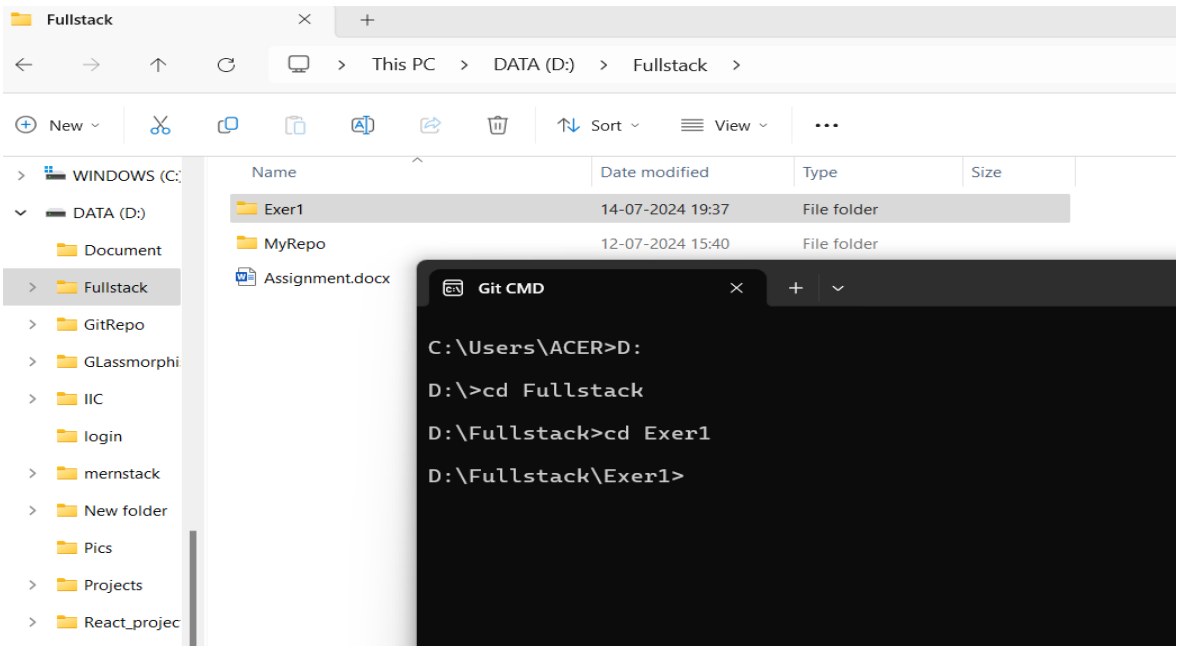
NAME : Harisha A

REG.NO : 73772121135

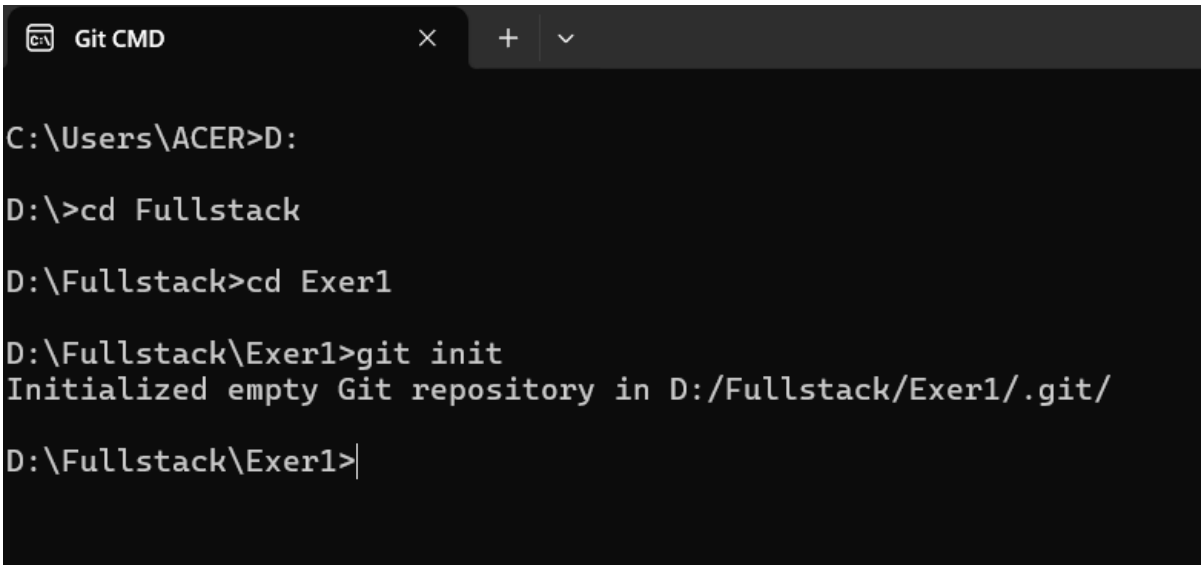
Exercise 1

Main Task

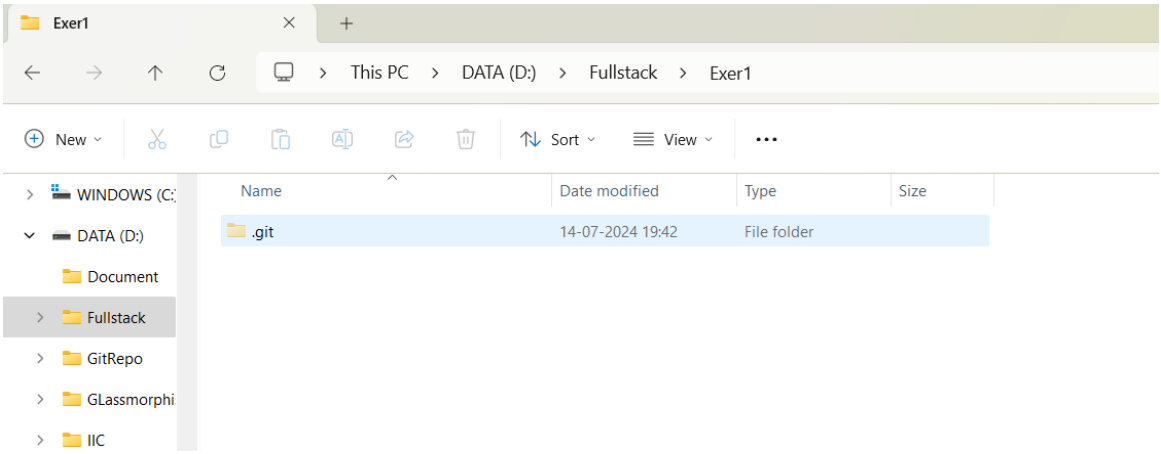
1. Create a new directory and change into it.



