Subject Name: **Source Code Management**

Subject Code: **CS181**

Cluster: **Beta**

Department: **DCSE**

Submitted By:
AARYAN
PRABHAKAR
2110990018
G01

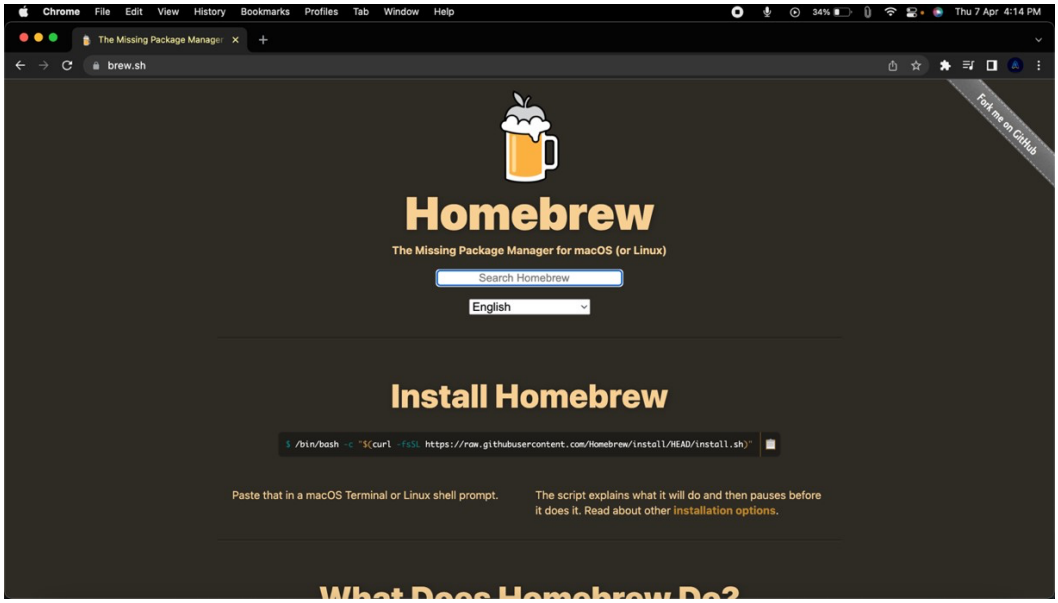
Submitted To:
Dr. Monit Kapoor

Experiment No. 01

Aim: To set up Git Client.

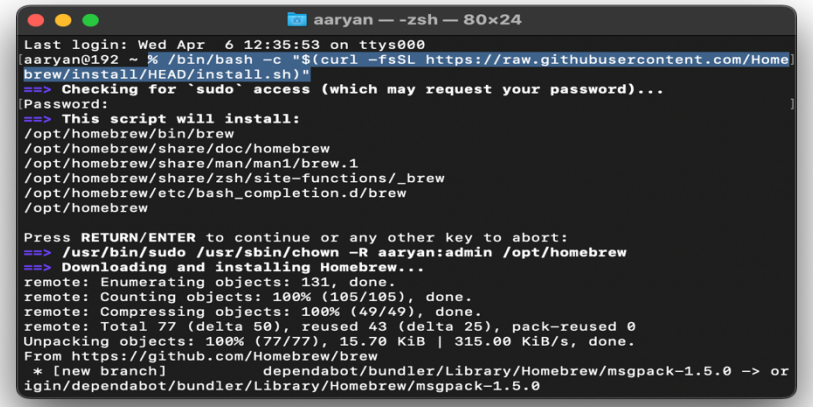
Procedure:

1. Visit the provided link for the installation of Homebrew in your system (https://brew.sh)



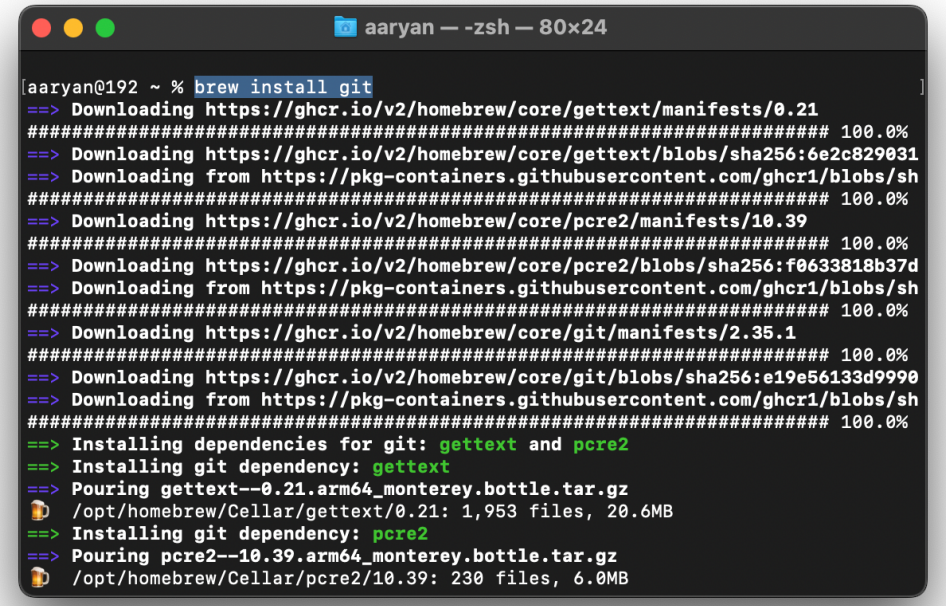
2. After opening this site, copy the command and paste it in your terminal and wait for installation to complete.

```
/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
```



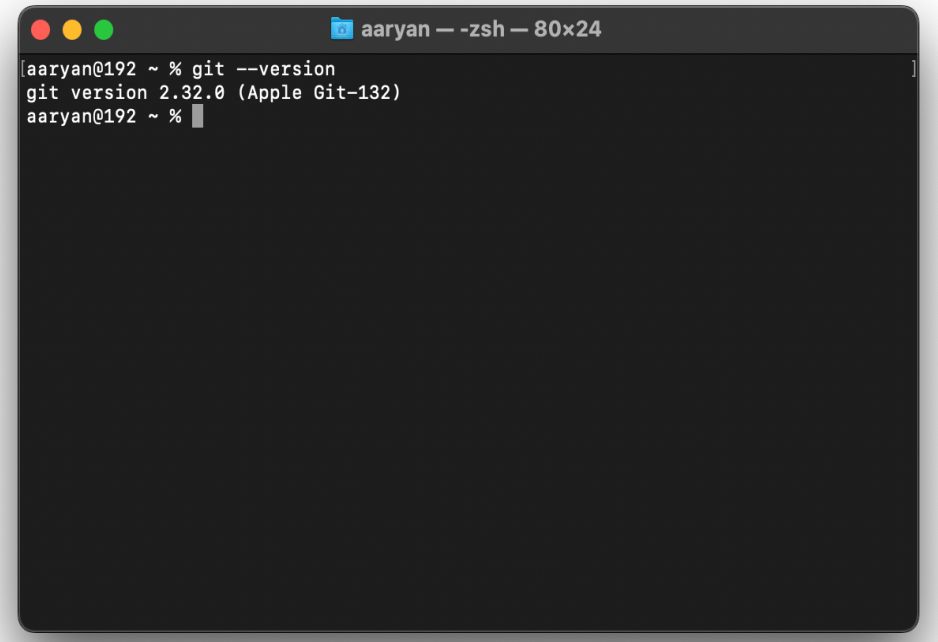
3. Now type the following command in terminal to install git in your system

```
brew install git
```



4. After the installation of git you can check whether it is installed or not by typing the following command in your terminal.

```
git --version
```



