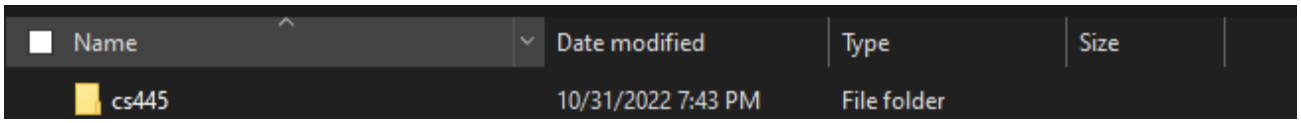
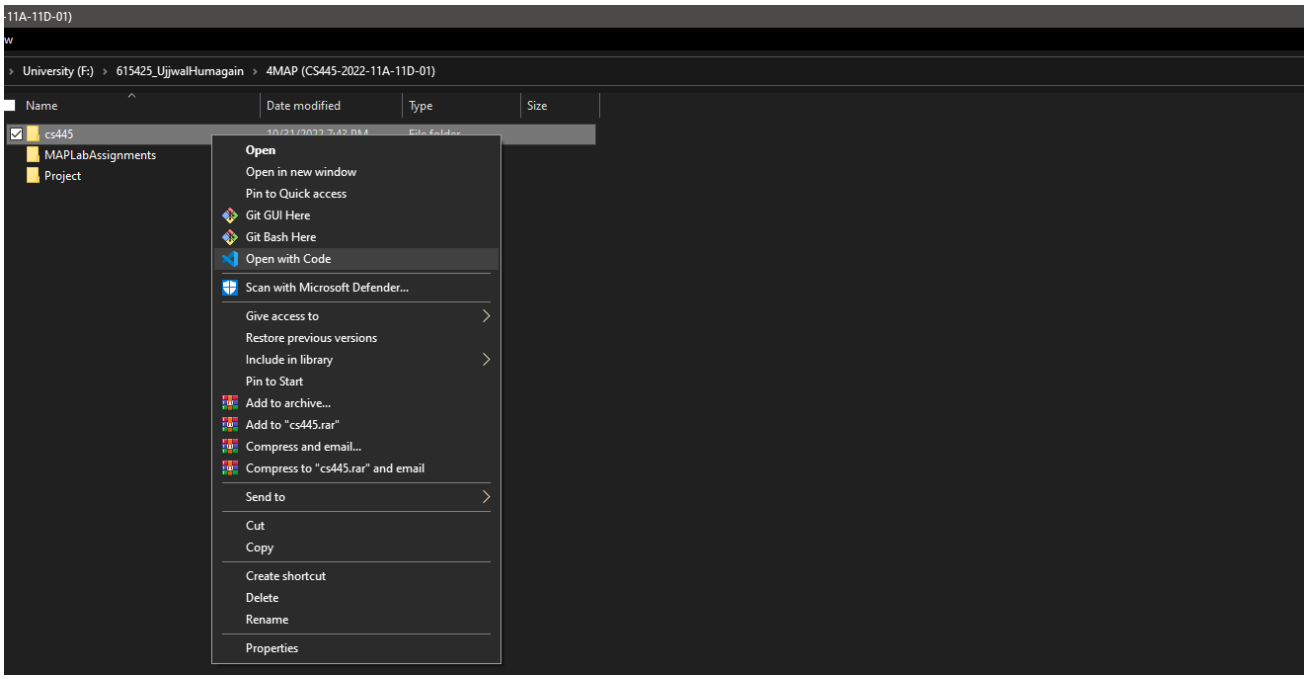
Practice with all the local repository commands:

Create a new folder:

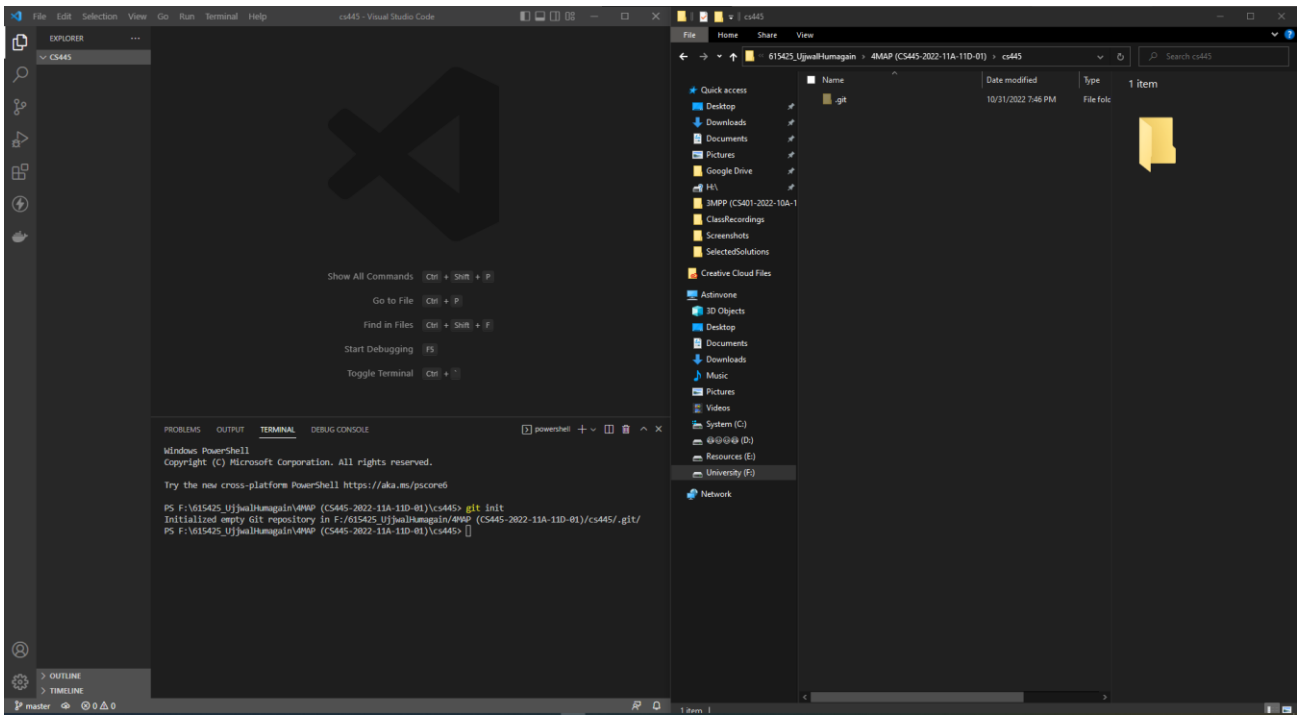


Open the folder in VS Code:



1. git init

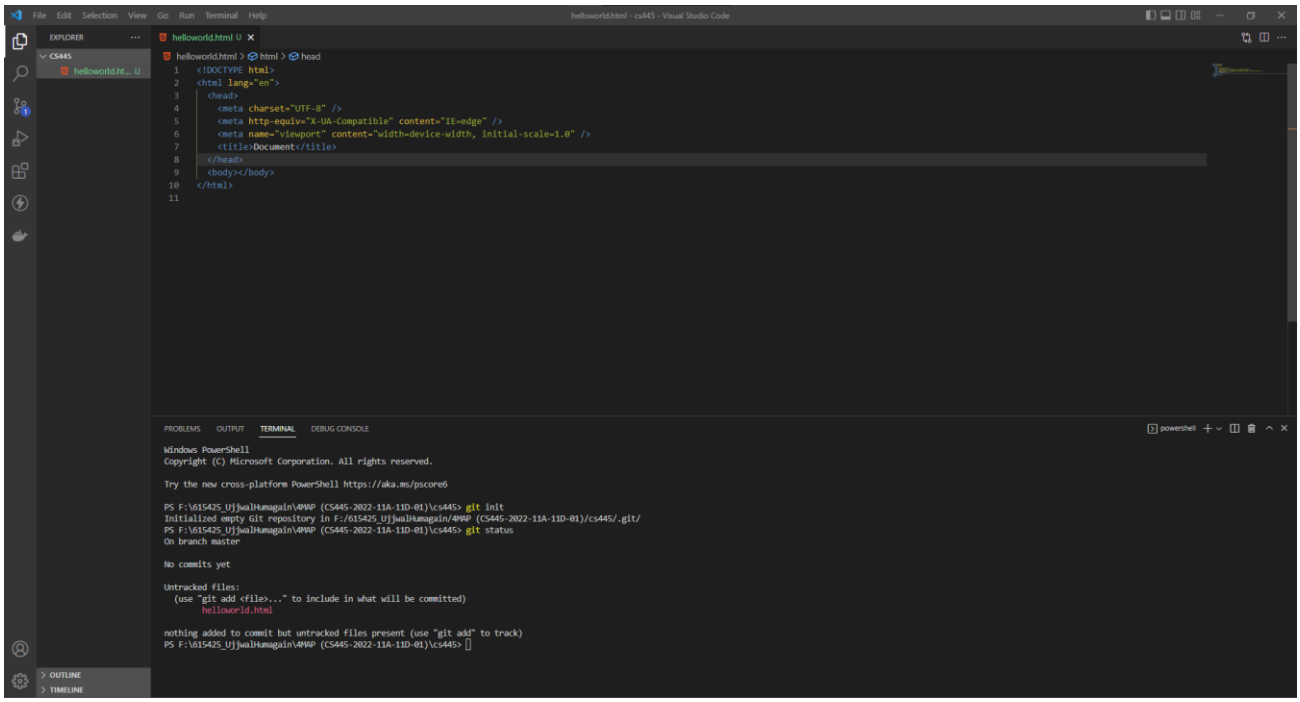
Initialize the folder as a git repository using ‘git init’ command



A new folder '.git' is created inside the root folder.

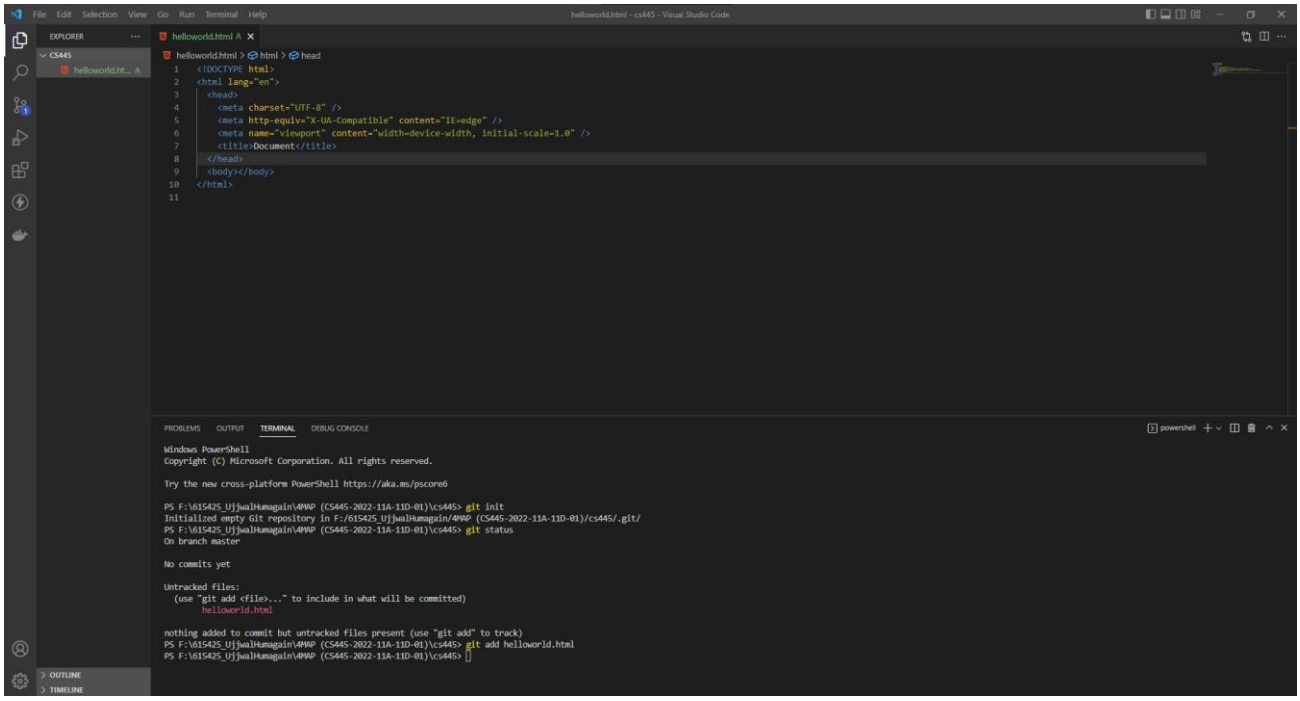
2. git status

Now let's add a new html file inside the working directory and run the 'git status' command so as to initiate the tracking of new files/folders.