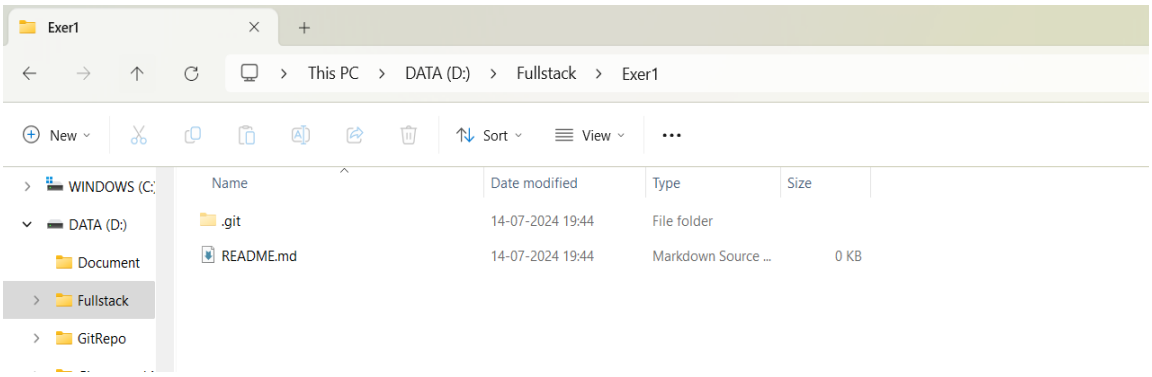
2. Use the init command to create a Git repository in that directory.



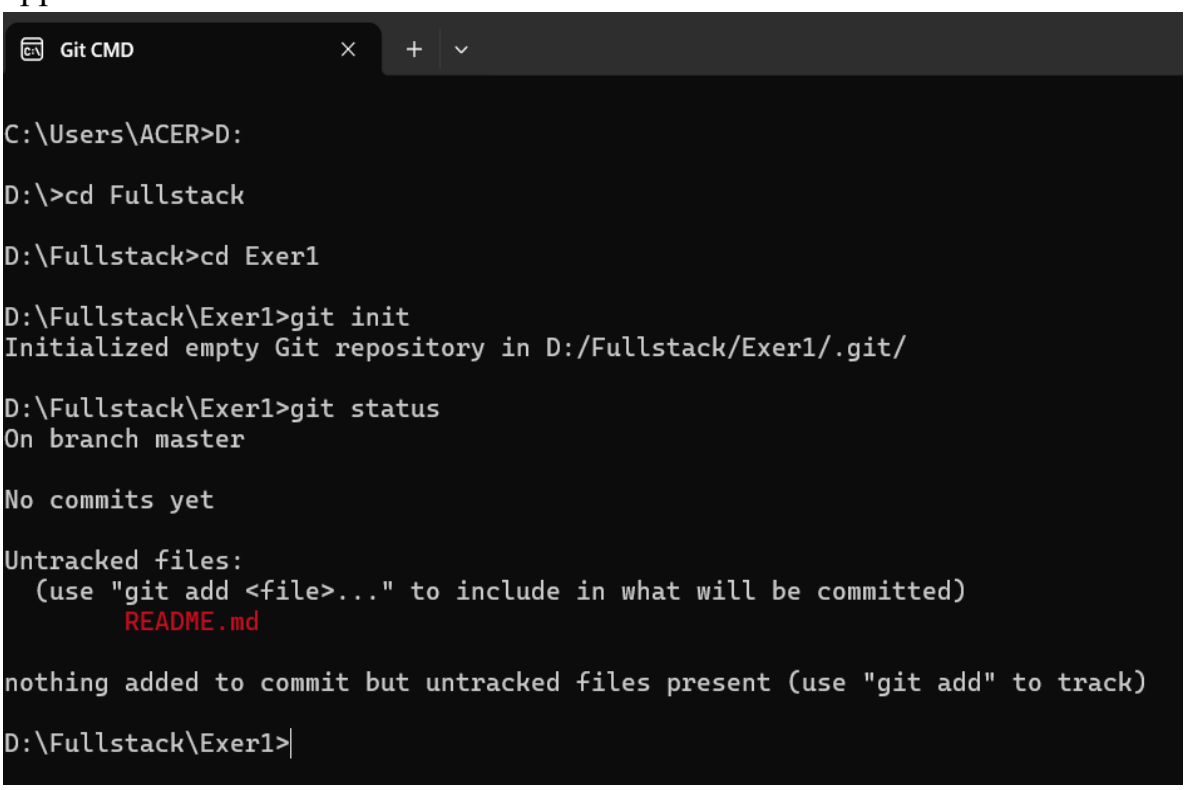
3. Observe that there is now a .git directory.