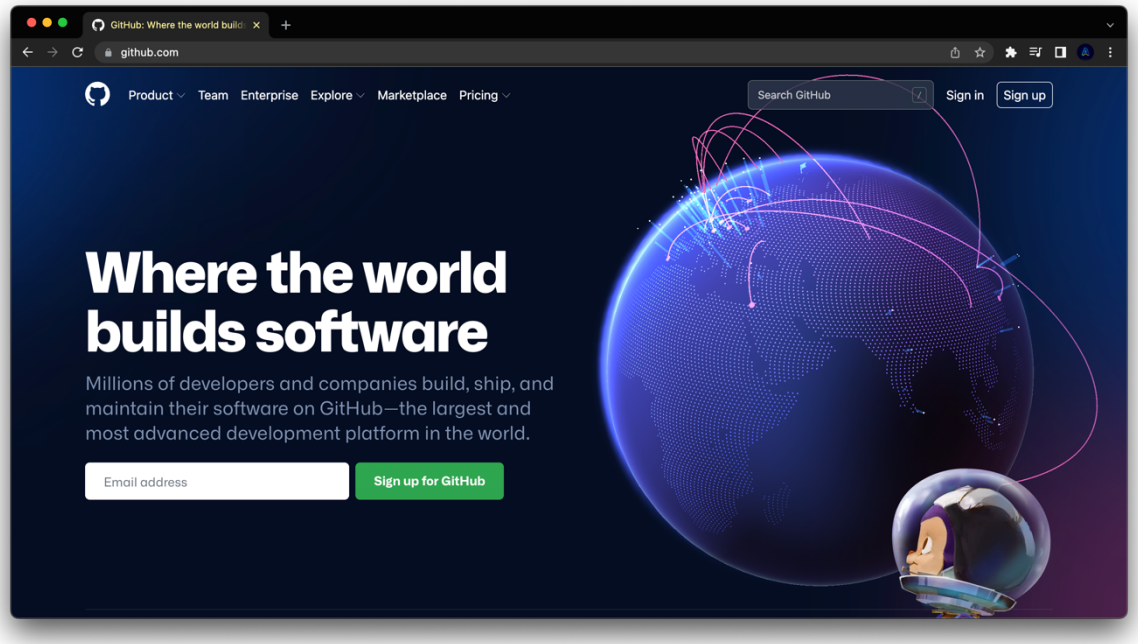
5)As we can see we have successfully installed git into are system.

Experiment No. 02

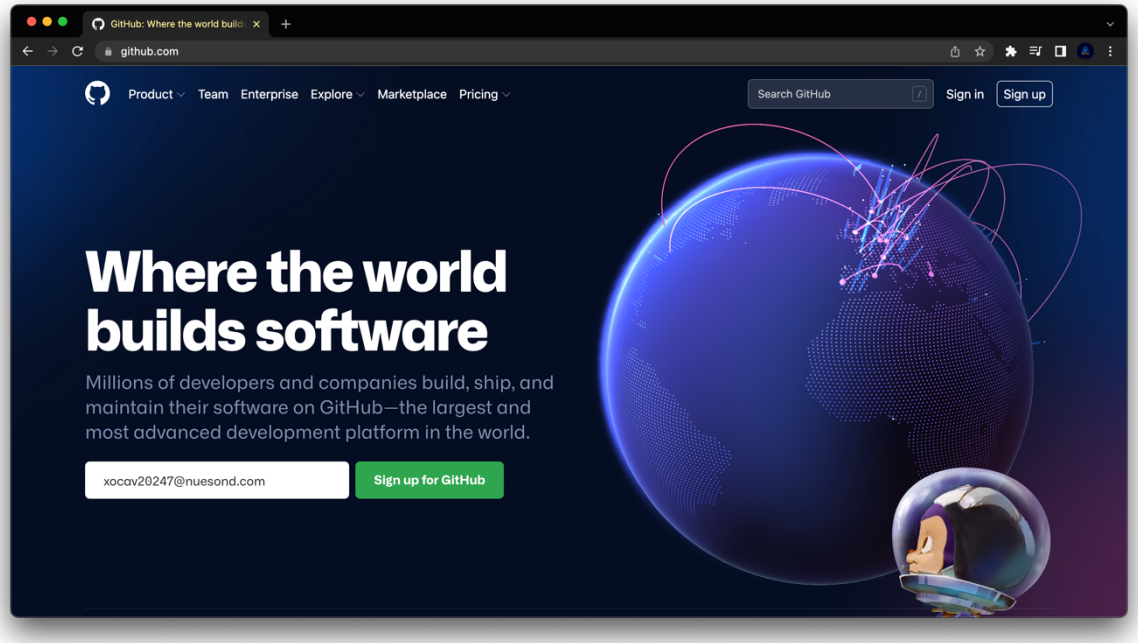
Aim: To set up GitHub Account

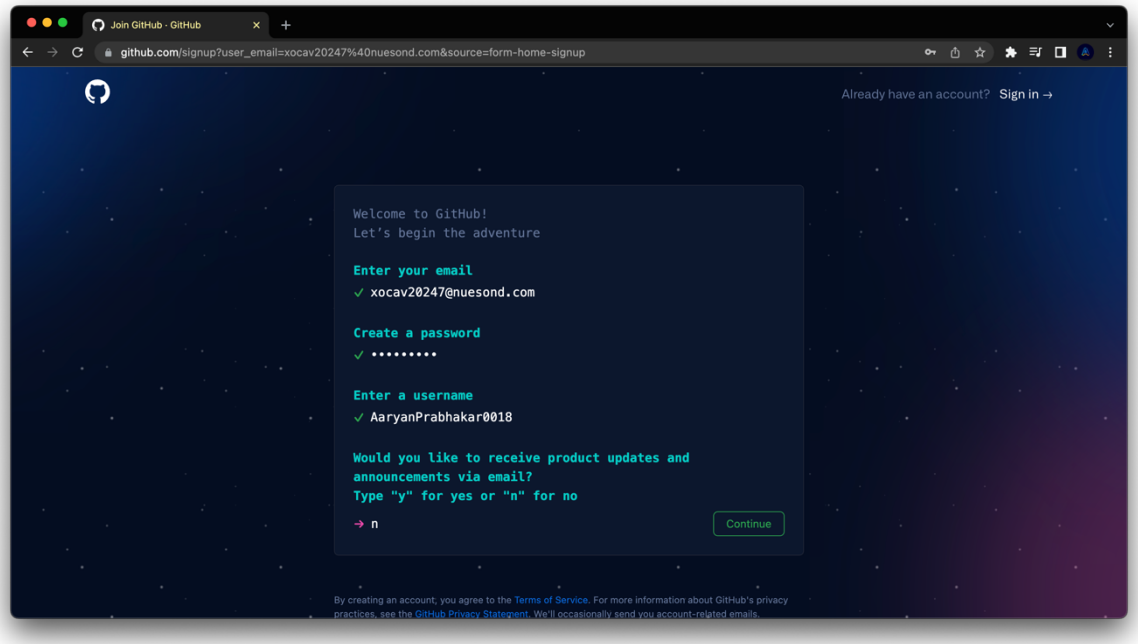
Procedure:

1. Click on https://github.com to open GitHub on browser.

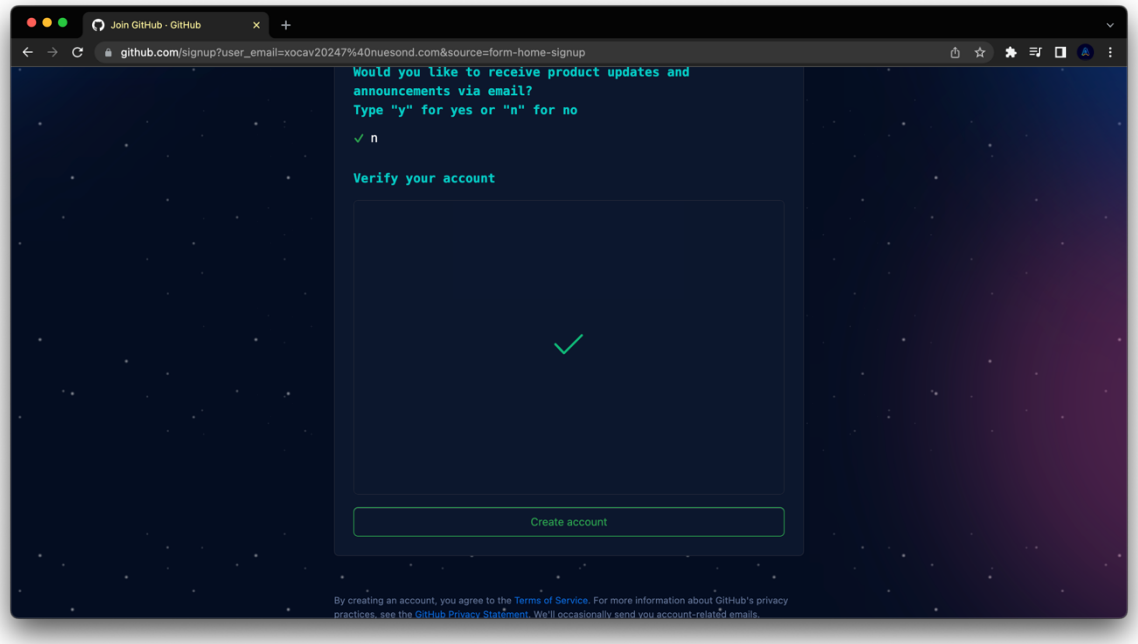


2. Type in your email in the Email address box and click on Sign up for GitHub. On the next page click on continue.





3. Solve the puzzle and Click on Create Account.



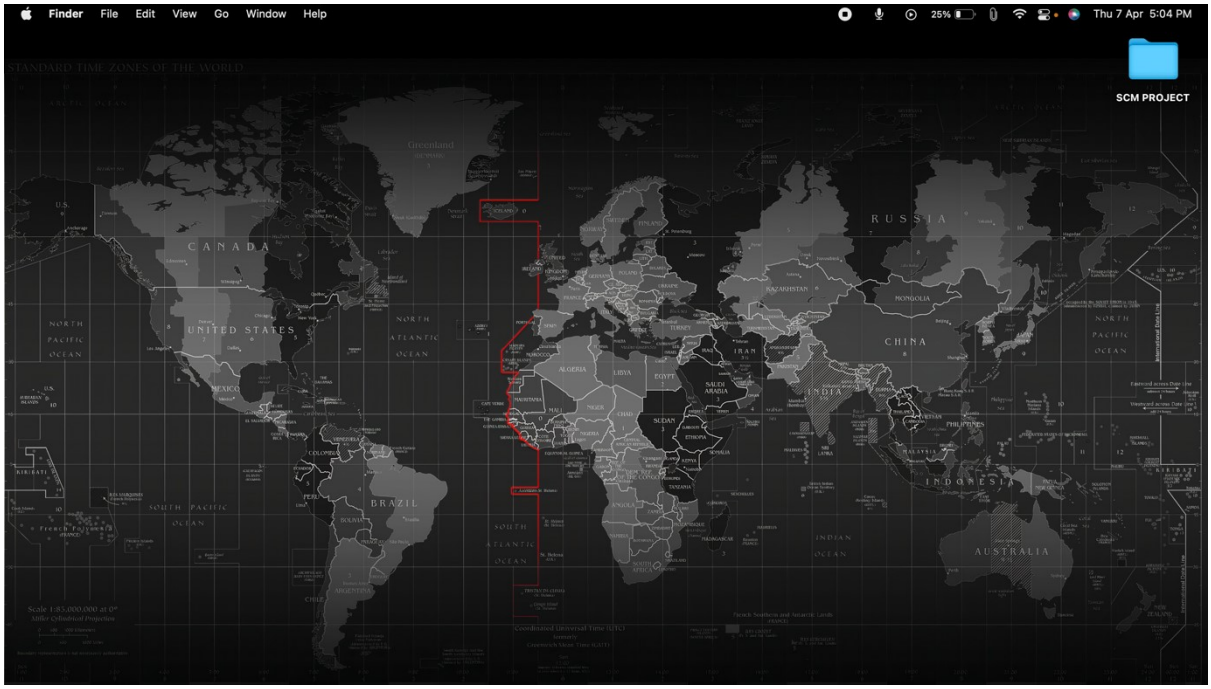
4. Your homepage of GitHub will be loaded and your GitHub account is created successfully.

Experiment No. 03

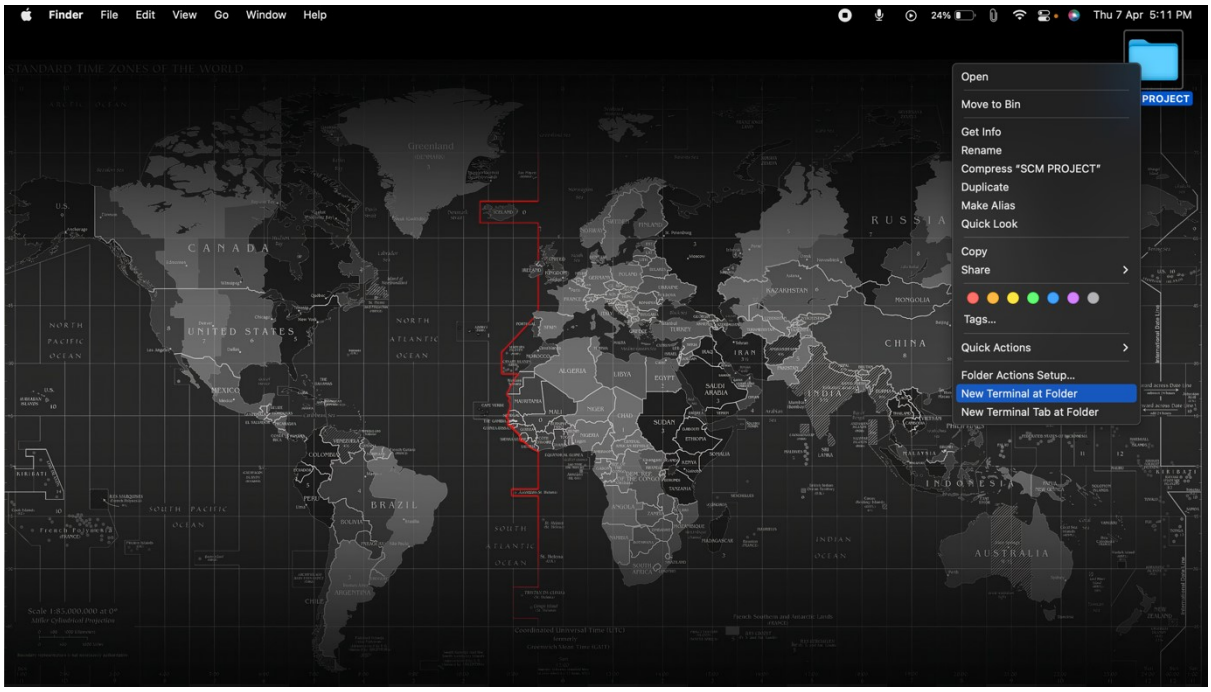
Aim: To create a program to generate logs.