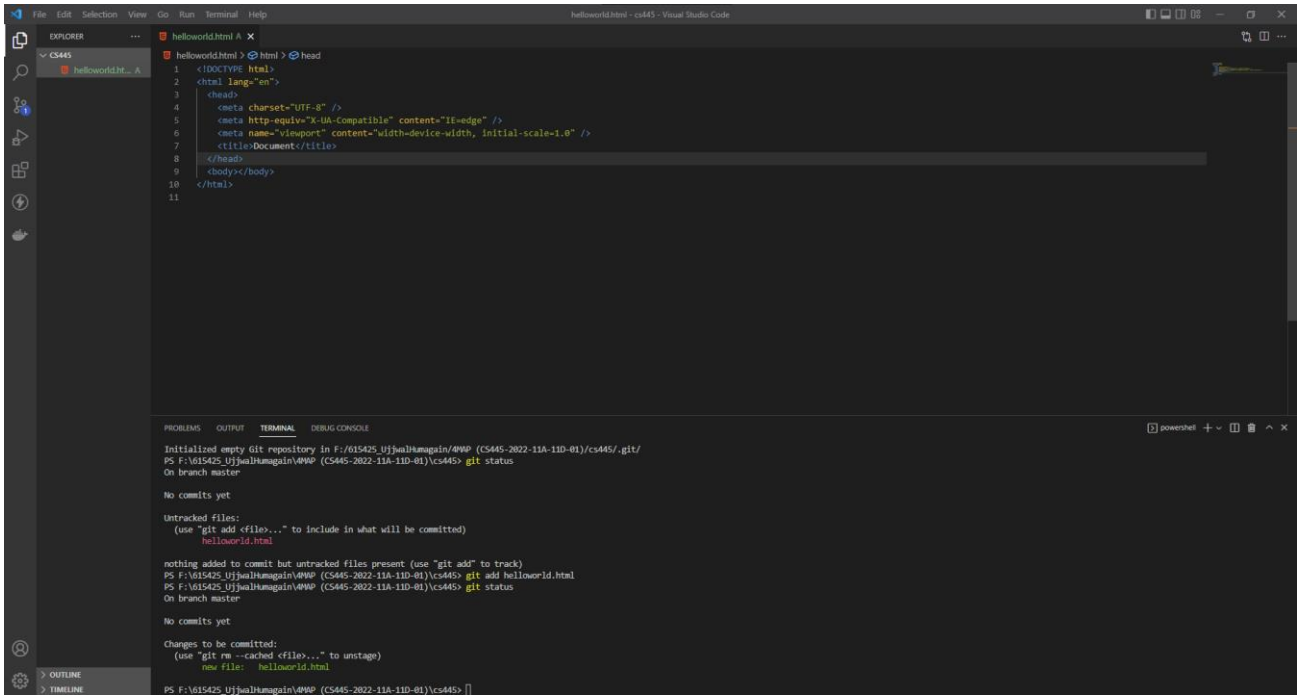


3. git add

Now in order to track the newly added file by adding it to the staging area we have to use 'git add *filename OR *'* (Here * is as wildcard) command as;

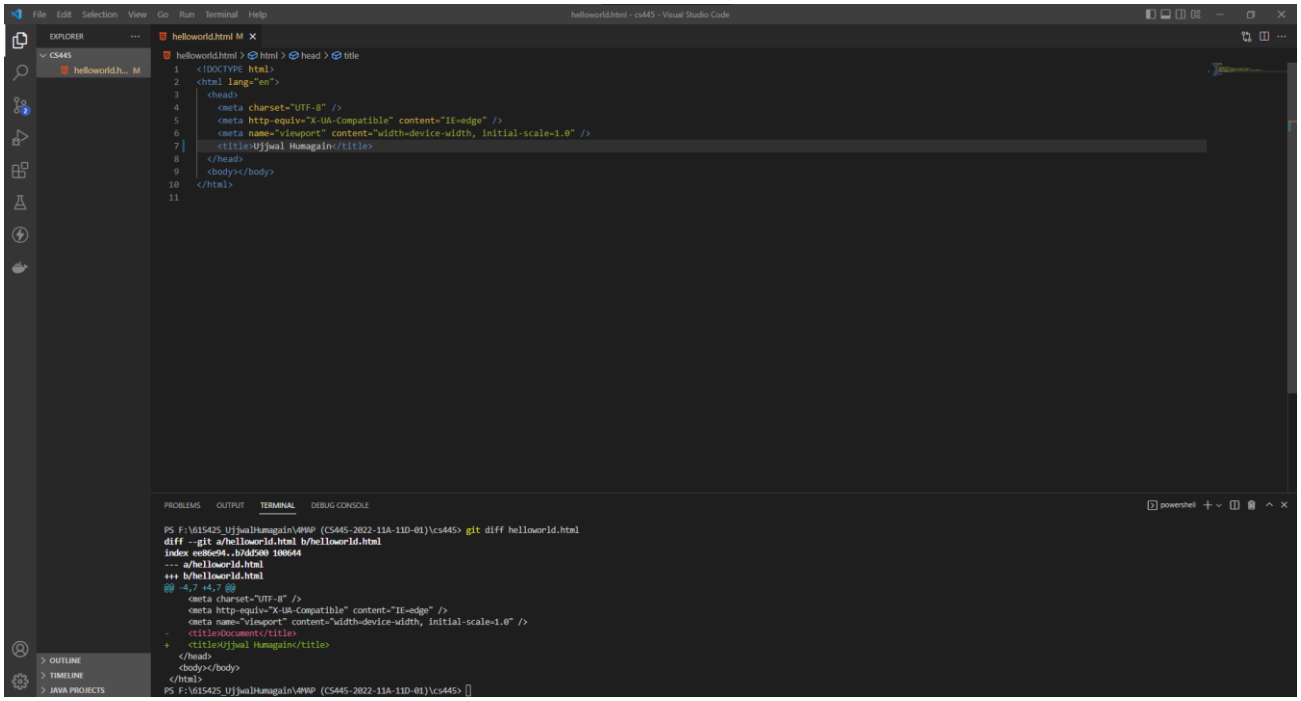


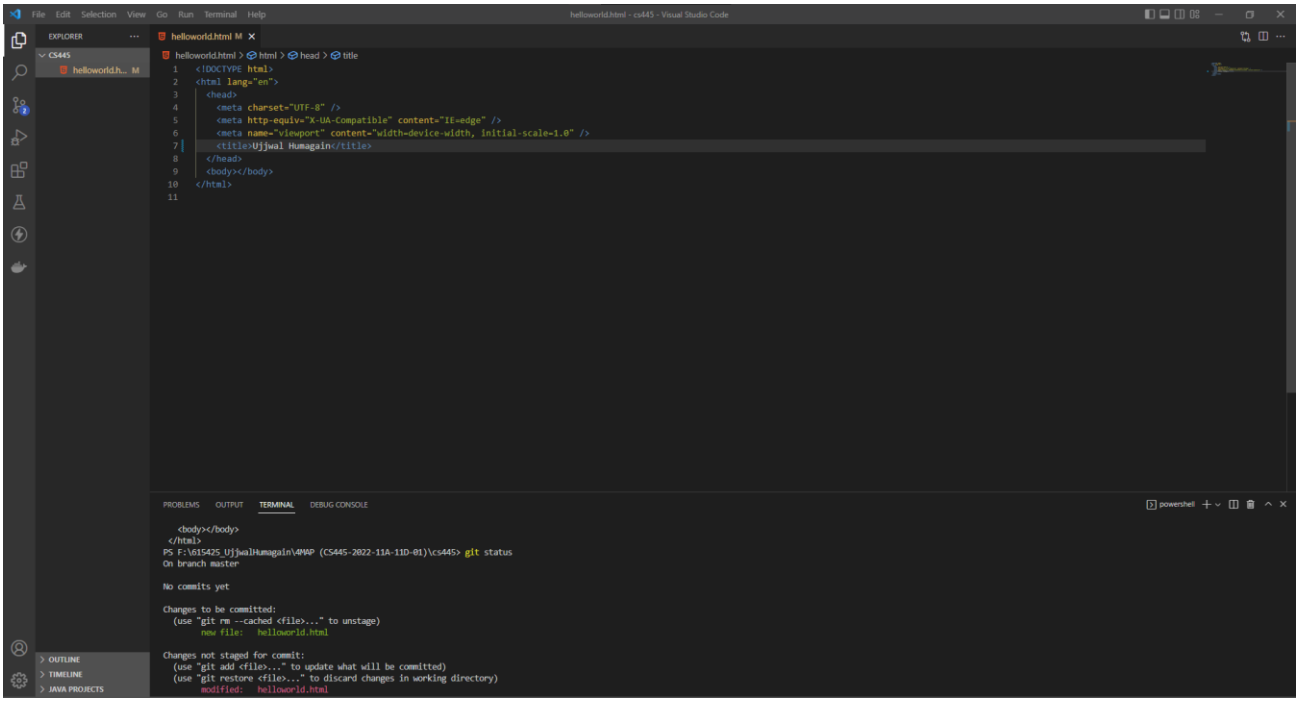
After this let us again see whether the file has been tracked or not using ‘git status’ command;



4. git diff

Since, we have the file in staging area, so any changes made to it is tracked and we have to add it again. In order to see the changes we have made to the file there is a command ‘git diff *filename*’ which shows what changes are made to the specific file.



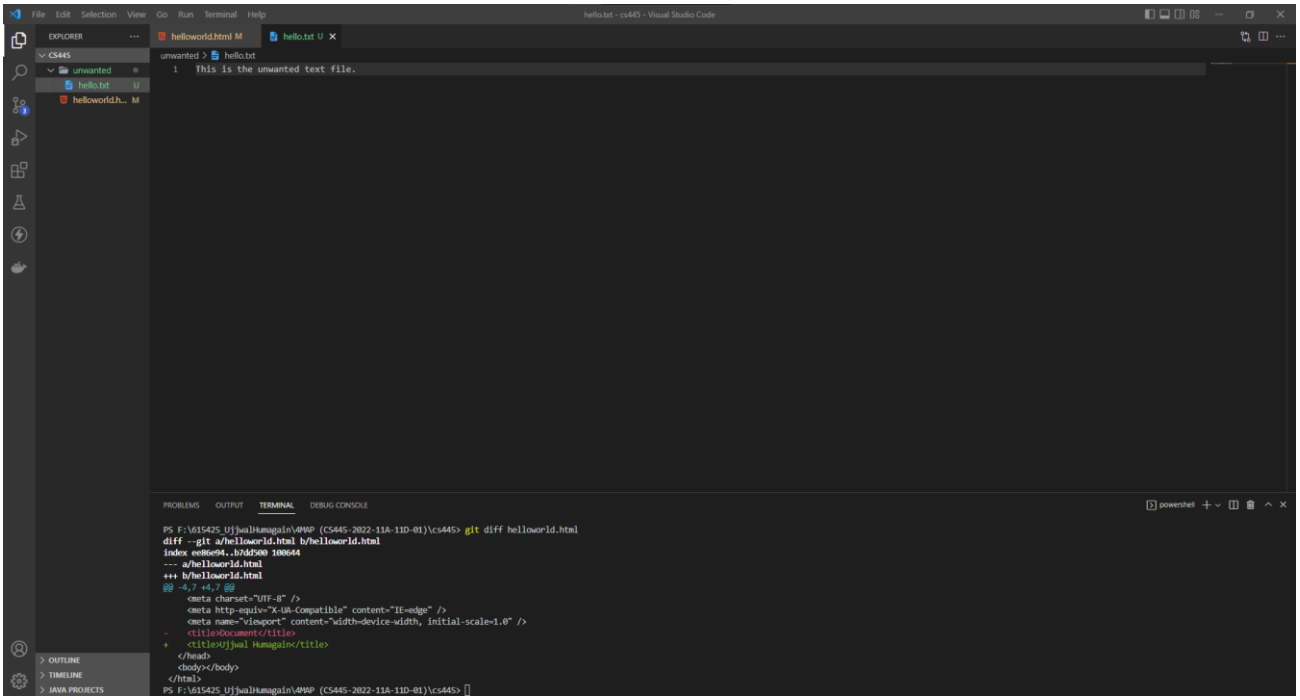


We have to add the file again to stage changes made to it.

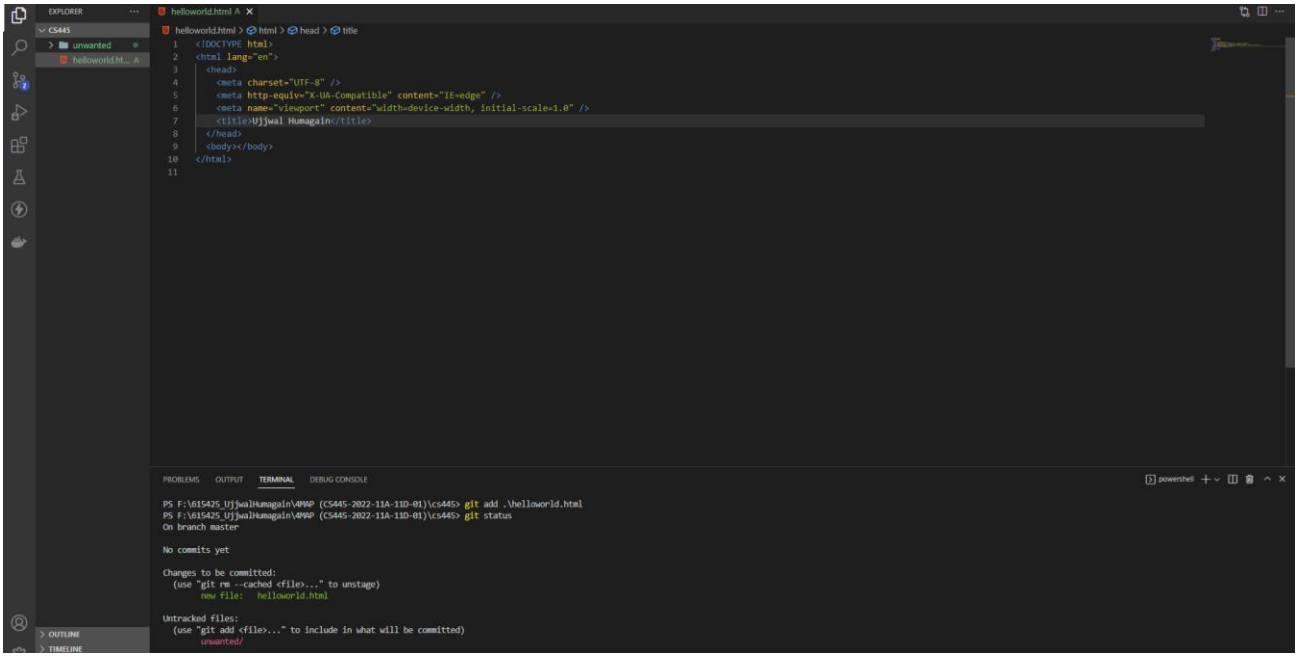
5. .gitignore

In most of the software projects there are different files and folders which are machine generated and build artifacts. So, in order to track the version of those projects by considering the development code only we have to ignore the generated files and folders. In order to do so, we have to create a new file '.gitignore' inside the working directory and add path to the respective files and folders which are to be ignored and discarded from being tracked during the versioning of the project.