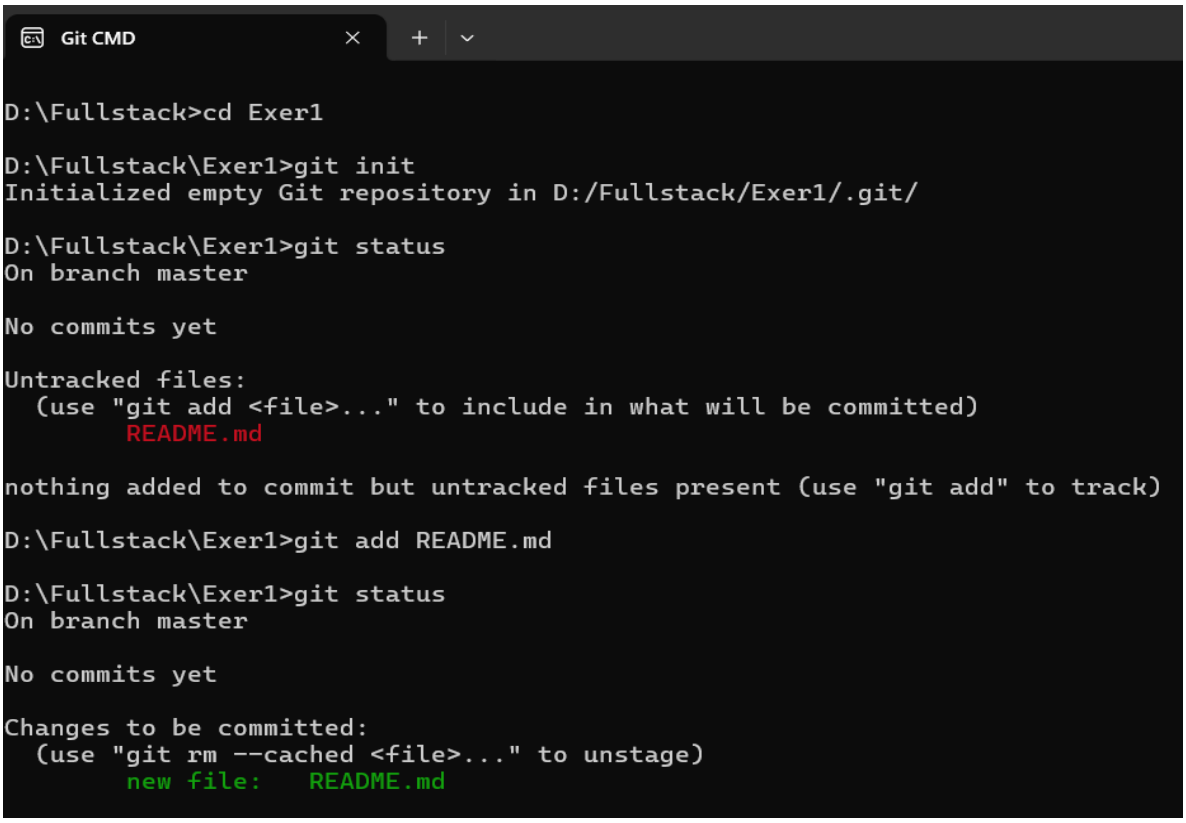
4. Create a README file



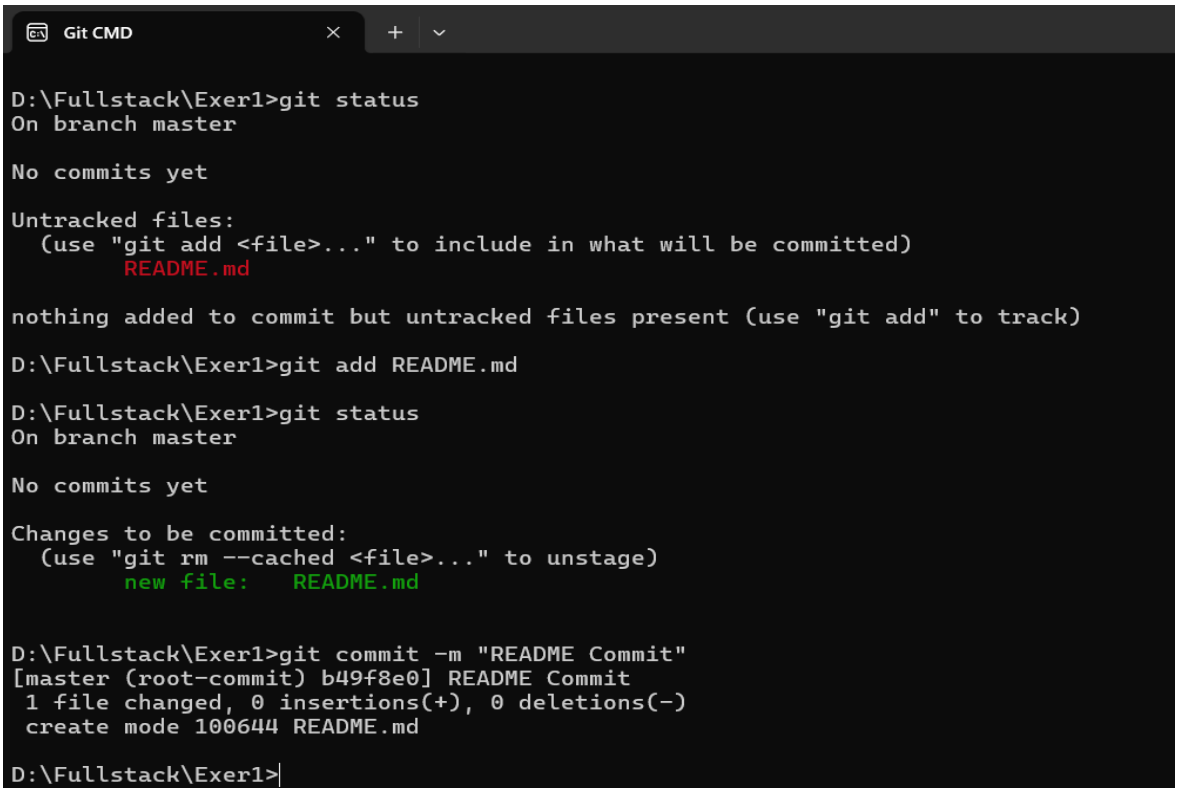
5. Look at the output of the status command; the README you created should appear as an untracked file.