Procedure:

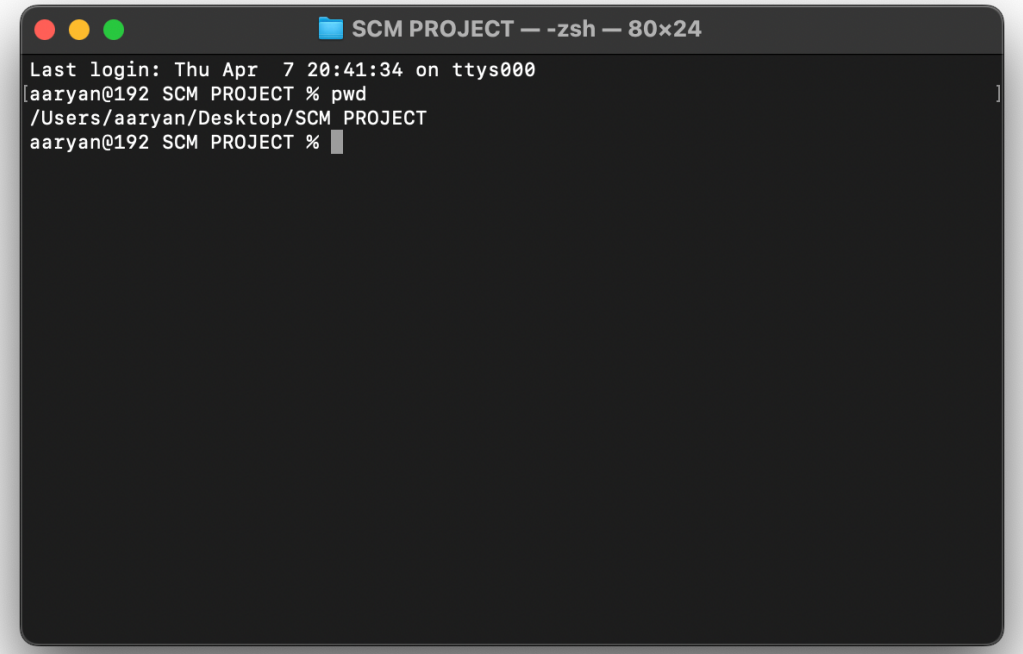
- **Initialising Git Repository:**
  1. Make a new folder and name it SCM PROJECT.



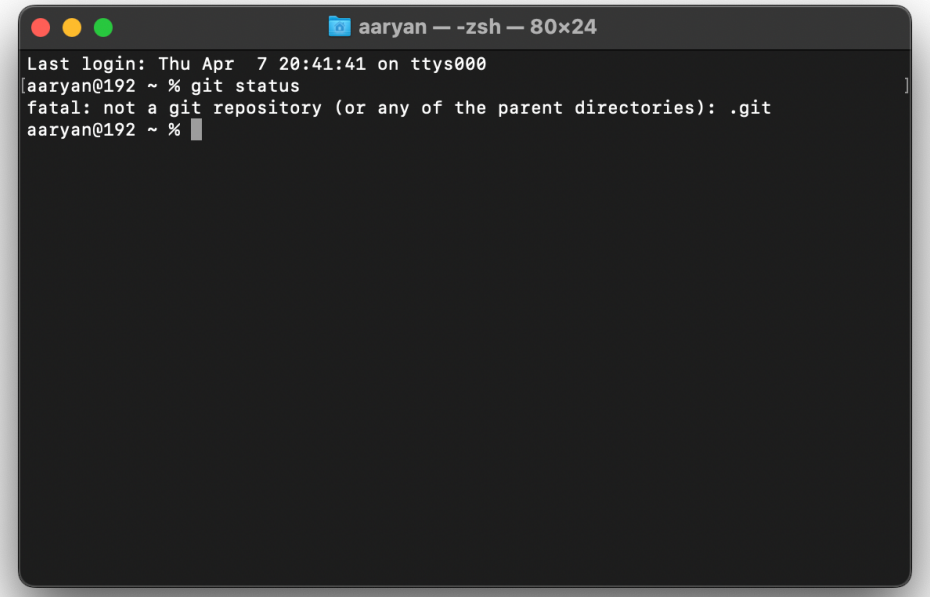
2)Now right click on the folder and select **new terminal at folder** option from the menu.



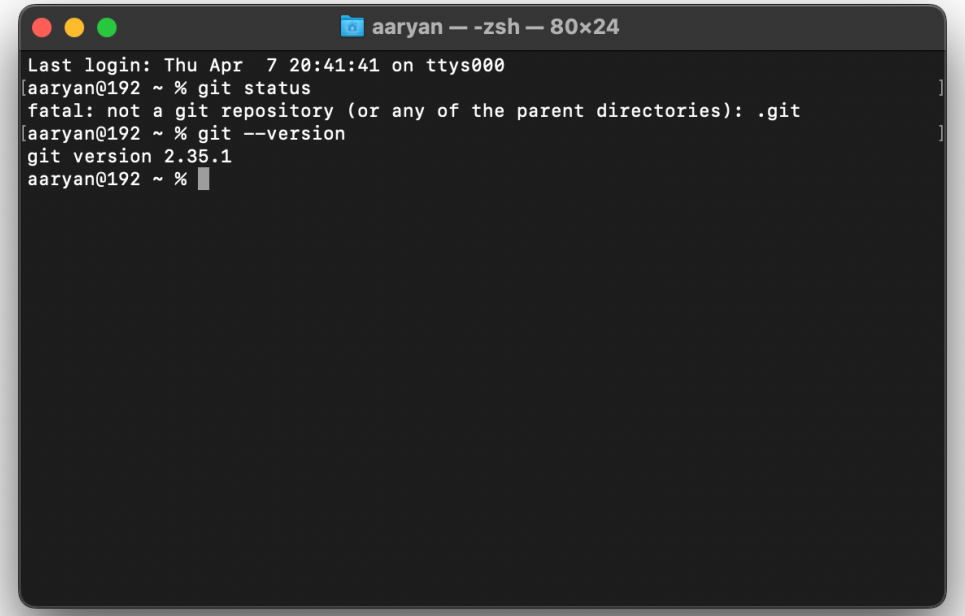
3) Firstly, check the Present Working Directory by typing the command **pwd** and press Enter.



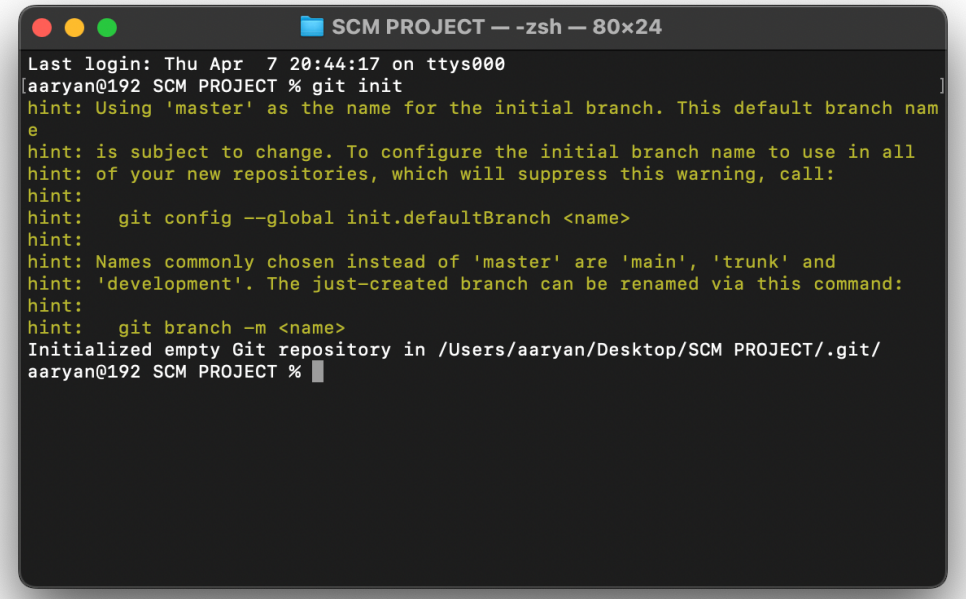
4) Now, to know every status about the files in the repository, type command ***git status.***