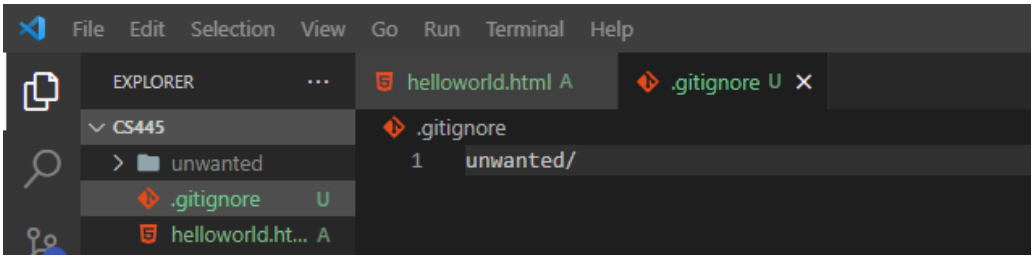
To demonstrate the working of .gitignore at first let us create a new folder and add a file inside it as;



Since we created a new folder and file which are untracked at first as;



But we want to ignore the folder and files inside it so for this at first we have to add .gitignore in the working directory and add the path to the folder in it as;



Again, let us do the 'git status'. This time the .gitignore file is untracked instead of the folder that we created previously.

```
PS F:\615425_UjjwalHumagain\4MAP (CS445-2022-11A-11D-01)\cs445> git status
On branch master

No commits yet

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

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

PS F:\615425_UjjwalHumagain\4MAP (CS445-2022-11A-11D-01)\cs445>
```

Now, let us add the .gitignore file and move everything in the staging area as;

```
PS F:\615425_UjjwalHumagain\4MAP (CS445-2022-11A-11D-01)\cs445> git add .gitignore
PS F:\615425_UjjwalHumagain\4MAP (CS445-2022-11A-11D-01)\cs445> git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   .gitignore
        new file:   helloworld.html
```

6. git commit

Since our main intention is to create a local repository and version the working directory so we have to create checkpoints for tracking every change made inside it. For this we have to use “git commit -m ‘commit message’ ” command. This saves the everything presented in the staging area to the repository. The commit message attached in this command explains what changes has been done in a single unit of codebase.

```
PS F:\615425_UjjwalHumagain\4MAP (CS445-2022-11A-11D-01)\cs445> git commit -m "Initial Commit"
[master (root-commit) 39ee077] Initial Commit
 2 files changed, 11 insertions(+)
 create mode 100644 .gitignore
 create mode 100644 helloworld.html
PS F:\615425_UjjwalHumagain\4MAP (CS445-2022-11A-11D-01)\cs445>
```

7. git log

In every project there are more than one commits so to see all the commits there is a command as ‘git log’ that displays every commit made in that repository

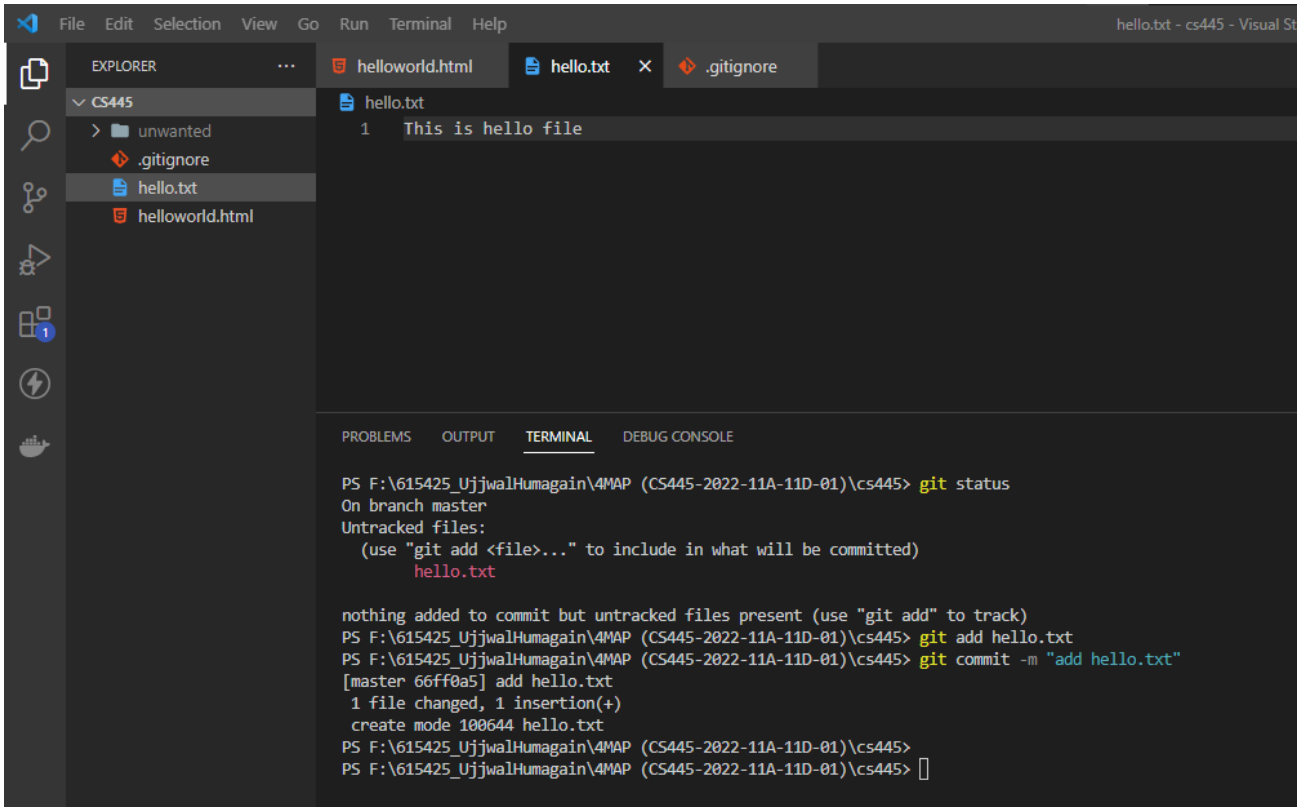
```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE

PS F:\615425_UjjwalHumagain\4MAP (CS445-2022-11A-11D-01)\cs445> git commit -m "Initial Commit"
[master (root-commit) 39ee077] Initial Commit
 2 files changed, 11 insertions(+)
 create mode 100644 .gitignore
 create mode 100644 helloworld.html
PS F:\615425_UjjwalHumagain\4MAP (CS445-2022-11A-11D-01)\cs445> git log
commit 39ee077704e1b486708384b28d7fc8decb1610ed (HEAD -> master)
Author: deStarxis <humagainuzjwol@gmail.com>
Date:   Mon Oct 31 20:26:29 2022 -0500

    Initial Commit
PS F:\615425_UjjwalHumagain\4MAP (CS445-2022-11A-11D-01)\cs445>
```

We can also format the git log using other optional command attributes as;

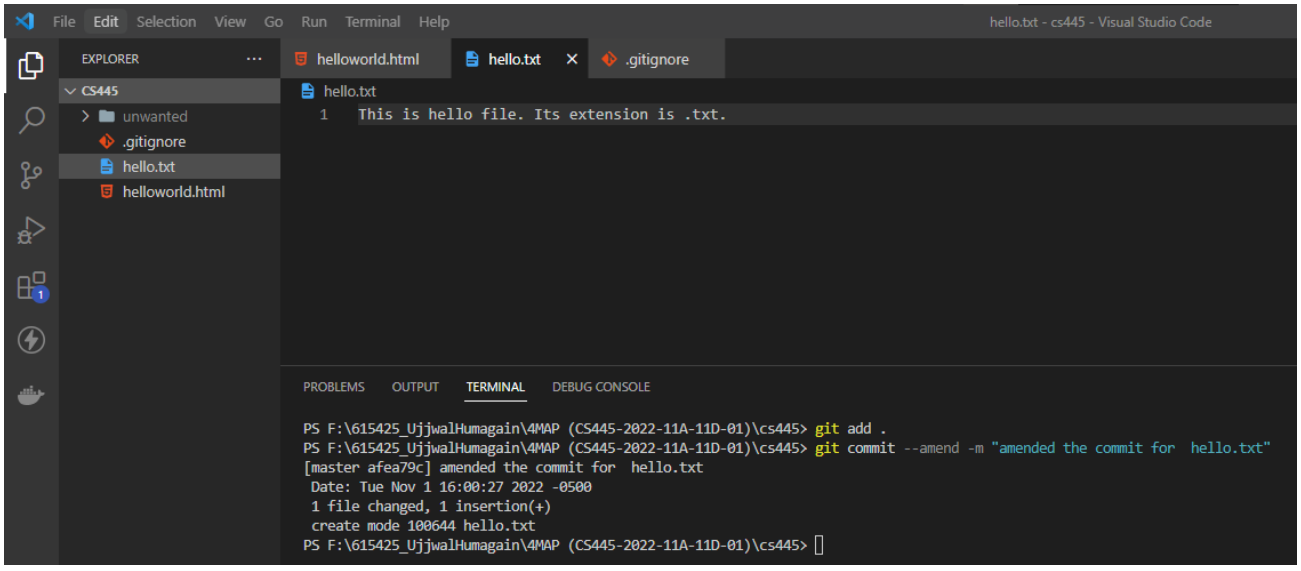
```
PS F:\615425_UjjwalHumagain\4MAP (CS445-2022-11A-11D-01)\cs445> git log
commit 39ee077704e1b486708384b28d7fc8decb1610ed (HEAD -> master)
Author: deStarxis <humagainuzjwol@gmail.com>
Date:   Mon Oct 31 20:26:29 2022 -0500

    Initial Commit
PS F:\615425_UjjwalHumagain\4MAP (CS445-2022-11A-11D-01)\cs445> git log --oneline
39ee077 (HEAD -> master) Initial Commit
PS F:\615425_UjjwalHumagain\4MAP (CS445-2022-11A-11D-01)\cs445> git log --all --pretty=format:"% h %cd %s (%an)" --since="7 days ago"
 39ee077 Mon Oct 31 20:26:29 2022 -0500 Initial Commit (deStarxis)
PS F:\615425_UjjwalHumagain\4MAP (CS445-2022-11A-11D-01)\cs445>
```

8. git commit -amend

We can also update the previous commit by using git commit -amend command so that the changes done multiple times can be added under same commit. For the demonstration of this, at first let's add one commit and amend another commit to it as;

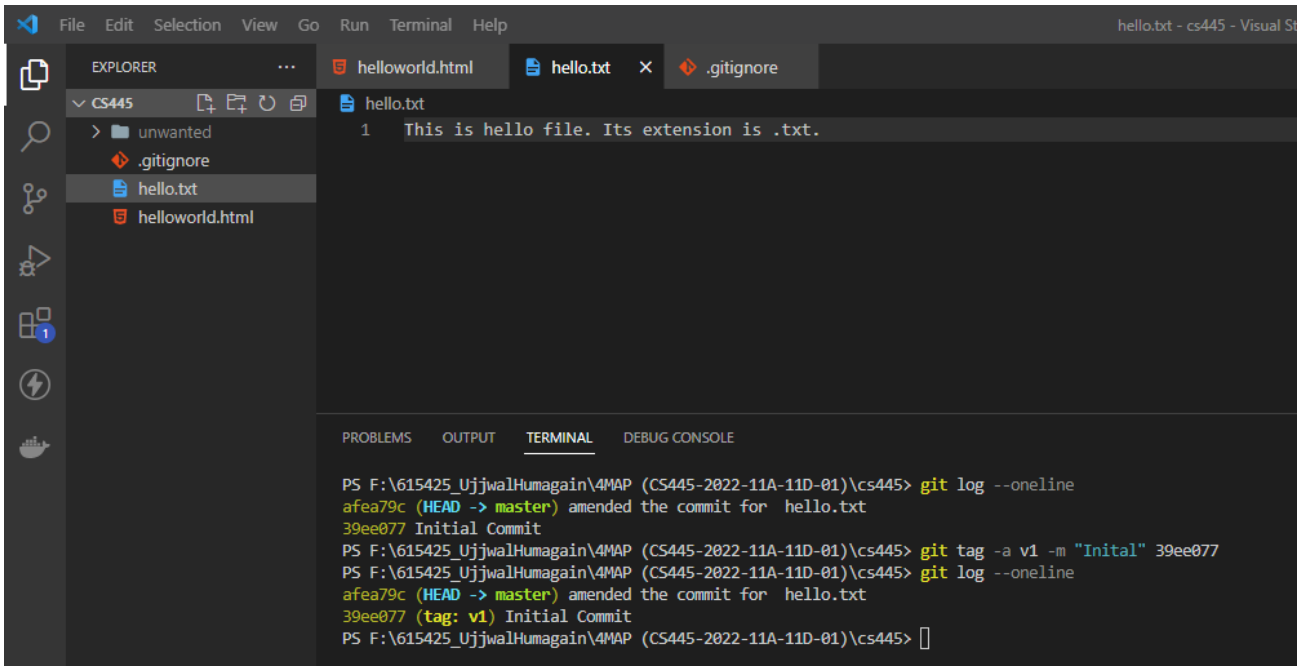


Again, the hello.txt file will be modified once, and it will be again added to the repository. But this time instead of creating a new commit, the changes will be amended to the previous commit as;



9. git tag

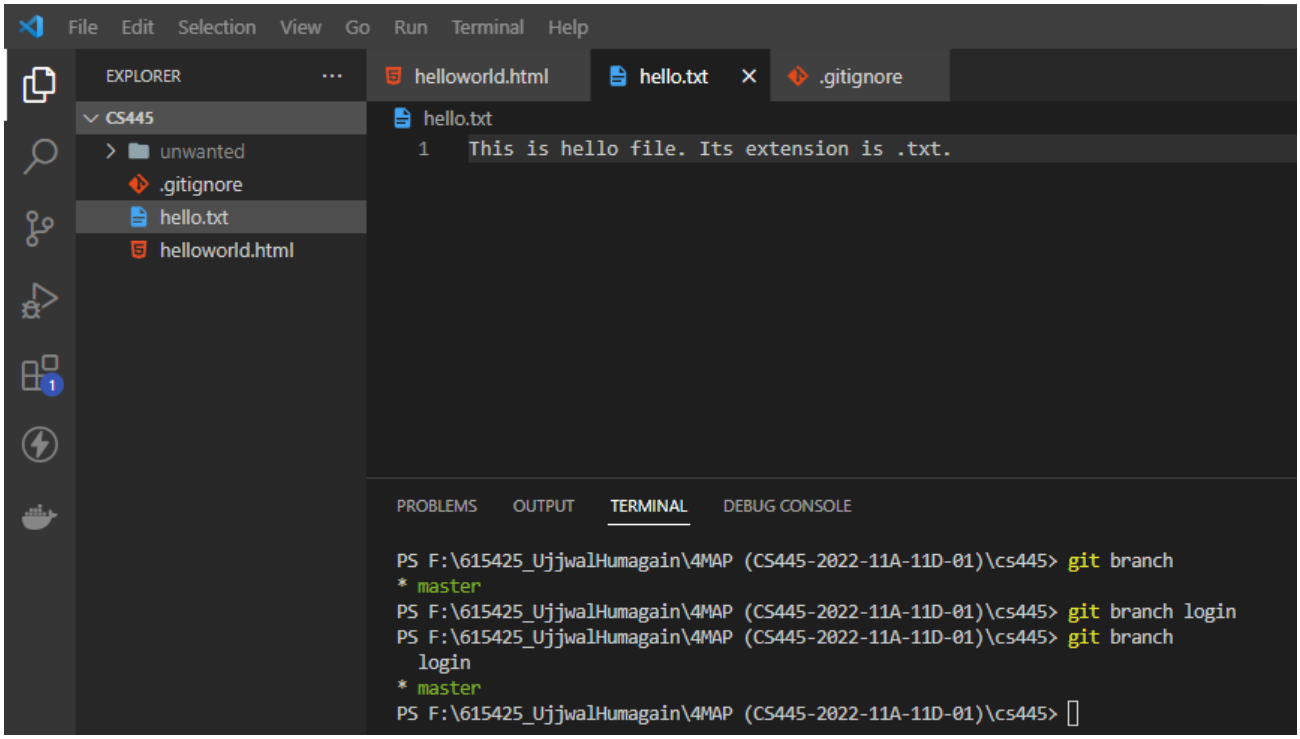
We can also add marker to each of the commits so as to recognize them easily while performing checkout and reset. So, let us add tag to the first commit that we created in the repository as;



Now with the help of this tag we can easily perform operations on that commit.