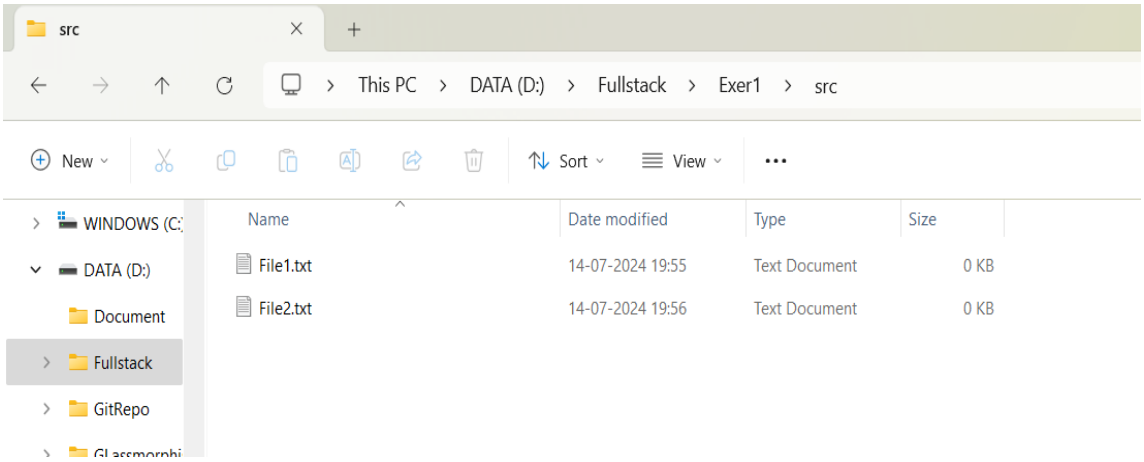
6. Use the add command to add the new file to the staging area. Again, look at the output of the status command.



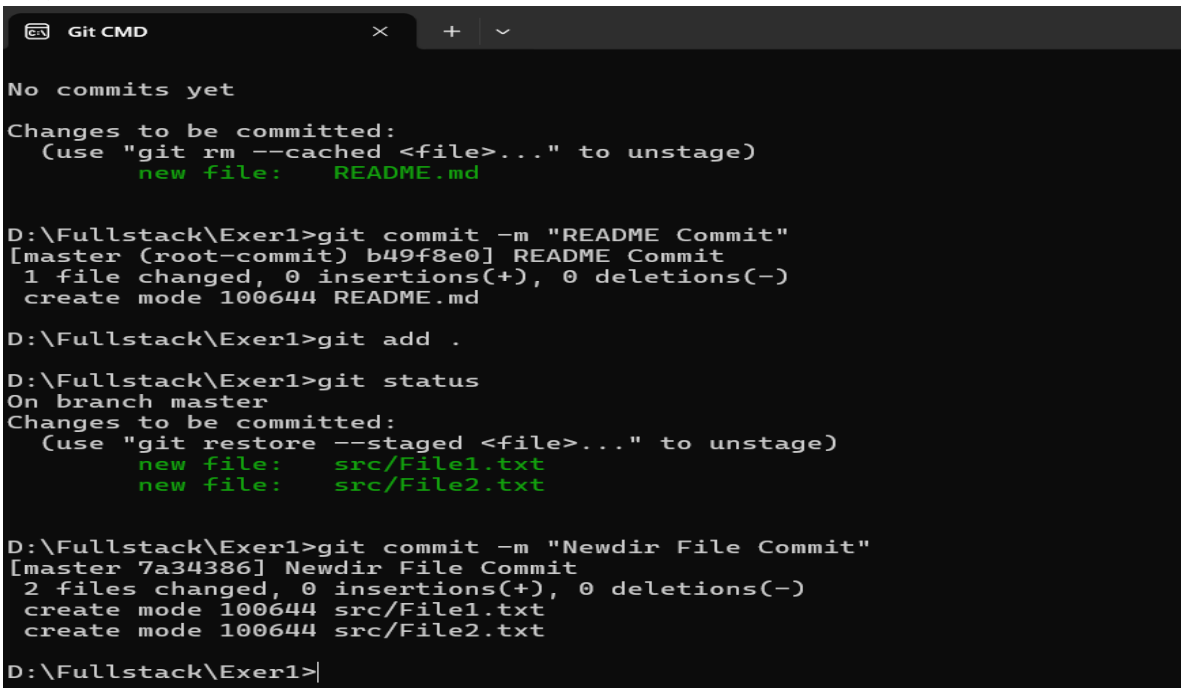
7. Now use the commit command to commit the contents of the staging area.

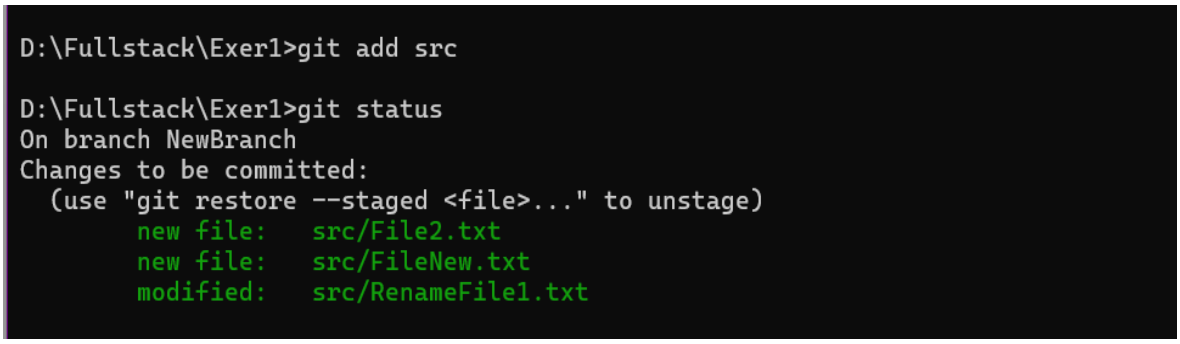


8. Create a src directory and add a couple of files to it.



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. (Git add src)