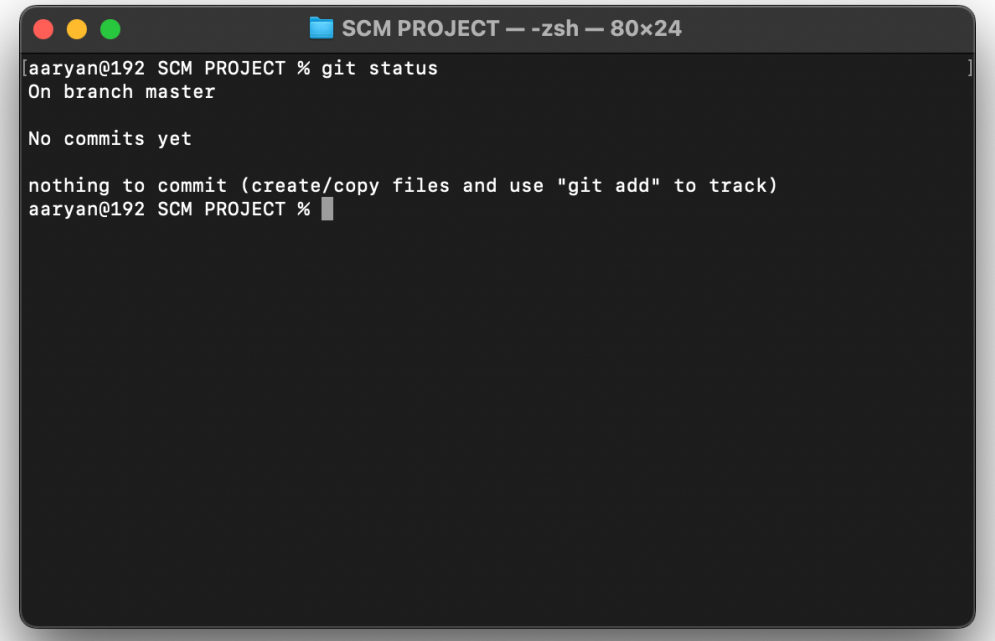
5) To get the information about the version of your git, type the command ***git --version***



6) Now, to initialize git repository type in command ***git init***

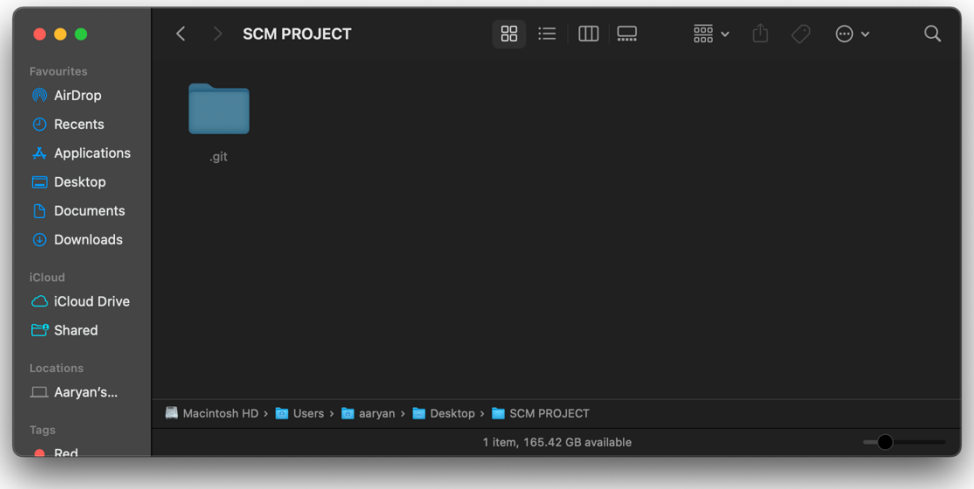


7) Check the status of your repository by typing the command **git status**



8)Now go to your folder.

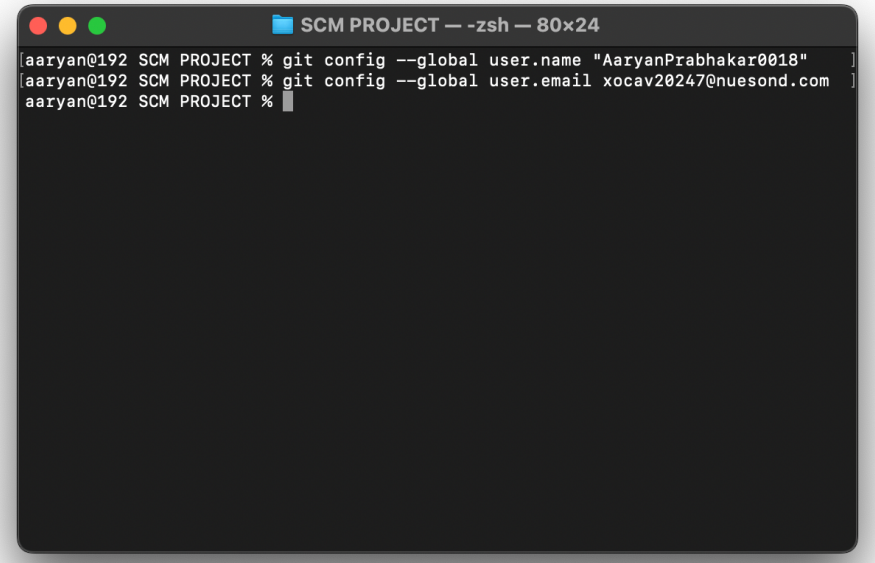
9) By default, the **.git** folder is hidden. To see the **.git** folder go inside the folder and press **Command+shift+period(.)**



- **Configuring Username and User Email:**

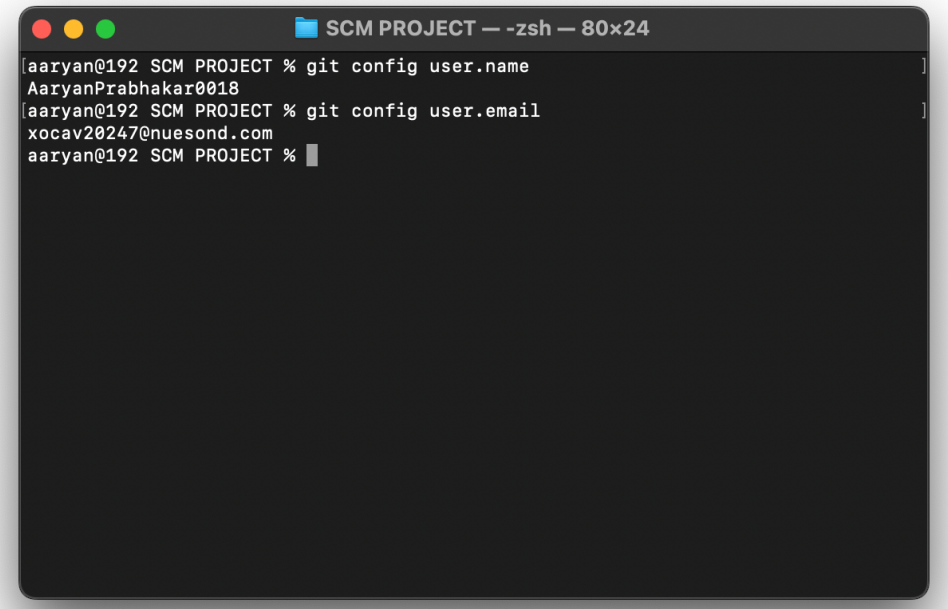
So, now that you have successfully created your GitHub account. It is best to use your username and email of GitHub in Git. So, let's update our username and user email

1) Type command *git config --global user.name "YOUR USERNAME".* It will update your username.
2) Similarly, you can update your email using the command *git config --global user.email "YOUR EMAIL"*.