10. git branch

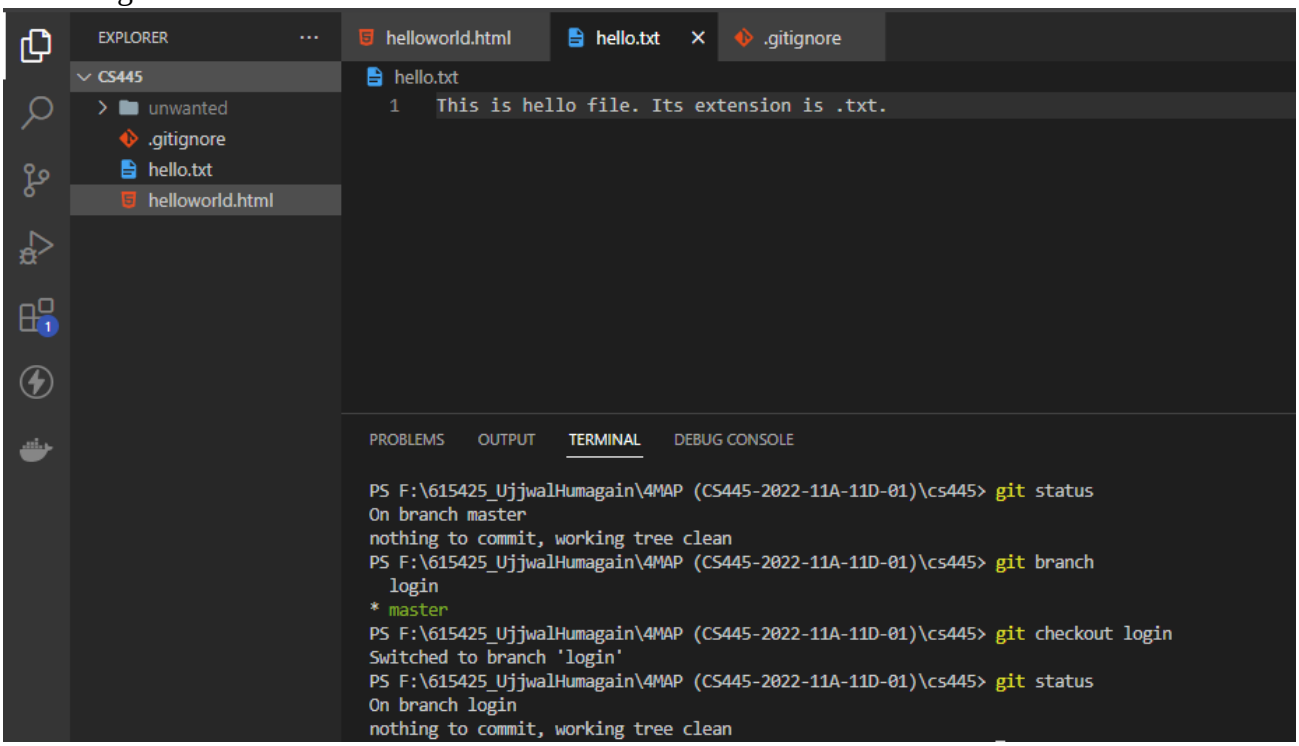
In repository a default branch is present initially, but we can have more based on the requirements and use case. Using ‘git branch’ command we can list all the branches present in the repository. We can also create a new branch using ‘git branch <branchname>’ command as;



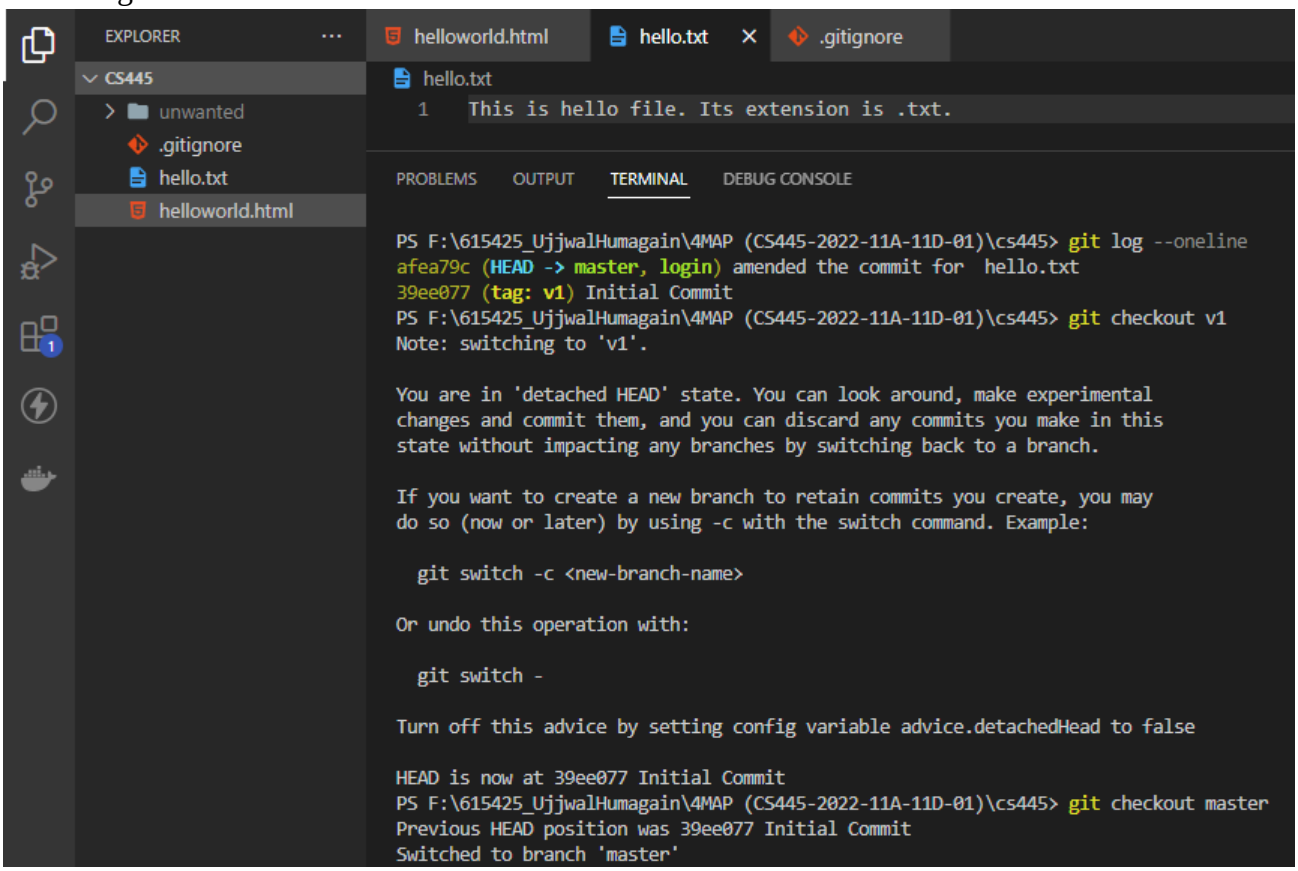
11. git checkout

We can also switch from one branch to another and from one commit to another using ‘git checkout’ command. Here, instead of checkout we can also use switch.

Switching branch:

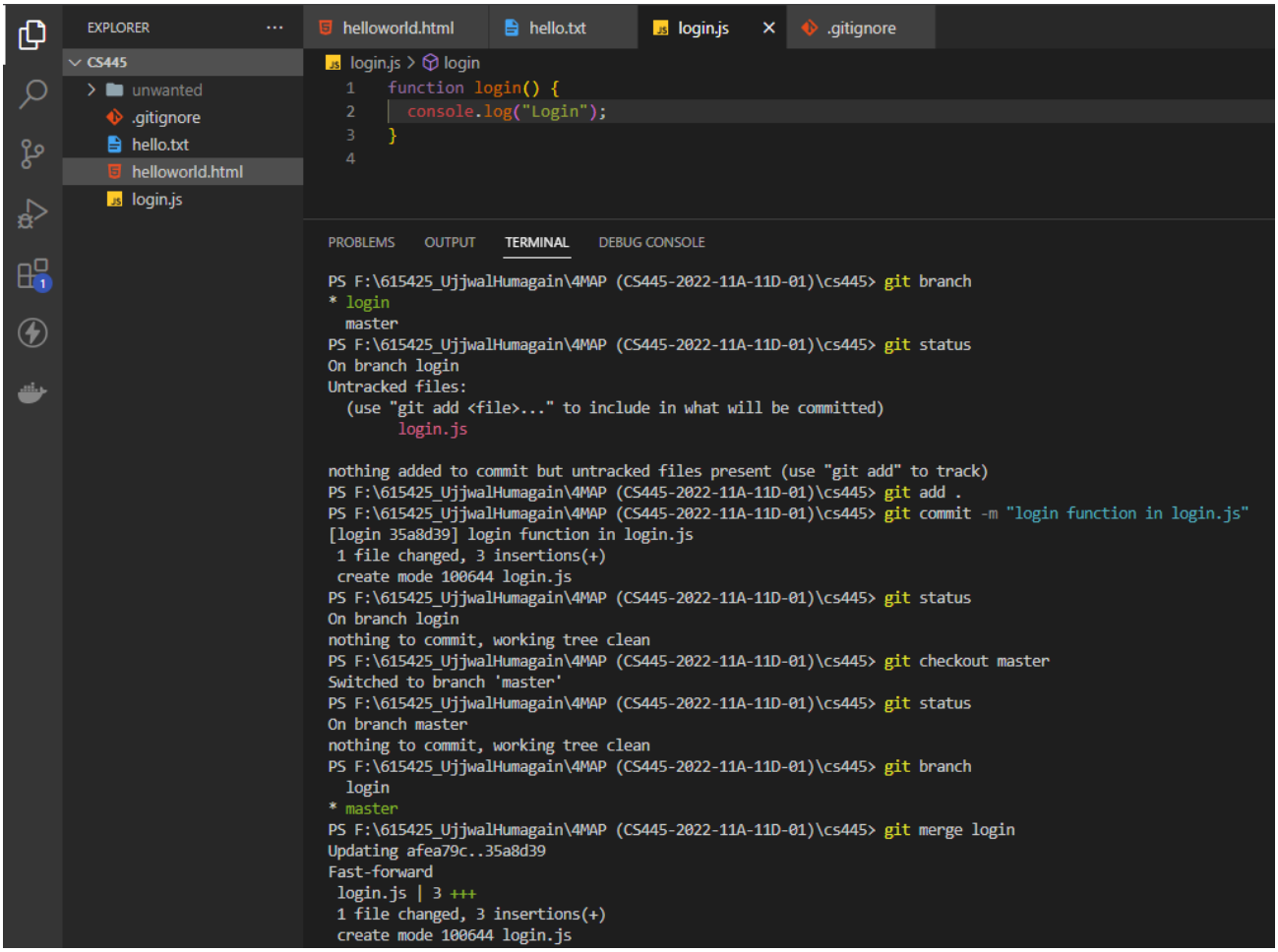


Switching commit:



12. git merge

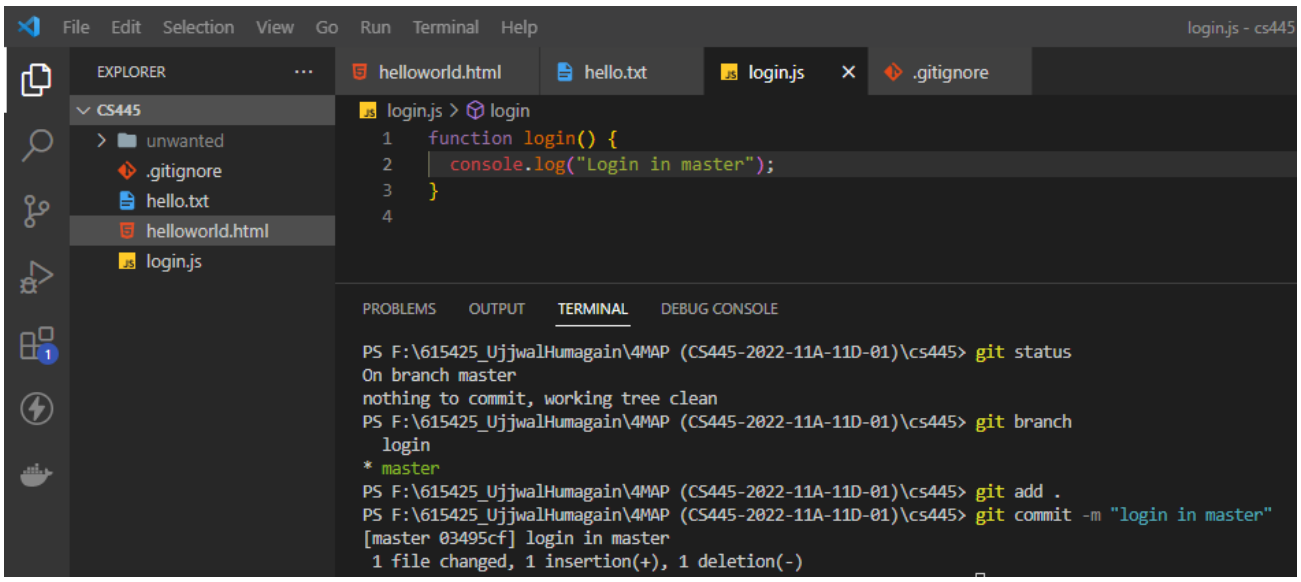
This command is used to merge two branches. For example, we have two branches master and login. Here, in the login branch we added the code related to the login feature and committed it. But later we need that code in the master branch as well. For doing so at first we need to checkout to the master branch and add ‘git merge login’ command to merge the login branch in it as;



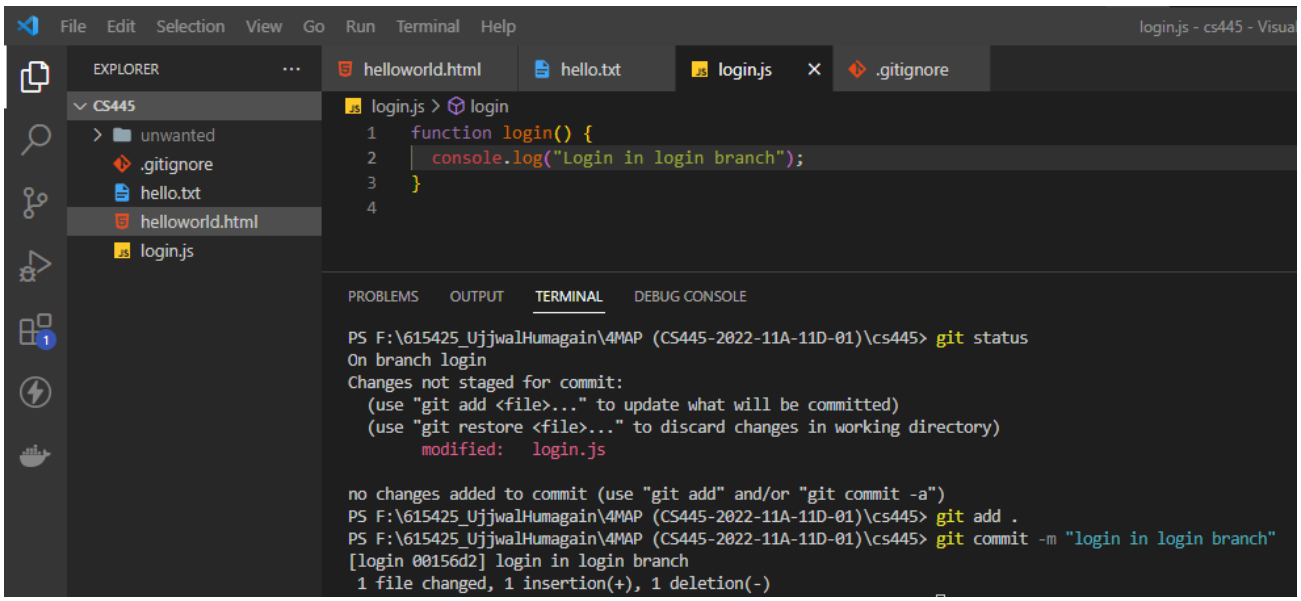
i. Create a conflict and solve the conflict