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

```
D:\Fullstack\Exer1>git diff 6cb1d10 c32824f
diff --git a/src/File1.txt b/src/File1.txt
index 321b6a5..970733e 100644
--- a/src/File1.txt
+++ b/src/File1.txt
@@ -1,2 +1,2 @@
 Hi i am Prathiksha
-and I am from Tirupur
+I am from Tirupur

D:\Fullstack\Exer1>
```

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?Git diff –staged) We have to commit and using the commit id or two files after the completion of commit to see the changes in staging area.

```
D:\Fullstack\Exer1>git add .

D:\Fullstack\Exer1>git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   src/File1.txt


D:\Fullstack\Exer1>git diff

D:\Fullstack\Exer1>
```

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.

```
D:\Fullstack\Exer1>git diff 6cb1d10 c32824f
diff --git a/src/File1.txt b/src/File1.txt
index 321b6a5..970733e 100644
--- a/src/File1.txt
+++ b/src/File1.txt
@@ -1,2 +1,2 @@
 Hi i am Prathiksha
-and I am from Tirupur
+I am from Tirupur

D:\Fullstack\Exer1>
```

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

```
Git CMD - "C:\Program Files\(

D:\Fullstack\Exer1>git log
commit c32824f49d761bcbff40a6c563ee2e942dc8796b (HEAD -> master)
Author: PrathikshaVS <prathiksha4915@gmail.com>
Date:   Sun Jul 14 20:10:16 2024 +0530

    Modify3 Commit

commit 6cb1d10b301c4b572d109d6793db4c909a87b9e8
Author: PrathikshaVS <prathiksha4915@gmail.com>
Date:   Sun Jul 14 20:02:12 2024 +0530

    Modify2 Commit

commit 2aff31790a214312e3b1e86af1a06a386ca09692
Author: PrathikshaVS <prathiksha4915@gmail.com>
Date:   Sun Jul 14 20:01:17 2024 +0530

    Modify1 Commit

commit 7a34386367f241ff7230e6fed74bcae741ede18f
Author: PrathikshaVS <prathiksha4915@gmail.com>
Date:   Sun Jul 14 19:59:16 2024 +0530

    Newdir File Commit

commit b49f8e00c8cd03c2dfe723736023bec2d2625708
Author: PrathikshaVS <prathiksha4915@gmail.com>
Date:   Sun Jul 14 19:50:25 2024 +0530

    README Commit
```

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?

```
D:\Fullstack\Exer1>git show
commit c32824f49d761bcbff40a6c563ee2e942dc8796b (HEAD -> master)
Author: PrathikshaVS <prathiksha4915@gmail.com>
Date:   Sun Jul 14 20:10:16 2024 +0530

    Modify3 Commit

diff --git a/src/File1.txt b/src/File1.txt
index 321b6a5..970733e 100644
--- a/src/File1.txt
+++ b/src/File1.txt
@@ -1,2 +1,2 @@
 Hi i am Prathiksha
-and I am from Tirupur
+I am from Tirupur

D:\Fullstack\Exer1>
```

The minimum characters are four but it has some risk so we have to give Atleast minimum7-8.

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

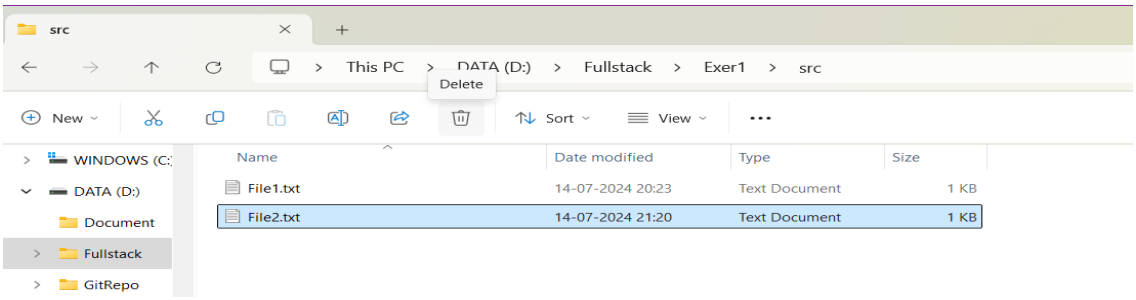
```
D:\Fullstack\Exer1>git add .

D:\Fullstack\Exer1>git commit -m "ADD Commit"
[master 97b48b5] ADD Commit
 2 files changed, 2 insertions(+)
 create mode 100644 src/File3.txt

D:\Fullstack\Exer1>git add .

D:\Fullstack\Exer1>git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   src/File2.txt


D:\Fullstack\Exer1>git commit -m "Couple Commit"
[master 47361be] Couple Commit
 1 file changed, 1 insertion(+)

D:\Fullstack\Exer1>
```

StretchTask

1. Use the Git rm command to remove a file. Look at the status afterwards. Now commit the deletion.

```
D:\Fullstack\Exer1\src>git rm File3.txt
rm 'src/File3.txt'

D:\Fullstack\Exer1\src>git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        deleted:    File3.txt


D:\Fullstack\Exer1\src>
```

2. Delete another file, but this time do not use Git to do it; e.g. if you are on Linux, just use the normal (non-Git) rm command; on Windows use del.