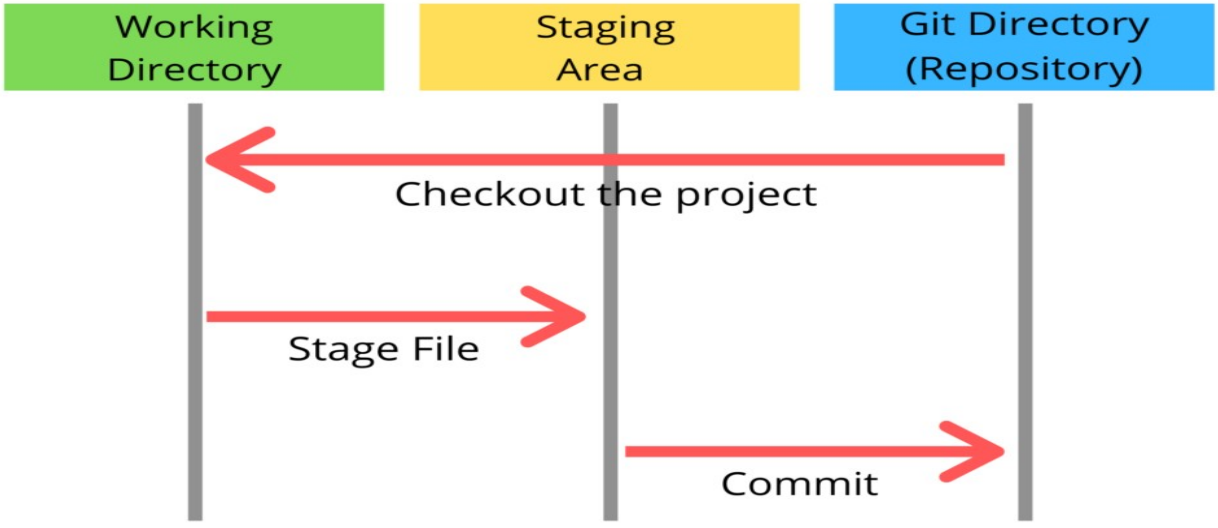
3)To verify the updated username type command ***git config user.name*** **.**

4) Similarly, email can be verified using ***git config user.email***



- **Three-Stage Architecture:**

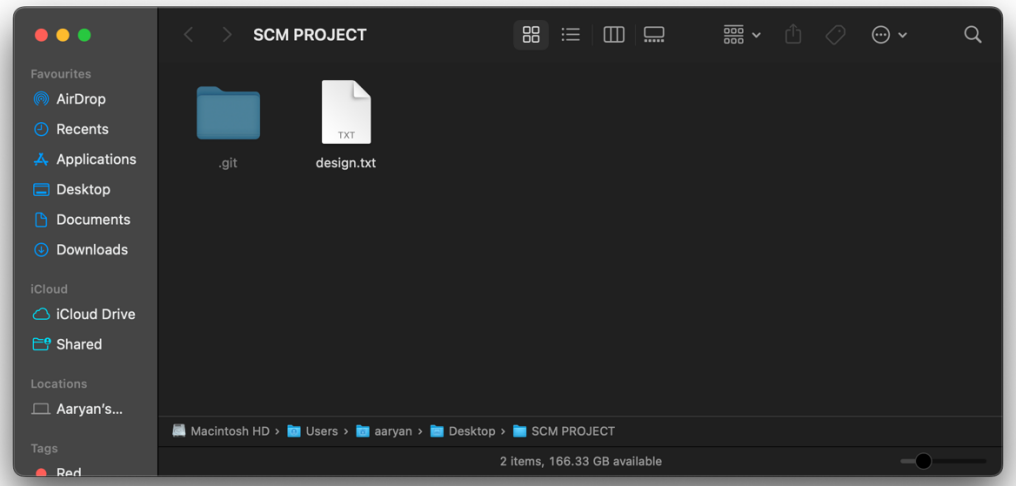
Many VCS's use a two-stage architecture i.e., a repository and a working copy. Git uses three-stage architecture i.e., a working directory, staging area and local repository.
Working Directory is the folder which we created i.e., SCM Project.
Working Directory was converted into Repository where we used ***git init*** command to initial the repository.



Working directory specifies the file explorer's folder where your files are stored, Staging area is an area where those files are present which you want to send to commit (to create snapshot of files). After commit is fired, files which are in staging area will move to Git Repository.

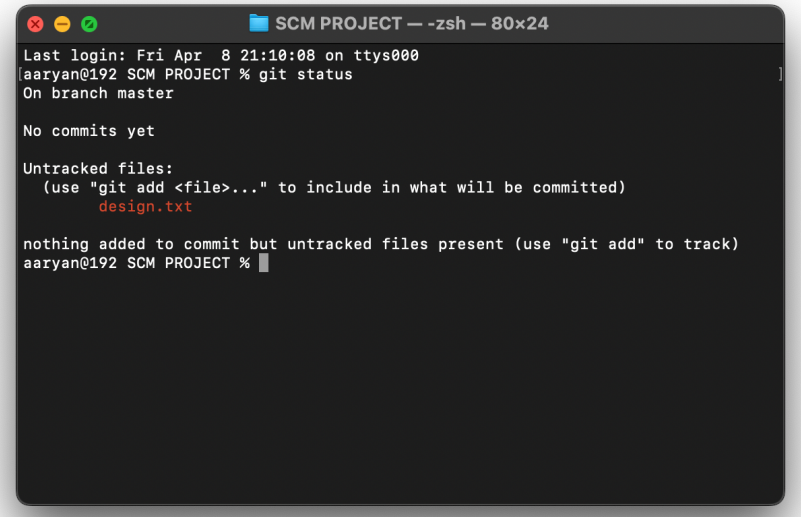
Now if you made any changes in your files which are in Git repository, those have to be added into the Staging area and Committed

1)Create a file in the empty Git Repository (SCM Project folder). Let's create a .txt file and name it as design.txt



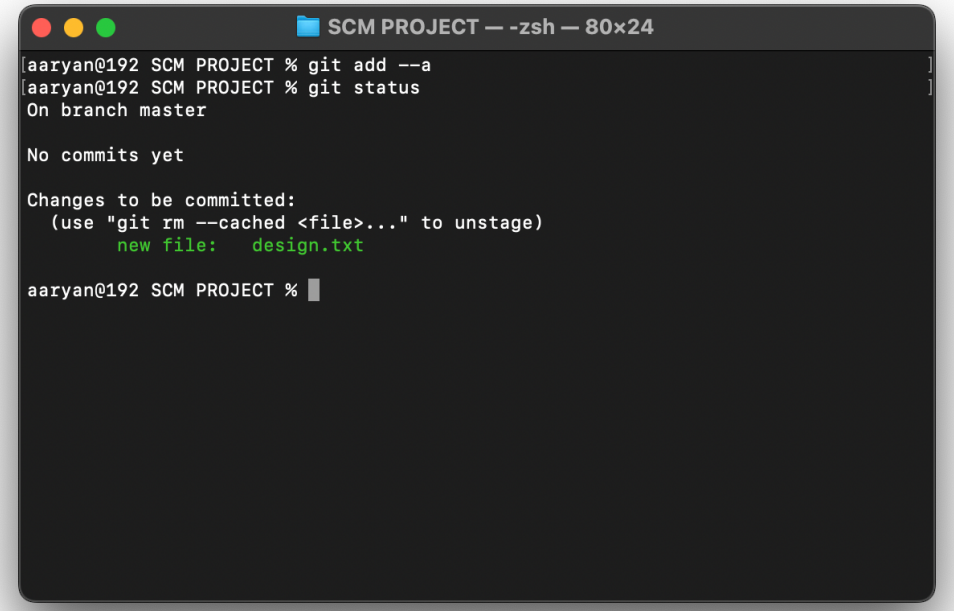
2)Now, open Git Bash and type command *git status.*

We can clearly see that the design file is in Untracked files and it is colored in red