Sometimes during the merging of two branches conflict might occur because of the simultaneous changes done to the same file form different branches. In that case we need to fix the merging conflict manually by deciding the changes to keep and then commit the new changes and solve the conflict. Demonstration of this scenario is presented as;

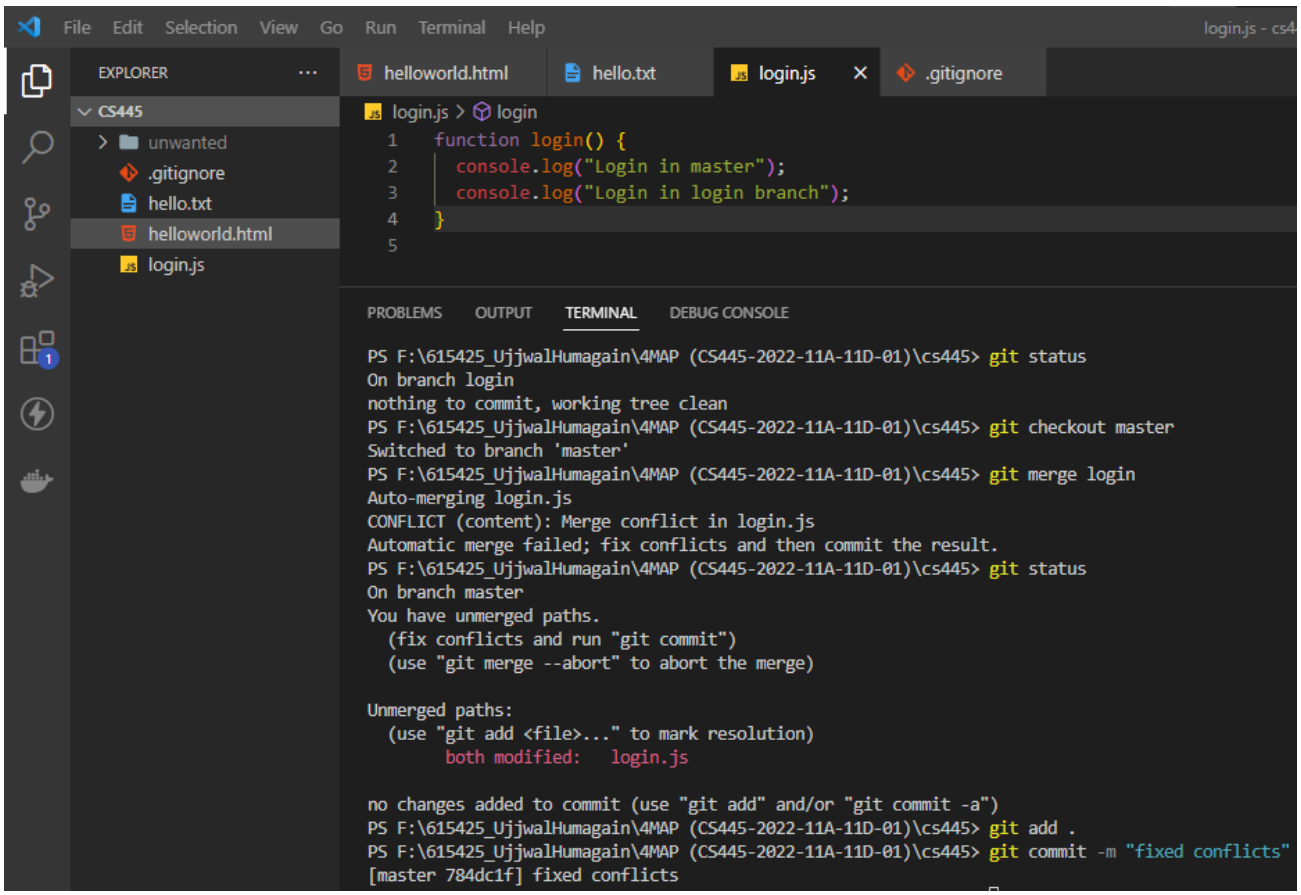
a. Modified login.js in master branch



b. Modified login.js in login branch

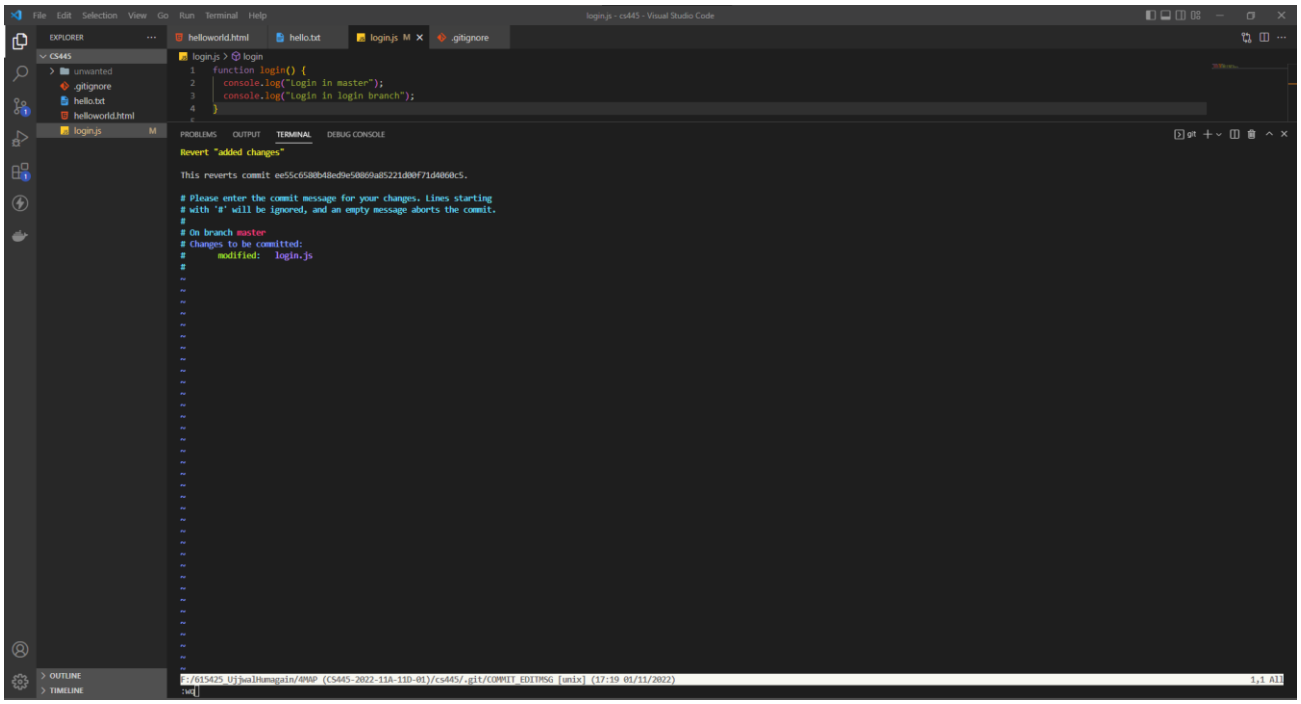


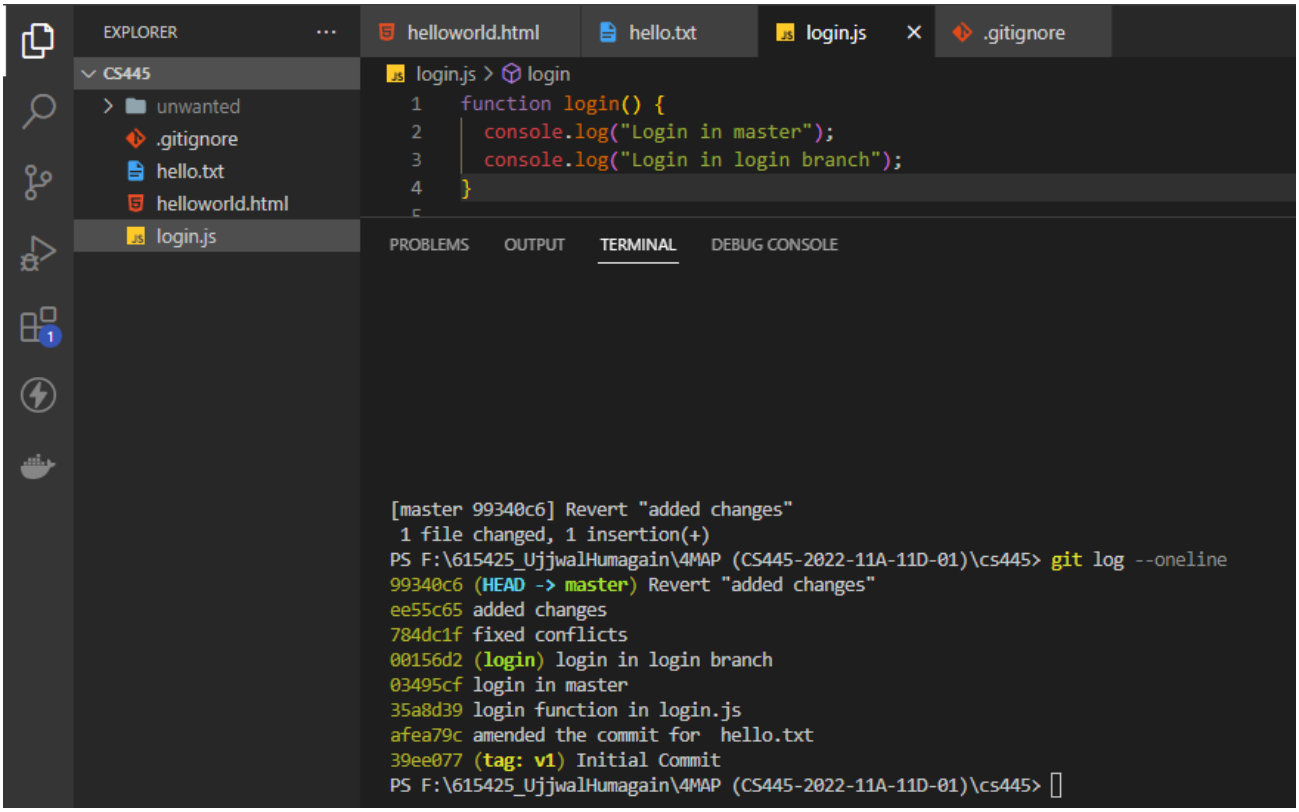
c. Merging the login branch in master branch by fixing issue



13. git revert

We can also revert the changes that have been committed to a local repository. For this we need to specify the HEAD or the specific commit hash and the commit will be reverted as;





Reverting will create a new commit and contains only those changes which were present before the reverted commit.

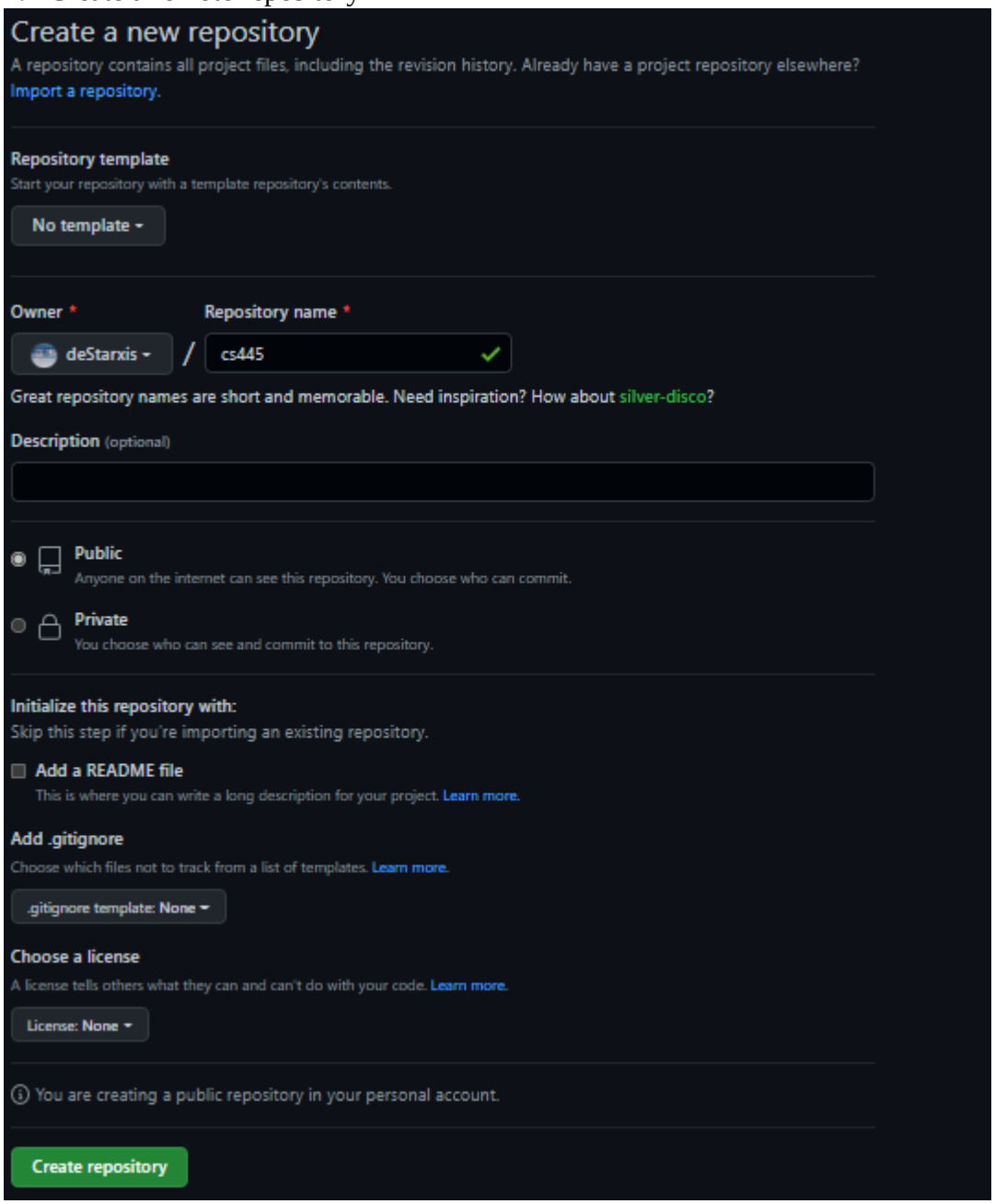
14. git reset

Similar to that of the revert we can also reset the commit history of the repository to the desired one by specifying the specific commit via ‘git reset’ command. Instead of creating a new commit like that of the ‘git revert’, the command resets the current branch to the specified commit and deletes the newer commits after that.

```
PS F:\615425_UjjwalHumagain\4MAP (CS445-2022-11A-11D-01)\cs445> git log --oneline
99340c6 (HEAD -> master) Revert "added changes"
ee55c65 added changes
784dc1f fixed conflicts
00156d2 (login) login in login branch
03495cf login in master
35a8d39 login function in login.js
afea79c amended the commit for  hello.txt
39ee077 (tag: v1) Initial Commit
PS F:\615425_UjjwalHumagain\4MAP (CS445-2022-11A-11D-01)\cs445> git reset --hard ee55c65
HEAD is now at ee55c65 added changes
PS F:\615425_UjjwalHumagain\4MAP (CS445-2022-11A-11D-01)\cs445> git log --oneline
ee55c65 (HEAD -> master) added changes
784dc1f fixed conflicts
00156d2 (login) login in login branch
03495cf login in master
35a8d39 login function in login.js
afea79c amended the commit for  hello.txt
39ee077 (tag: v1) Initial Commit
PS F:\615425_UjjwalHumagain\4MAP (CS445-2022-11A-11D-01)\cs445>
```