3. Look at the status. Compare it to the status output you had after using the Git built-in rm command. Is anything different? After this, commit the deletion.

```
D:\Fullstack\Exer1\src>git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        deleted:    File3.txt


D:\Fullstack\Exer1\src>git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        deleted:    File3.txt

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:    File2.txt
```

The changes were done in working area so the changes was untracked now before it was tracked

4. Use the Git mv command to move or rename a file; for example, rename README to README.txt. Look at the status. Commit the change.

```
D:\Fullstack\Exer1>git mv README.md README.txt

D:\Fullstack\Exer1>git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        renamed:    README.md -> README.txt
        deleted:    src/File3.txt

Changes not staged for commit:
```

5. Now do another rename, but this time using the operating system's command to do so. How does the status look? Will you get the right outcome if you were to commit at this point? (Answer: almost certainly not, so don't. □) Work out how to get the status to show that it will not lose the file, and then commit. Did Git at any point work out that you had done a rename?

```
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:    src/File1.txt
        deleted:    src/File2.txt

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        src/RenameFile1.txt
```

6. Use git help log to find out how to get Git to display just the most recent 3 commits. Try it.

```
D:\Fullstack\Exer1>git log -n 3
commit 807833ff7d91fcabf4aec81802e45d3e263596fa (HEAD -> master)
Author: PrathikshaVS <prathiksha4915@gmail.com>
Date:   Sun Jul 14 21:43:34 2024 +0530

     all commit

commit 47361be54a2326409c2ee3642526aebc0f208faf
Author: PrathikshaVS <prathiksha4915@gmail.com>
Date:   Sun Jul 14 21:22:54 2024 +0530

    Couple Commit

commit 97b48b5bdb0eab8f83b56dce94ba5607a95eaee5
Author: PrathikshaVS <prathiksha4915@gmail.com>
Date:   Sun Jul 14 21:19:55 2024 +0530

    ADD Commit

D:\Fullstack\Exer1>
```

7. If you don't remember, look back in the slides to see what the --stat option did on the diff command. Find out if this also works with the show command. How about the log command?

```
D:\Fullstack\Exer1>git show --stat 807833f
commit 807833ff7d91fcabf4aec81802e45d3e263596fa (HEAD -> master)
Author: PrathikshaVS <prathiksha4915@gmail.com>
Date:   Sun Jul 14 21:43:34 2024 +0530

     all commit

 README.md => README.txt | 0
 src/File3.txt           | 0
 2 files changed, 0 insertions(+), 0 deletions(-)

D:\Fullstack\Exer1>
```

8. Imagine you want to see a diff that summarizes all that happened between two commit identifiers. Use the diff command, specifying two commit identifiers joined by two dots (that is, something like abc123..def456). Check the output is what you expect.

```
D:\Fullstack\Exer1>git diff 2aff317..6cb1d10
diff --git a/src/File1.txt b/src/File1.txt
index 1110a39..321b6a5 100644
--- a/src/File1.txt
+++ b/src/File1.txt
@@ -1 +1,2 @@
 Hi i am Prathiksha
+and I am from Tirupur

D:\Fullstack\Exer1>
```

Exercise 2

Main Task

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

```
D:\Fullstack\Exer1>git status
On branch master
nothing to commit, working tree clean

D:\Fullstack\Exer1>
```

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

```
D:\Fullstack\Exer1>git status
On branch master
nothing to commit, working tree clean

D:\Fullstack\Exer1>git branch NewBranch

D:\Fullstack\Exer1>
```

3. Use the checkout command to switch to it.

```
D:\Fullstack\Exer1>git branch NewBranch

D:\Fullstack\Exer1>git checkout NewBranch
Switched to branch 'NewBranch'

D:\Fullstack\Exer1>
```

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

```
D:\Fullstack\Exer1>git commit -m "FirstCommit"
[NewBranch 31e099a] FirstCommit
 3 files changed, 2 insertions(+), 1 deletion(-)
 create mode 100644 src/File2.txt
 create mode 100644 src/FileNew.txt

D:\Fullstack\Exer1>git add .

D:\Fullstack\Exer1>git commit -m "SecondCommit"
[NewBranch 49a1feb] SecondCommit
 1 file changed, 1 insertion(+)
```

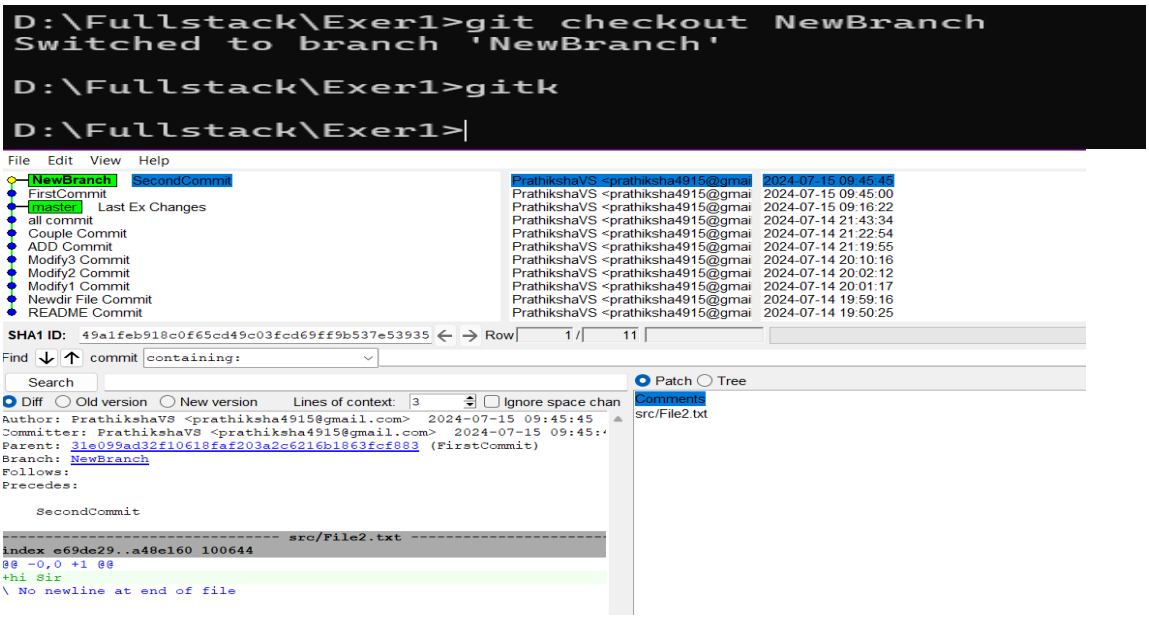
5. Use the log command to see the latest commits. The two you just made should be at the top of the list.

```
D:\Fullstack\Exer1>git log
commit 49a1feb918c0f65cd49c03fcd69ff9b537e53935 (HEAD -> NewBranch)
Author: PrathikshaVS <prathiksha4915@gmail.com>
Date:   Mon Jul 15 09:45:45 2024 +0530

    SecondCommit

commit 31e099ad32f10618faf203a2c6216b1863fcf883
Author: PrathikshaVS <prathiksha4915@gmail.com>
Date:   Mon Jul 15 09:45:00 2024 +0530

    FirstCommit
```

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.