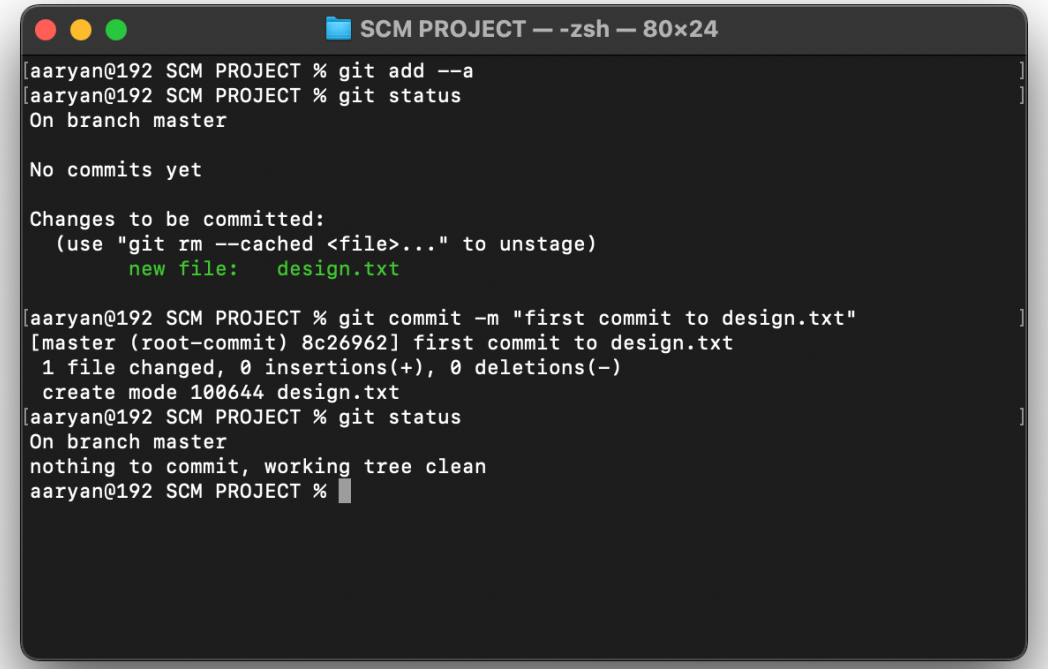
3)Now type ***git add --a*** which will move or add the files to the Staging area. Here --a means all the latest files changed or added. You can also specify a file like ***git add design.txt.***

4) After that check status using ***git status***

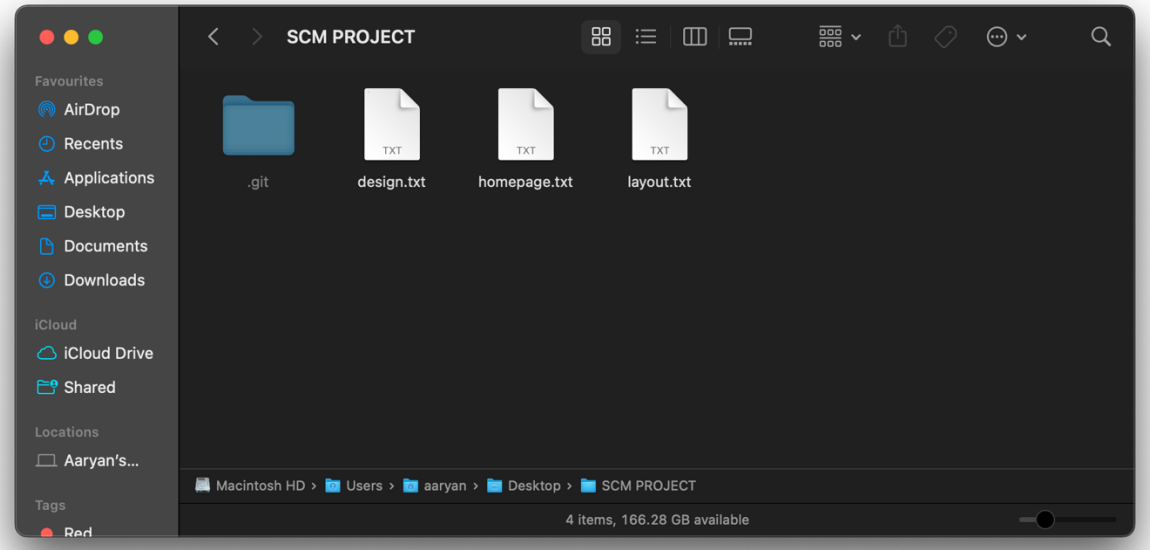


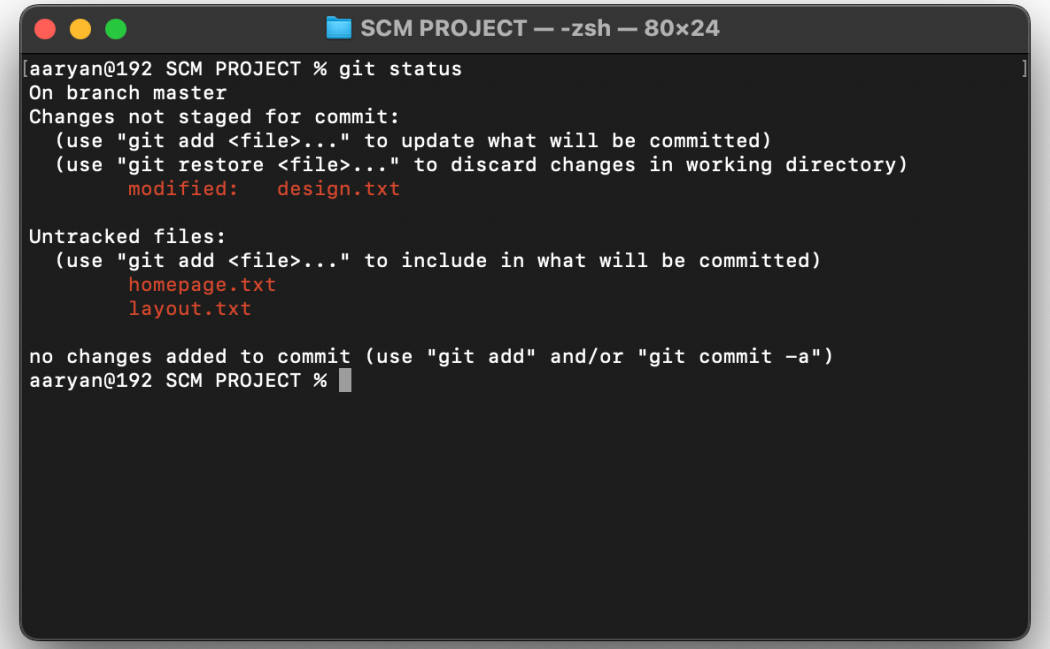
Observe that the design.txt has turned from red to green. This means that file has been added to the staging area and is ready to be committed

5) Now to commit the files in the staging area, use command ***git commit -m "Any Message"*** and check the status

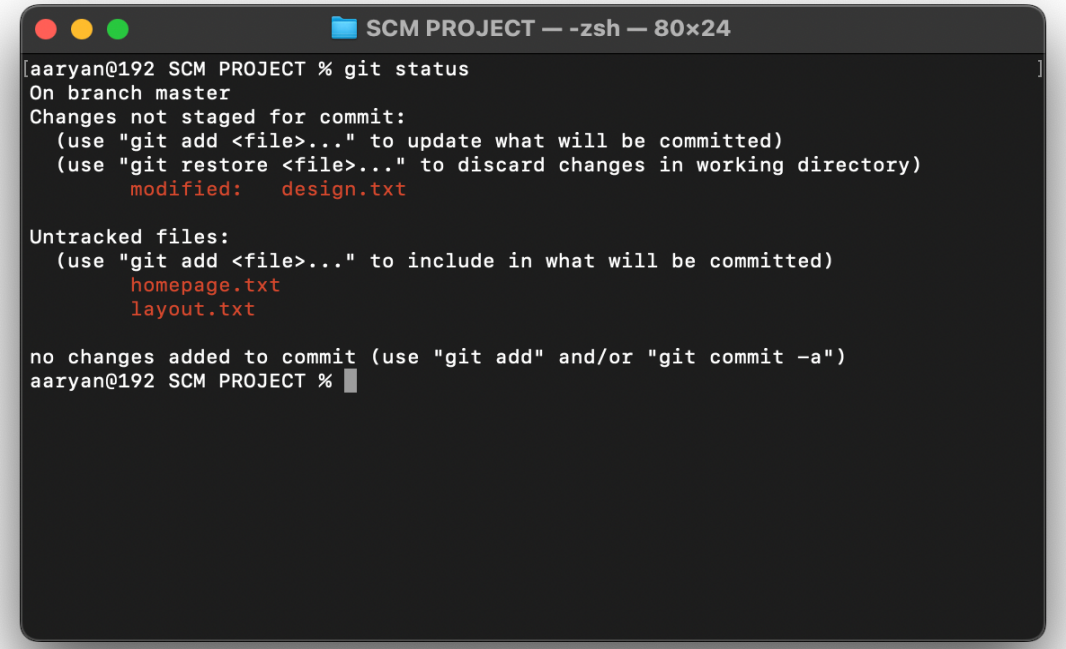


6) Now let's check the status. It shows that design.txt is modified and homepage.txt and layout.txt are untracked. All these changes have to be committed.