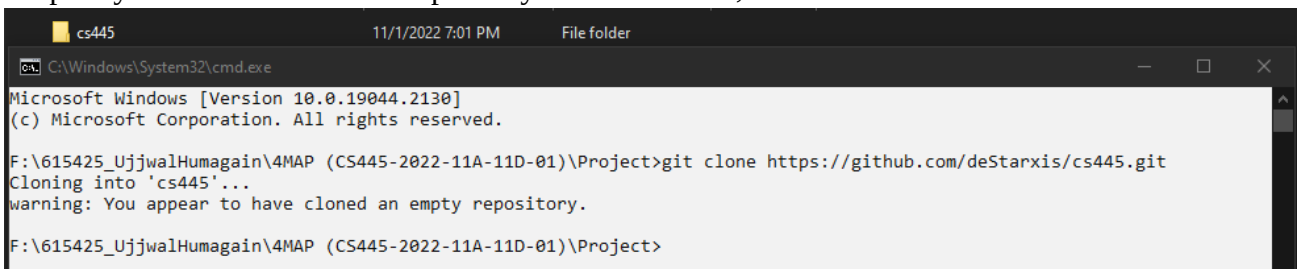
Practice with all the remote repository commands:

1. Create a remote repository



2. git clone

This command is used to clone the remote repository in the local machine. In this command we have to specify the url of the remote repository to be cloned as;



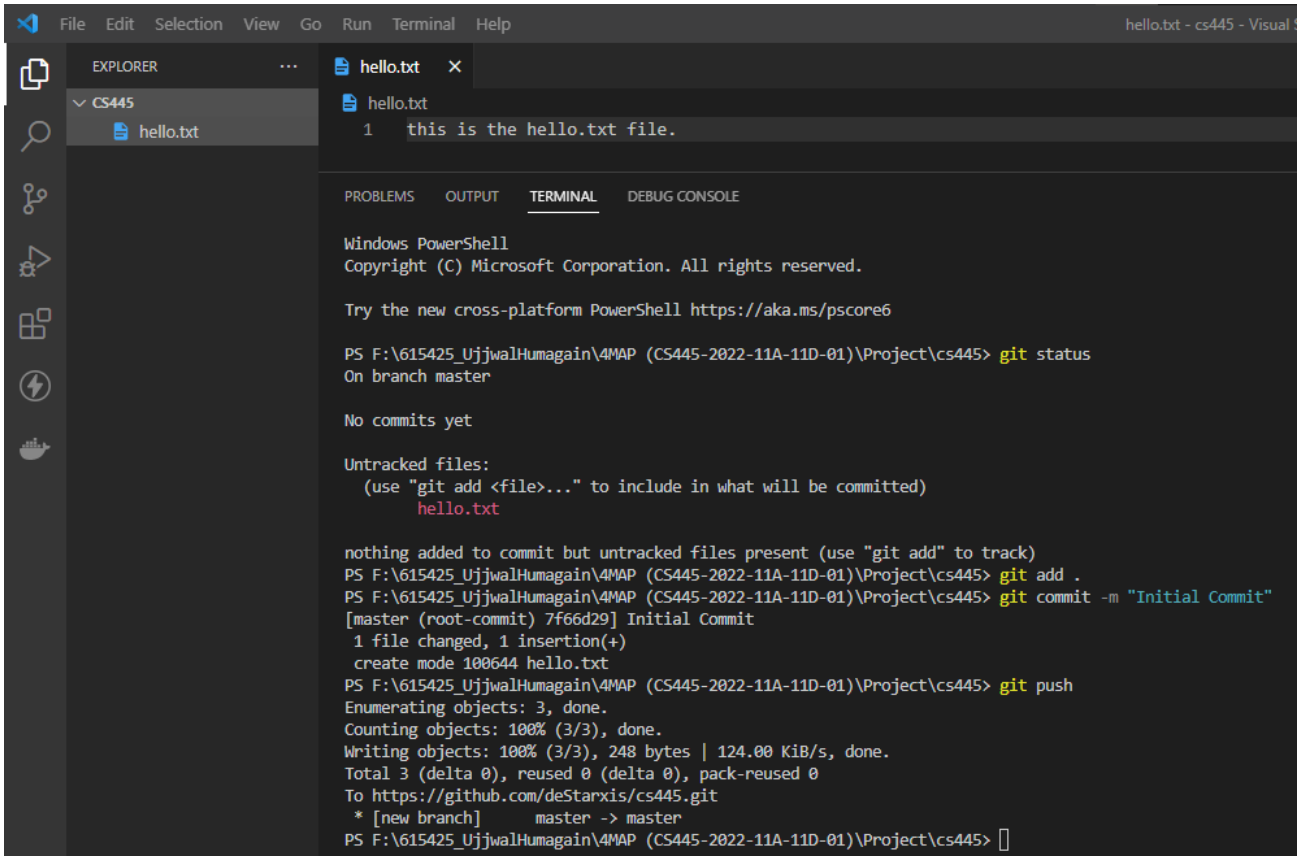
3. git remote

This command lists the current configured remote repository for the local repository.

```
F:\615425_UjjwalHumagain\4MAP (CS445-2022-11A-11D-01)\Project\cs445>git remote
origin
```

4. git push

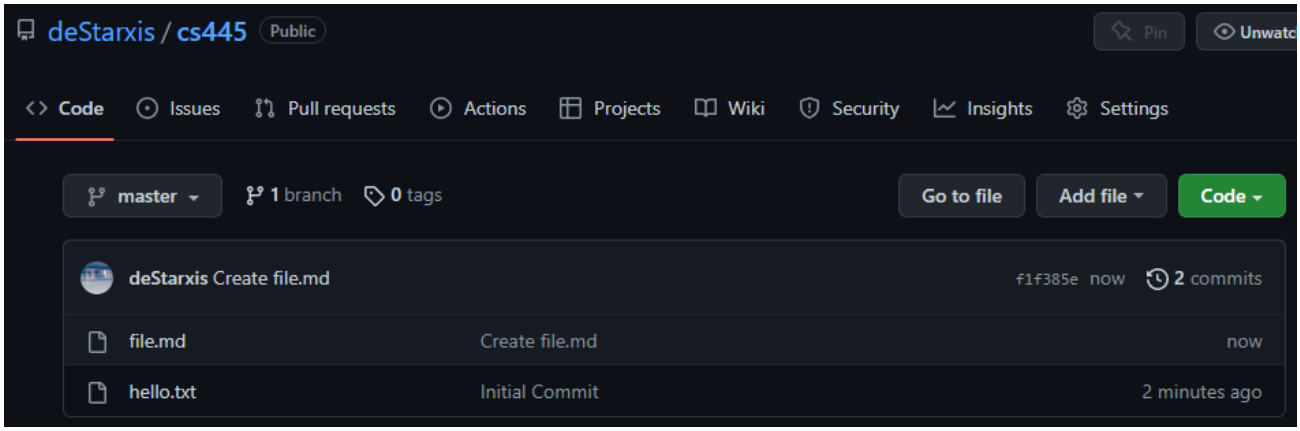
This command is used to add commits of the local repository to the remote repository.



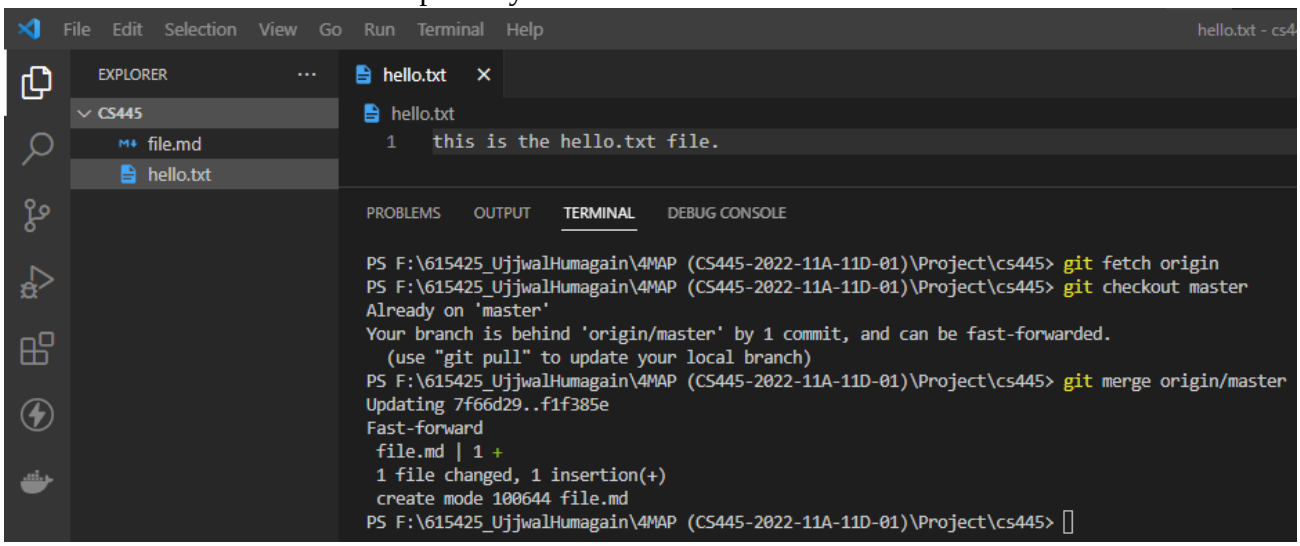
5. git fetch

This command is used to fetch the branches and their respective commits from the remote repository as;

Added new file to the remote repository which has to fetched in the local repository using ‘git fetch’.



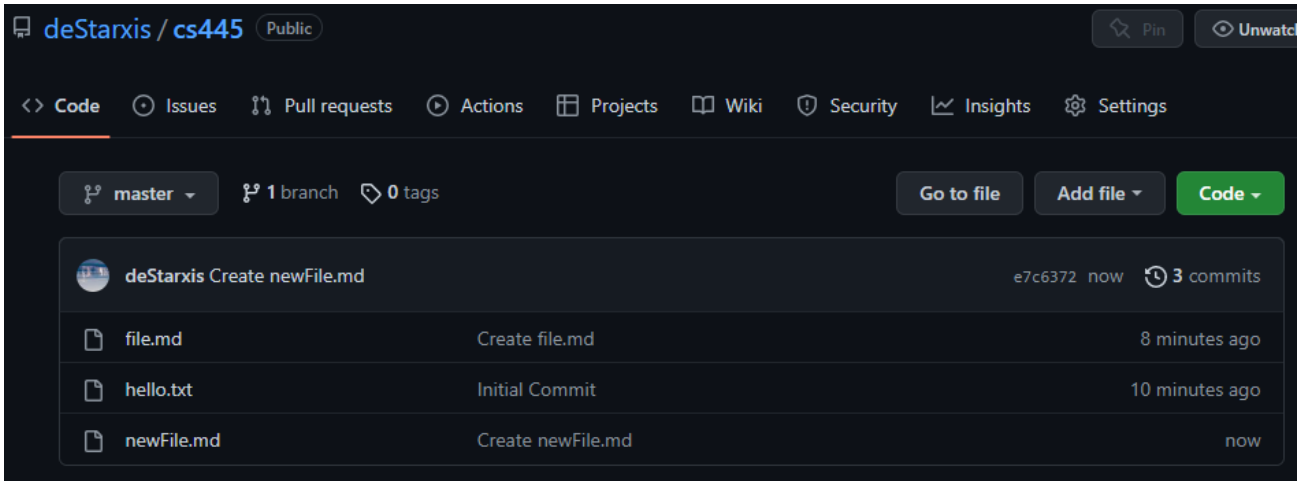
At first we need to fetch the changes from the remote and then we need to merge those changes to the default branch of the local repository.



