41b3ab219c481a5ec7437
Merge: a8e3ca7 cf5249e
Author: MuraliNarayanan01 <narayananmurali01@gmail.com>
Date:   Sat Jul 13 15:53:06 2024 +0530

    Merge branch 'newbranch'

commit a8e3ca73117bceeb9408c61abb9e69e5e7316eec
Author: MuraliNarayanan01 <narayananmurali01@gmail.com>
Date:   Sat Jul 13 15:52:03 2024 +0530

    git commit at master

commit cf5249e006008a945e79142904f55af9194dab3f (newbranch)
Author: MuraliNarayanan01 <narayananmurali01@gmail.com>
```

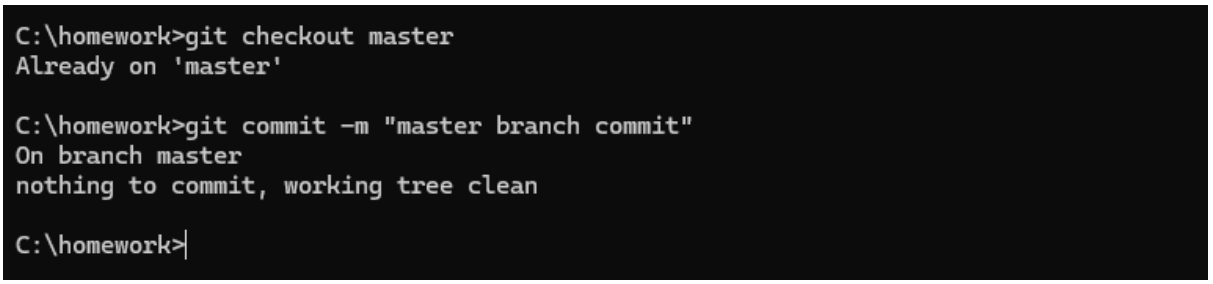
4. Now make a commit where you make a typo in one of the files. Once again, use --amend to magic away your problems.



5. Create a branch. Make a commit.



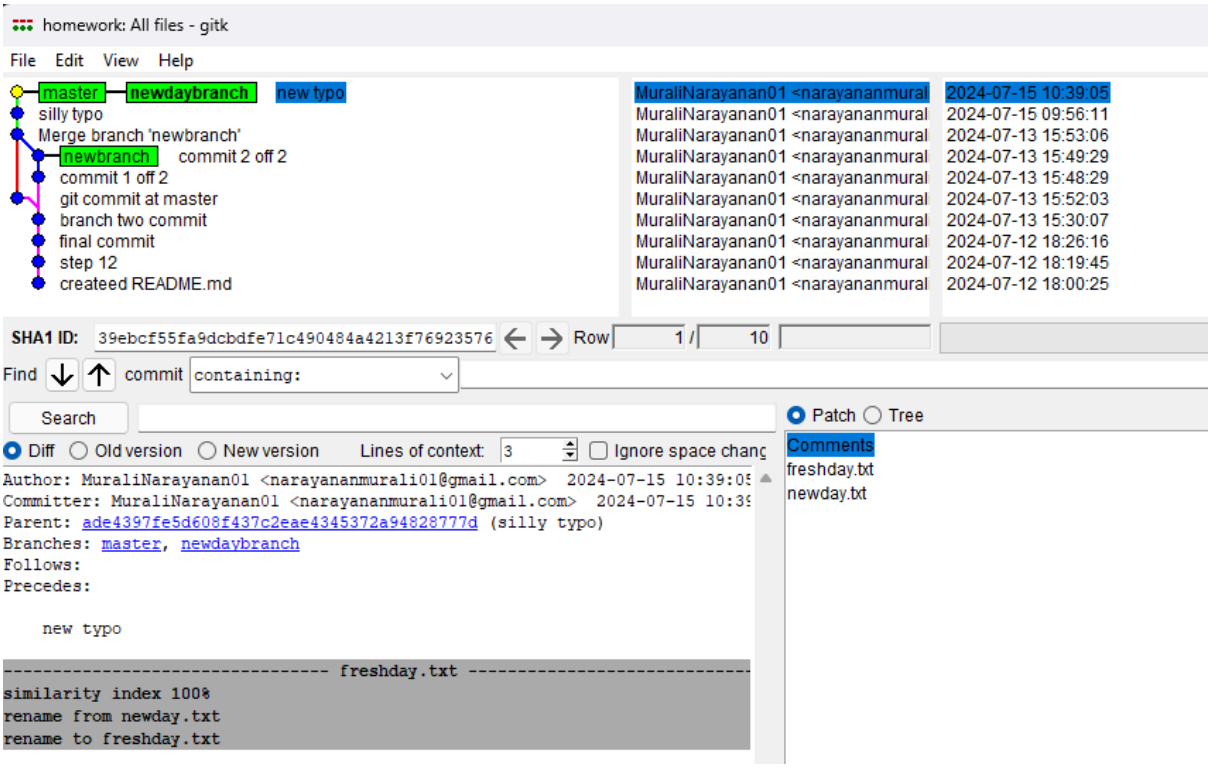
6. Now switch back to your master branch. Make a (non-conflicting) commit there also.



7. Now switch back to your branch.

8. Use the rebase command in your branch. Look at the DAG in gitk, and note that you have the commit from the master branch, but no merge commit.