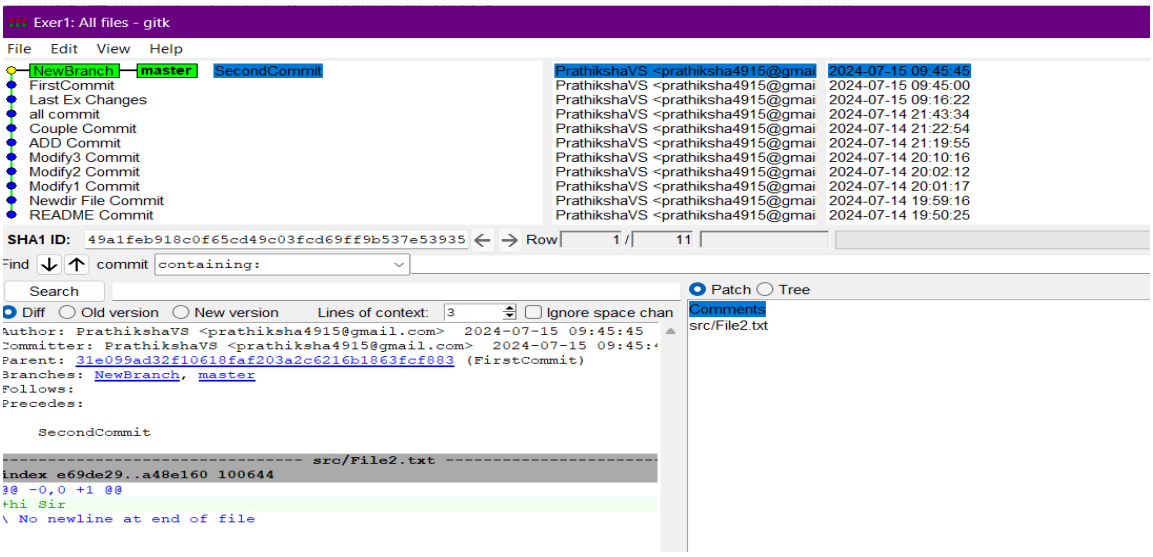
```
D:\Fullstack\Exer1>git checkout master
Switched to branch 'master'

D:\Fullstack\Exer1>git status
On branch master
nothing to commit, working tree clean

D:\Fullstack\Exer1>git log
commit b596444d78a6039e9acefcd43e51f89921fff443 (HEAD -> master)
Author: PrathikshaVS <prathiksha4915@gmail.com>
Date:   Mon Jul 15 09:16:22 2024 +0530

    Last Ex Changes

commit 807833ff7d91fcabf4aec81802e45d3e263596fa
Author: PrathikshaVS <prathiksha4915@gmail.com>
Date:   Sun Jul 14 21:43:34 2024 +0530

     all commit

commit 47361be54a2326409c2ee3642526aebc0f208faf
Author: PrathikshaVS <prathiksha4915@gmail.com>
Date:   Sun Jul 14 21:22:54 2024 +0530

    Couple Commit

commit 97b48b5bdb0eab8f83b56dce94ba5607a95eaee5
Author: PrathikshaVS <prathiksha4915@gmail.com>
Date:   Sun Jul 14 21:19:55 2024 +0530

    ADD Commit

commit c32824f49d761bcbff40a6c563ee2e942dc8796b
Author: PrathikshaVS <prathiksha4915@gmail.com>
Date:   Sun Jul 14 20:10:16 2024 +0530

    Modify3 Commit
```

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.

```
D:\Fullstack\Exer1>git checkout master
Switched to branch 'master'

D:\Fullstack\Exer1>git merge NewBranch
Updating b596444..49a1feb
Fast-forward
 src/File2.txt       | 1 +
 src/FileNew.txt     | 0
 src/RenameFile1.txt | 3 ++-
 3 files changed, 3 insertions(+), 1 deletion(-)
 create mode 100644 src/File2.txt
 create mode 100644 src/FileNew.txt

D:\Fullstack\Exer1>git log
commit 49a1feb918c0f65cd49c03fcd69ff9b537e53935 (HEAD -> master, NewBranch)
Author: PrathikshaVS <prathiksha4915@gmail.com>
Date:   Mon Jul 15 09:45:45 2024 +0530

    SecondCommit

commit 31e099ad32f10618faf203a2c6216b1863fcf883
Author: PrathikshaVS <prathiksha4915@gmail.com>
Date:   Mon Jul 15 09:45:00 2024 +0530

    FirstCommit

commit b596444d78a6039e9acefcd43e51f89921fff443
Author: PrathikshaVS <prathiksha4915@gmail.com>
Date:   Mon Jul 15 09:16:22 2024 +0530

    Last Ex Changes

commit 807833ff7d91fcabf4aec81802e45d3e263596fa
Author: PrathikshaVS <prathiksha4915@gmail.com>
Date:   Sun Jul 14 21:43:34 2024 +0530

     all commit
```



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

```
D:\Fullstack\Exer1>git checkout NewBranch
Switched to branch 'NewBranch'

D:\Fullstack\Exer1>git add .

D:\Fullstack\Exer1>git commit -m"commit1"
[NewBranch 8f7ddde] commit1
 1 file changed, 2 insertions(+), 1 deletion(-)

D:\Fullstack\Exer1>git add .

D:\Fullstack\Exer1>git commit -m"commit2"
[NewBranch 059fd7f] commit2
 1 file changed, 1 insertion(+)

D:\Fullstack\Exer1>
```