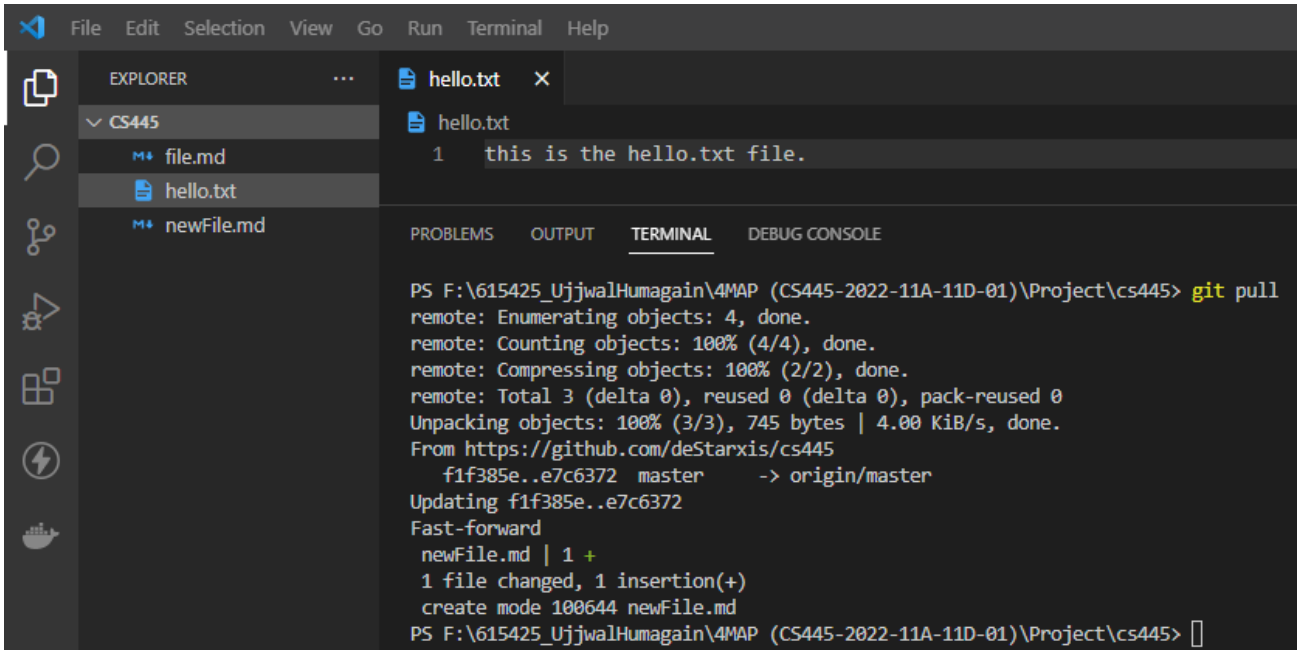
6. git pull

Despite using ‘git fetch’ and ‘git merge’ commands to get changes from remote repository to the local, we directly use the ‘git pull’ command.

Added another file to the remote repository.

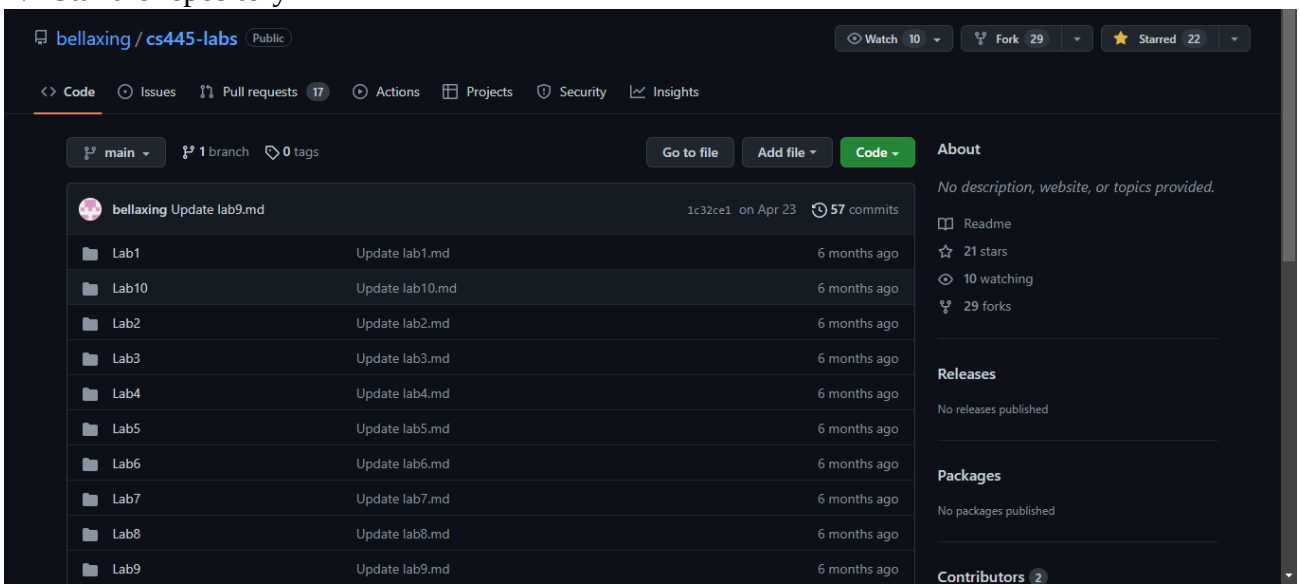


Using ‘git pull’ command to pull everything from remote repository to local.

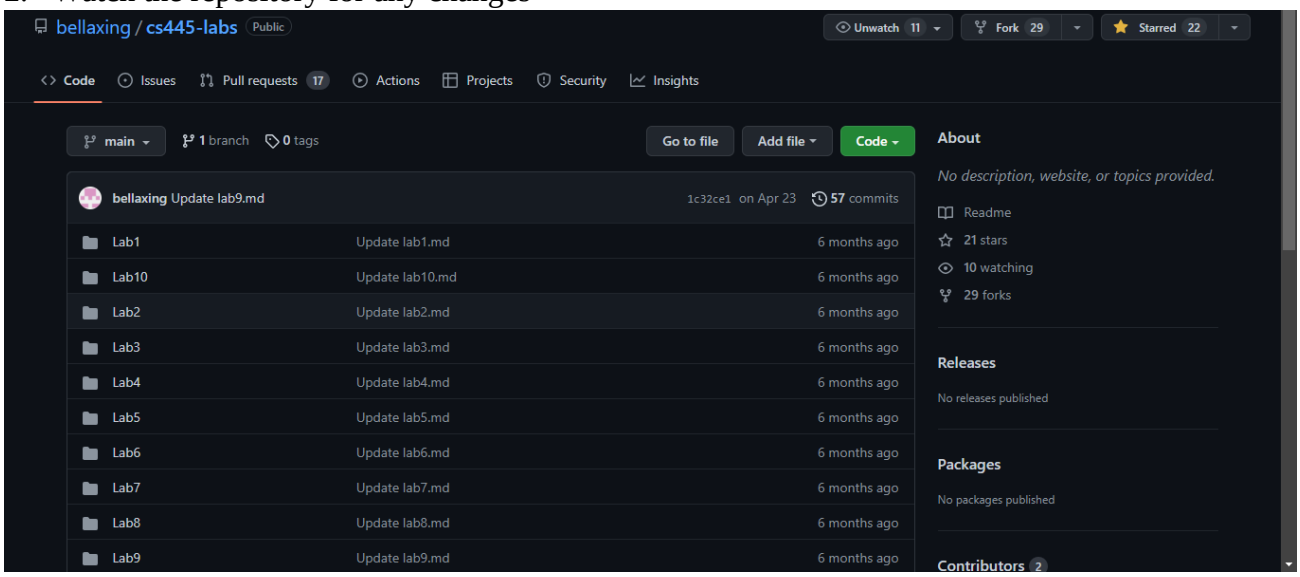


Practice with all the third-party repositories commands:

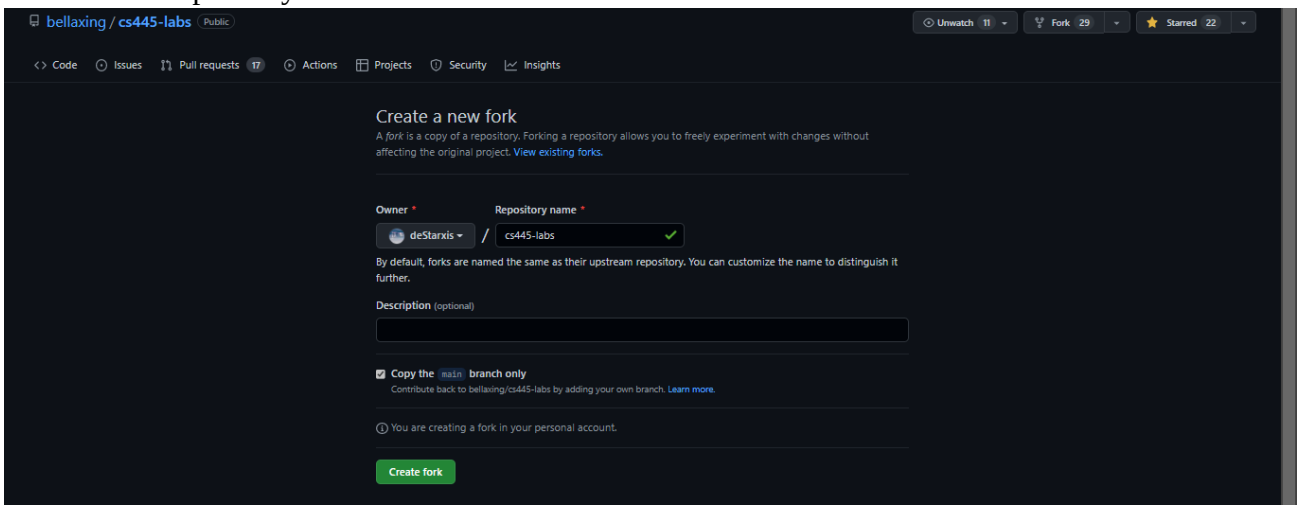
1. Star the repository



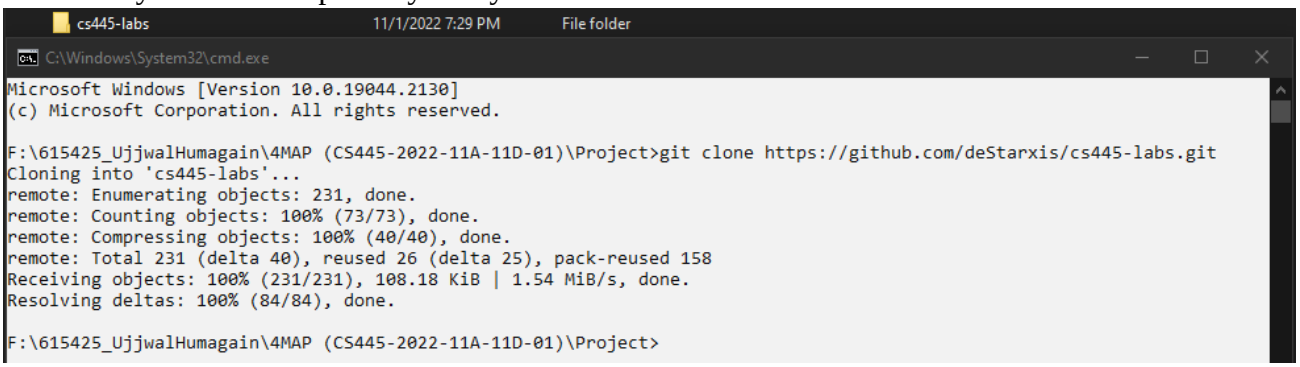
2. Watch the repository for any changes



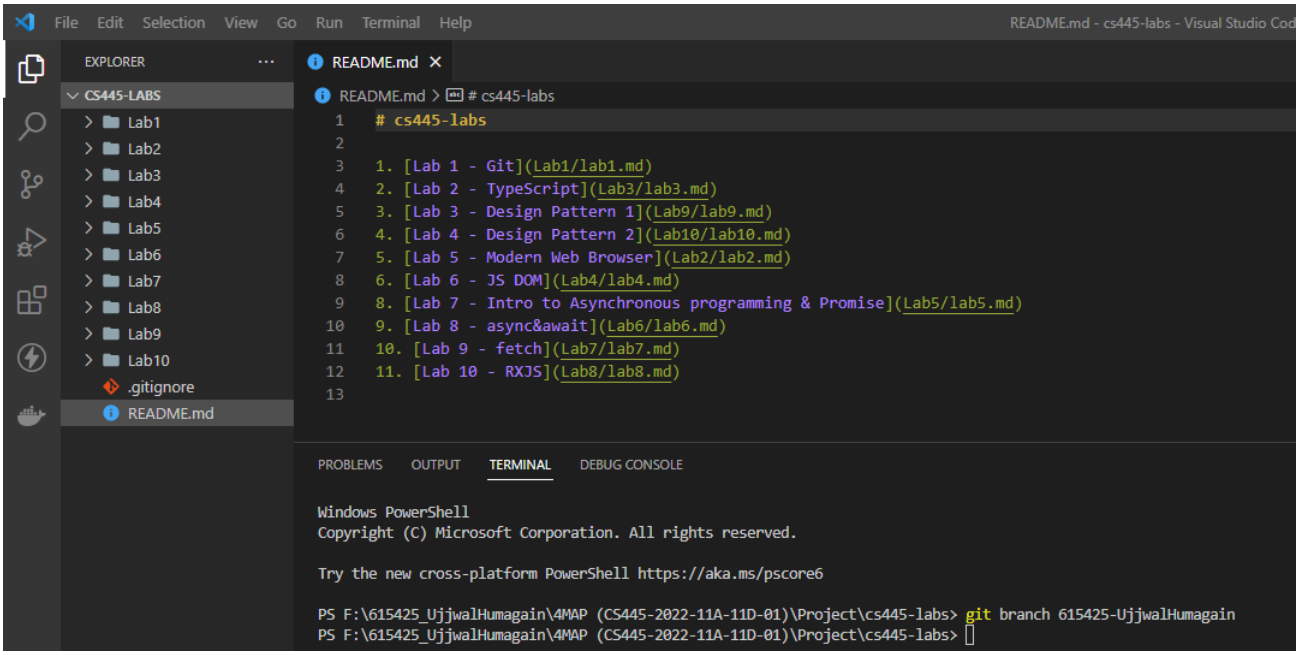
3. Fork the repository



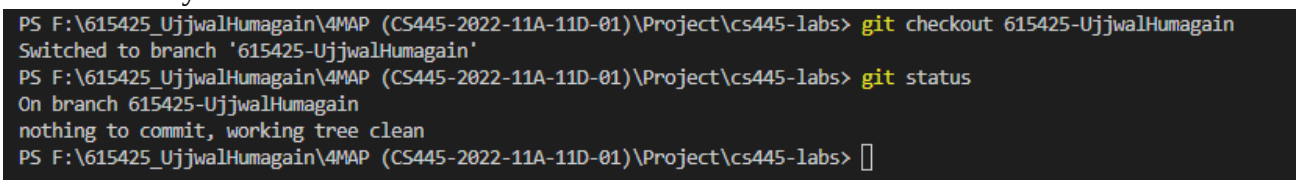
4. Clone your forked repository into your local machine