```
C:\homework>git rebase master
Current branch newdaybranch is up to date.

C:\homework>gitk

C:\homework>
```

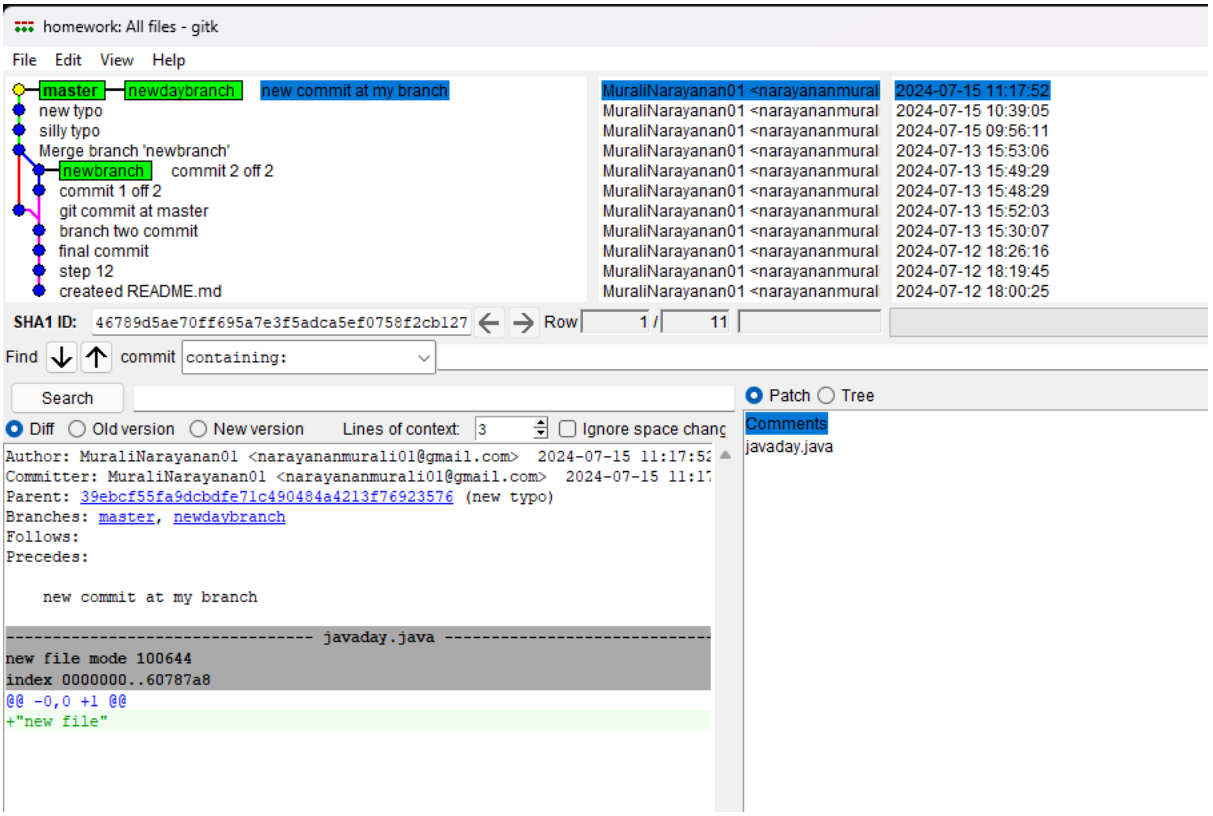


9. Make one more commit in your branch.

```
C:\homework>echo "new file" >javaday.java

C:\homework>git add .

C:\homework>git commit -m "new commit at my branch"
[newdaybranch 46789d5] new commit at my branch
 1 file changed, 1 insertion(+)
 create mode 100644 javaday.java
```

10. Return to master. Merge your branch. Notice how, thanks to the rebase, this is a fastforward merge.

```
C:\homework>git checkout master
Switched to branch 'master'

C:\homework>git merge newdaybranch
Updating 39ebcf5..46789d5
Fast-forward
 javaday.java | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 javaday.java
```



Exercise 5

Any time we have for this exercise, you are free to spend practicing whatever you find most interesting, or feel you have not fully grasped from the previous exercises and want another go through. Refer to the final section of the course for features you might like to explore.