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.

```
D:\Fullstack\Exer1>git checkout master
Switched to branch 'master'

D:\Fullstack\Exer1>git commit -m"Commitin Master"
On branch master
nothing to commit, working tree clean
```

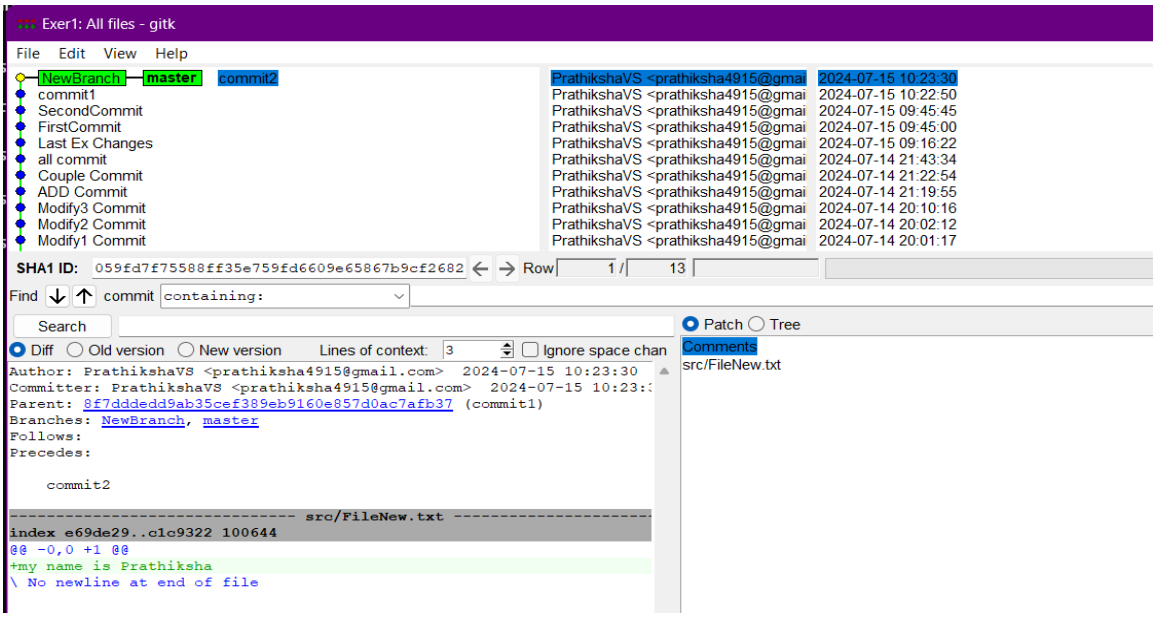
we cannot see the changes

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

```
D:\Fullstack\Exer1>git merge NewBranch
Updating 49a1feb..059fd7f
Fast-forward
 src/File2.txt   | 3 ++-
 src/FileNew.txt | 1 +
 2 files changed, 3 insertions(+), 1 deletion(-)

D:\Fullstack\Exer1>
```

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.

```
D:\Fullstack\Exer1>git log
commit 059fd7f75588ff35e759fd6609e65867b9cf2682 (HEAD -> master, NewBranch)
Author: PrathikshaVS <prathiksha4915@gmail.com>
Date:   Mon Jul 15 10:23:30 2024 +0530

    commit2

commit 8f7dddedd9ab35cef389eb9160e857d0ac7afb37
Author: PrathikshaVS <prathiksha4915@gmail.com>
Date:   Mon Jul 15 10:22:50 2024 +0530

    commit1

commit 49a1feb918c0f65cd49c03fcd69ff9b537e53935
Author: PrathikshaVS <prathiksha4915@gmail.com>
Date:   Mon Jul 15 09:45:45 2024 +0530

    SecondCommit

commit 31e099ad32f10618faf203a2c6216b1863fcf883
Author: PrathikshaVS <prathiksha4915@gmail.com>
Date:   Mon Jul 15 09:45:00 2024 +0530

    FirstCommit

commit b596444d78a6039e9acefcd43e51f89921fff443
Author: PrathikshaVS <prathiksha4915@gmail.com>
Date:   Mon Jul 15 09:16:22 2024 +0530

    Last Ex Changes
```



Stretch Task

1. Once again, checkout your branch. Make a couple of commits.

```
D:\Fullstack\Exer1>git add.
git: 'add.' is not a git command. See 'git --help'.

The most similar command is
        add

D:\Fullstack\Exer1>git add .

D:\Fullstack\Exer1>git commit -m "Couple Commit 1"
[NewBranch c88ccb0] Couple Commit 1
 1 file changed, 2 insertions(+), 1 deletion(-)

D:\Fullstack\Exer1>git add .

D:\Fullstack\Exer1>git commit -m "Couple Commit 2"
[NewBranch 79a4443] Couple Commit 2
 1 file changed, 0 insertions(+), 0 deletions(-)
 rename src/{File2.txt => FileNew2.txt} (100%)
```

2. Return to your master branch. Make a commit there that changes the exact same line, or lines, as commits in your branch did.

```
D:\Fullstack\Exer1>git add .

D:\Fullstack\Exer1>git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   src/FileNew.txt
        renamed:    src/File2.txt -> src/FileNew2.txt


D:\Fullstack\Exer1>git commit -m "Couple Commit in master"
[master 8b852ca] Couple Commit in master
 2 files changed, 2 insertions(+), 1 deletion(-)
 rename src/{File2.txt => FileNew2.txt} (100%)
```

3. Now try to merge your branch. You should get a conflict.

```
D:\Fullstack\Exer1\src>git add NewFile.txt

D:\Fullstack\Exer1\src>git commit -m "c1"
[NewBranch 5239930] c1
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 src/NewFile.txt

D:\Fullstack\Exer1\src>git checkout master
Switched to branch 'master'

D:\Fullstack\Exer1\src>git add NewFile.txt

D:\Fullstack\Exer1\src>git commit -m "c2"
[master 86b6ad3] c2
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 src/NewFile.txt

D:\Fullstack\Exer1\src>git merge NewBranch
Merge made by the 'ort' strategy.
```