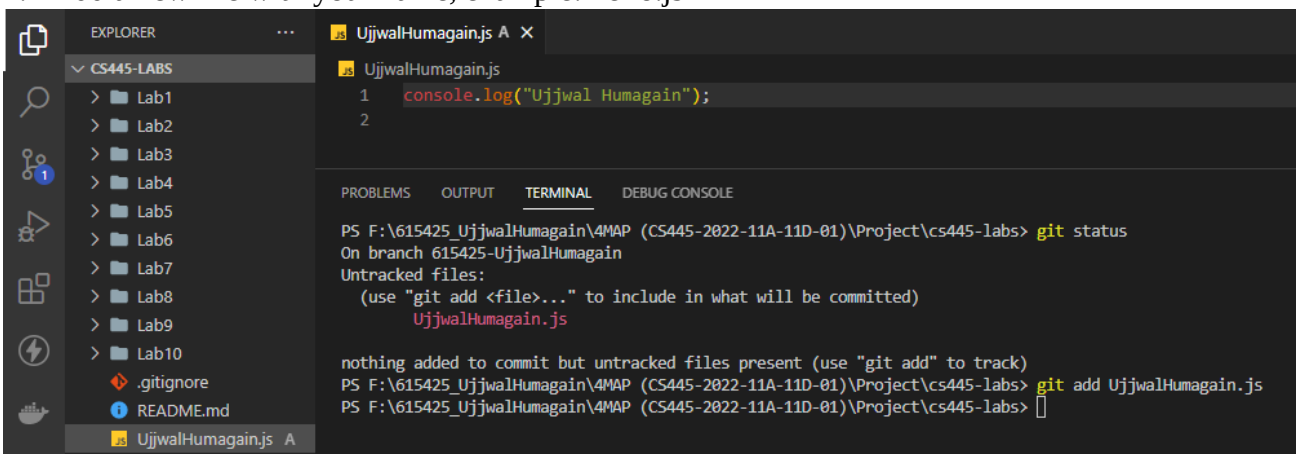
5. Create a new topic branch using studentId-studentName format, For example: 610001-JohnSmith



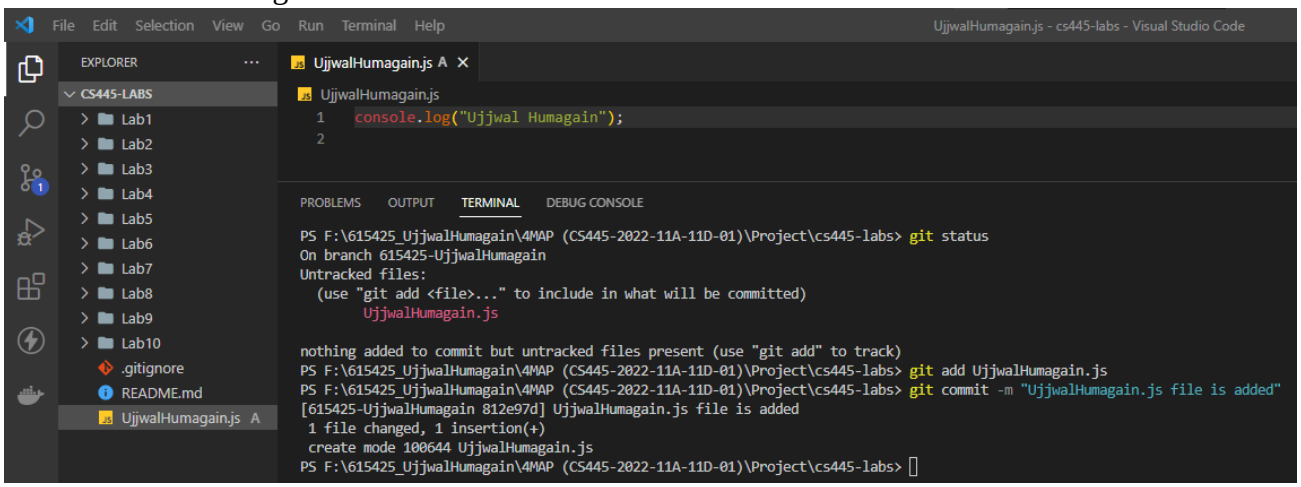
6. Checkout your new branch



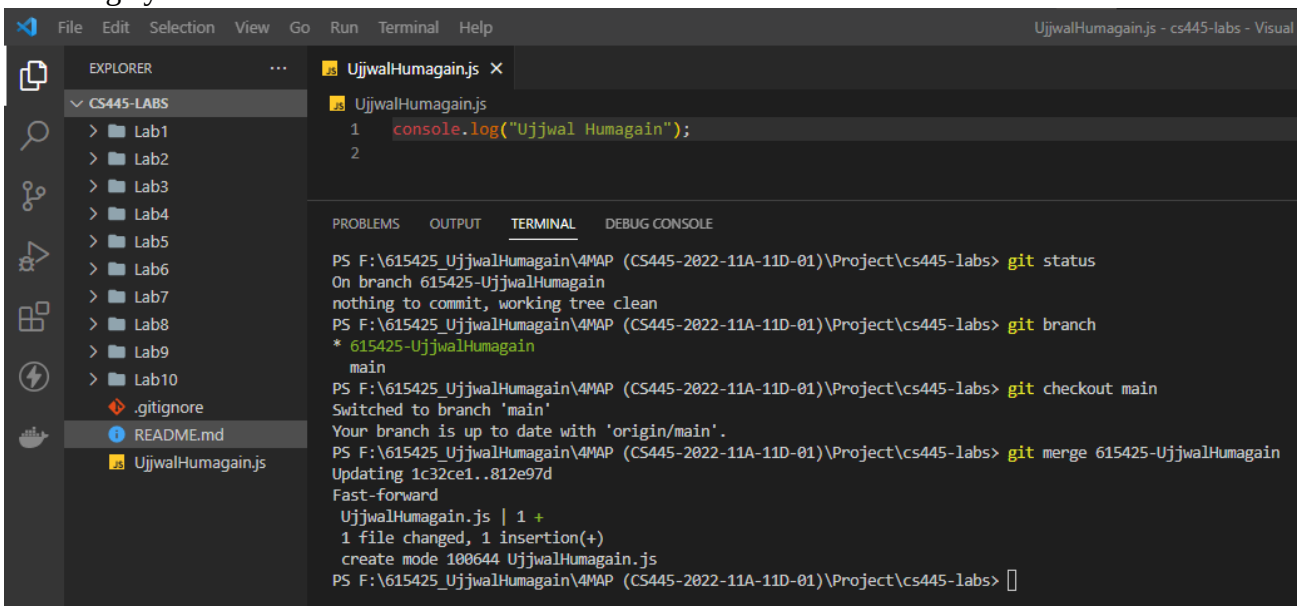
7. Add a new file with your name, example: hello.js



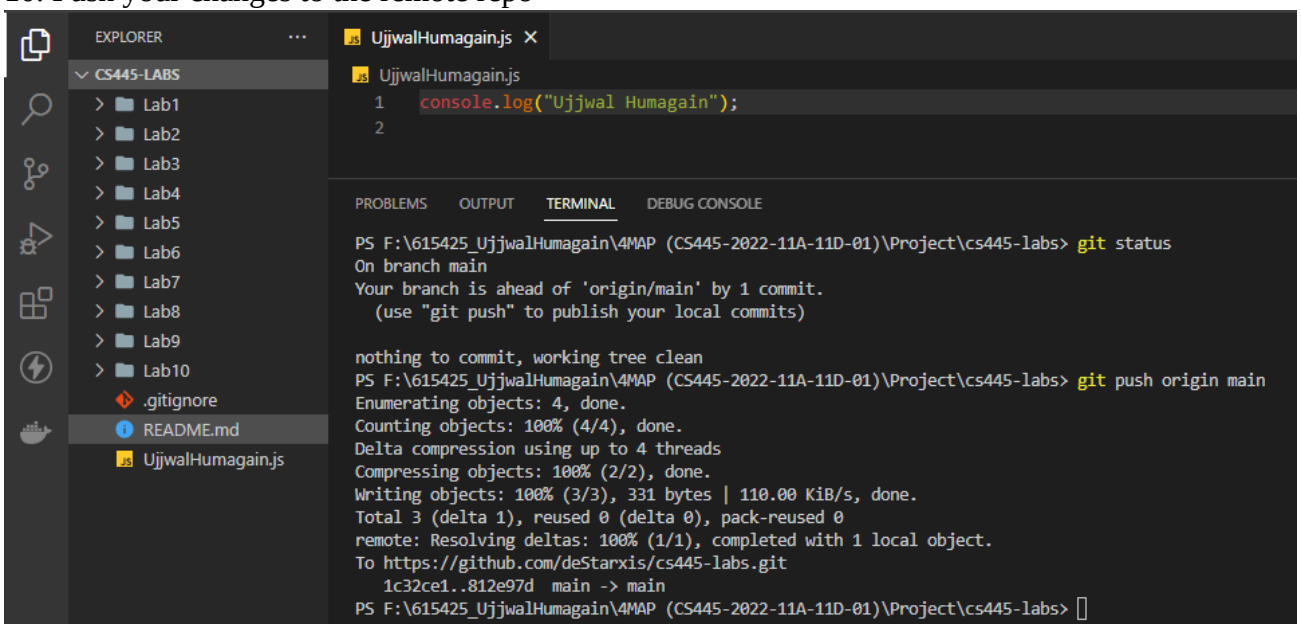
8. Commit the changes



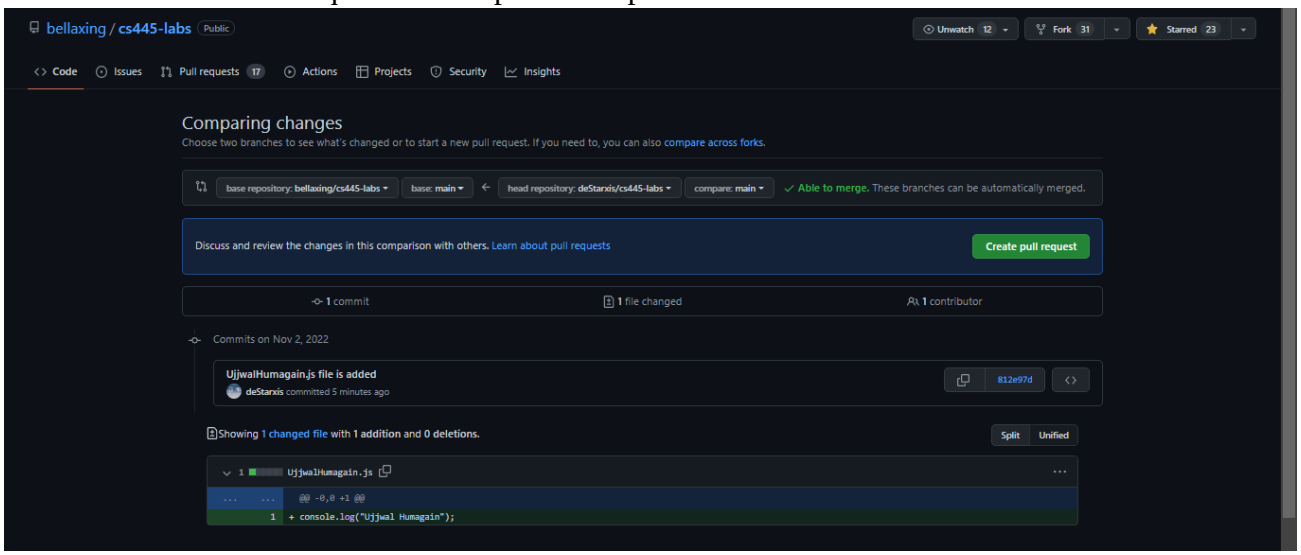
9. Merge your studentId-studentName branch with the main branch

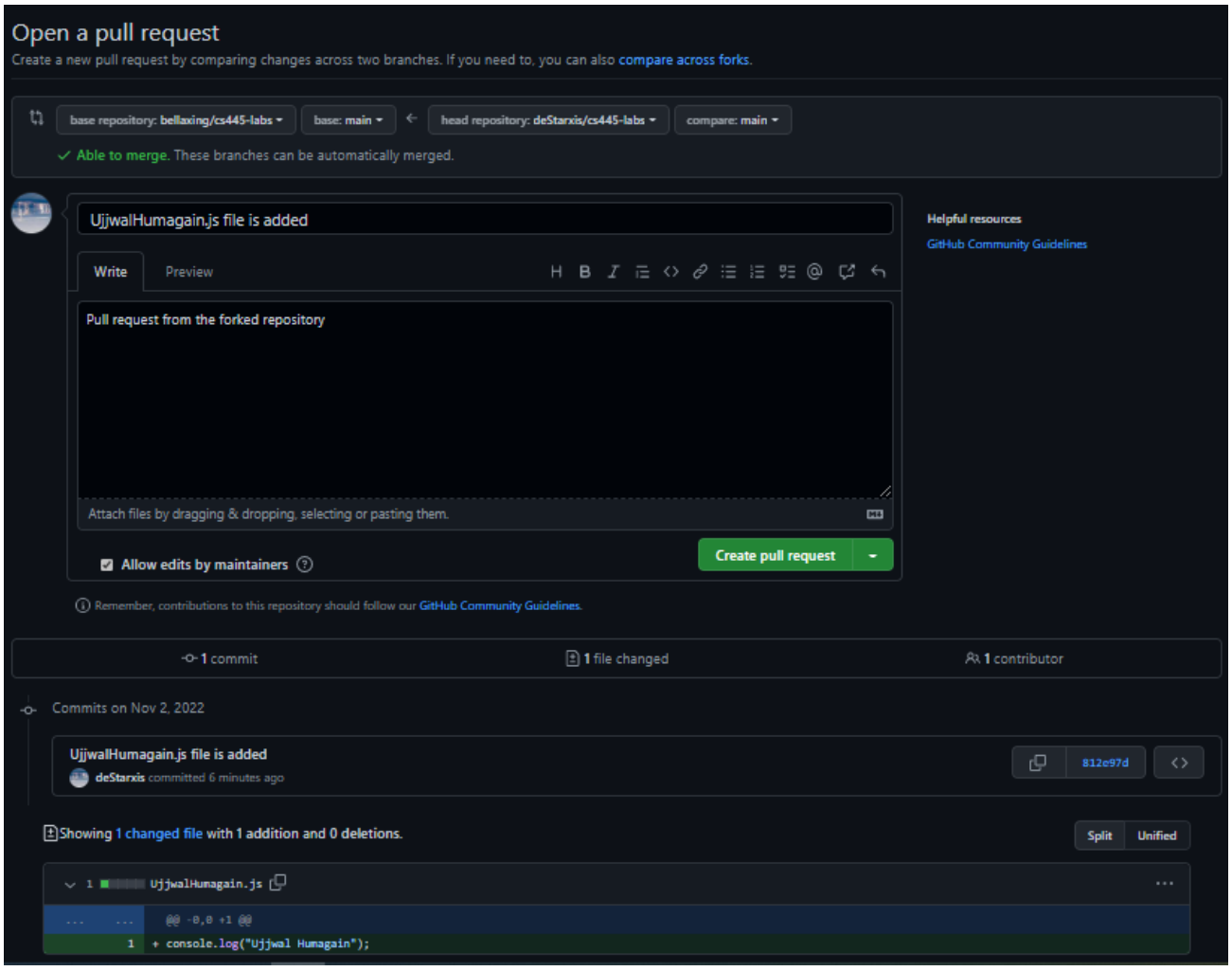


10. Push your changes to the remote repo



11. Create a new Pull Request to the upstream repo





The link to the forked repository is:
[deStarxis/cs445-labs (github.com)]