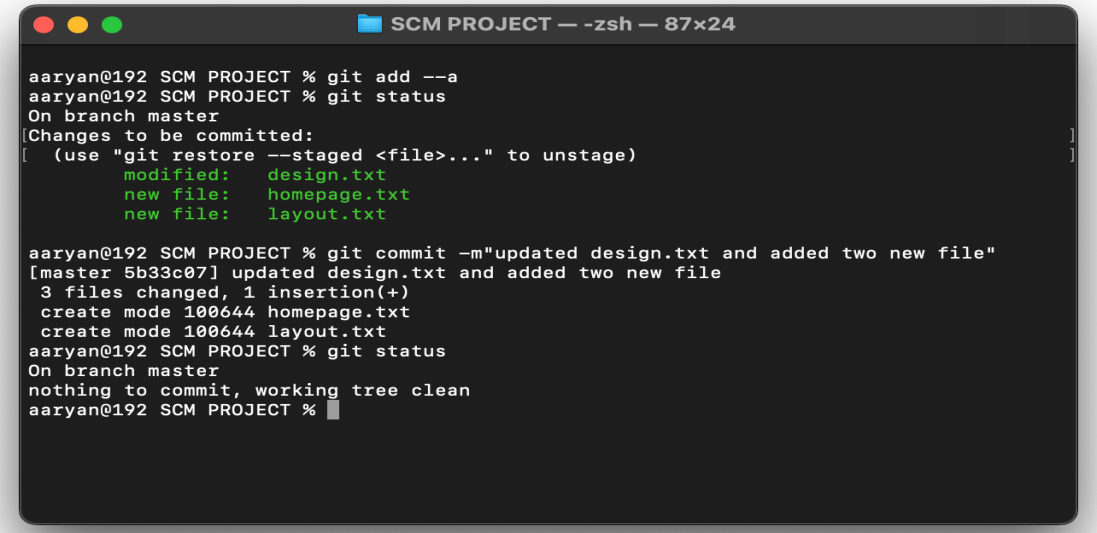




7)Now let's check the status. It shows that design.txt is modified and homepage.txt and layout.txt are untracked. All these changes have to be committed.

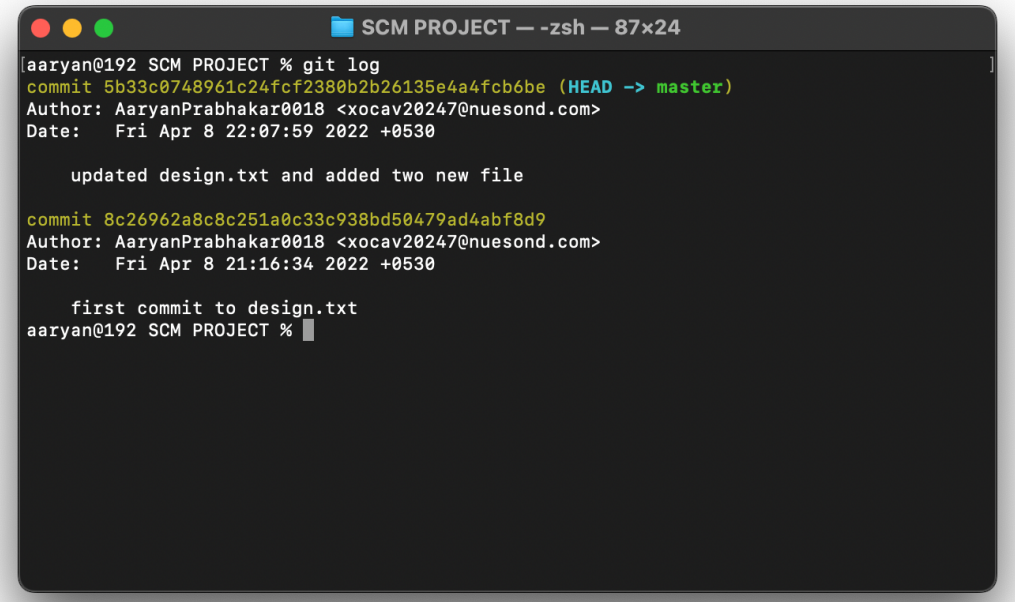


8)Now we have to commit these changes .



➢ **Logs:**

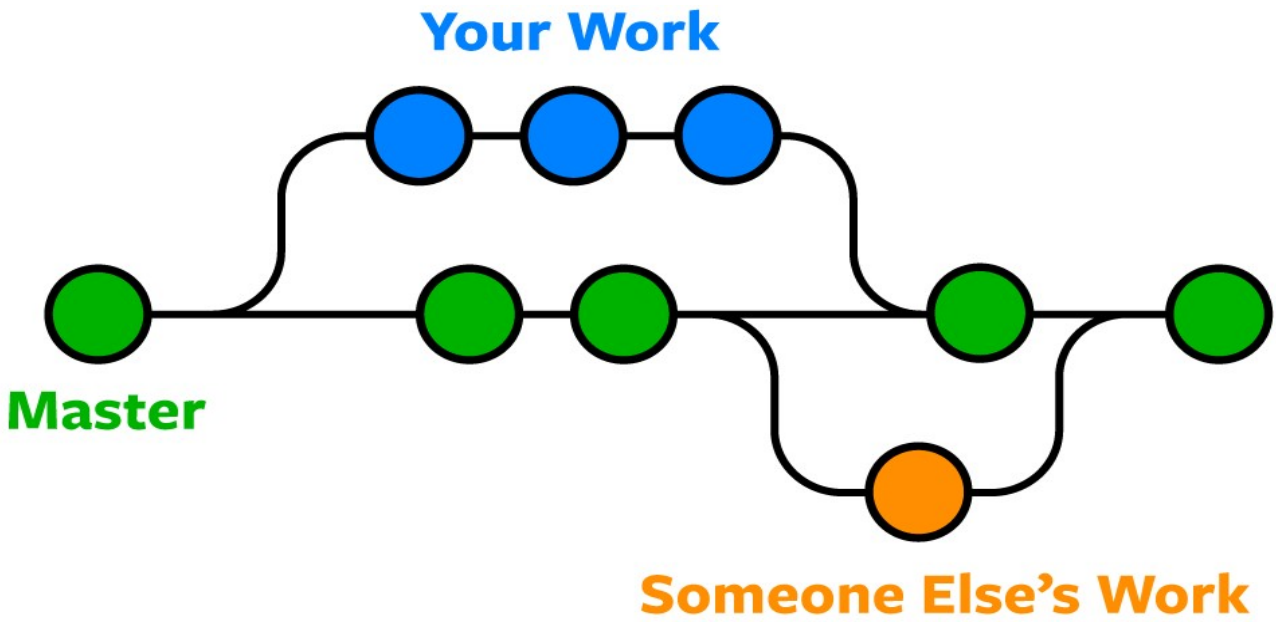
Git log is a utility tool to review and read a history of everything that happens to a repository. Every change committed and the message corresponding to it is recorded and can be seen in the git log. **Command:git log**



Experiment No. 04

Aim: To create and visualize branches.

Theory: Nearly every VCS has some form of branching support. Branching means you diverge from the main line of development and continue to do work without messing with that main line.
Branching simply means parallel development.

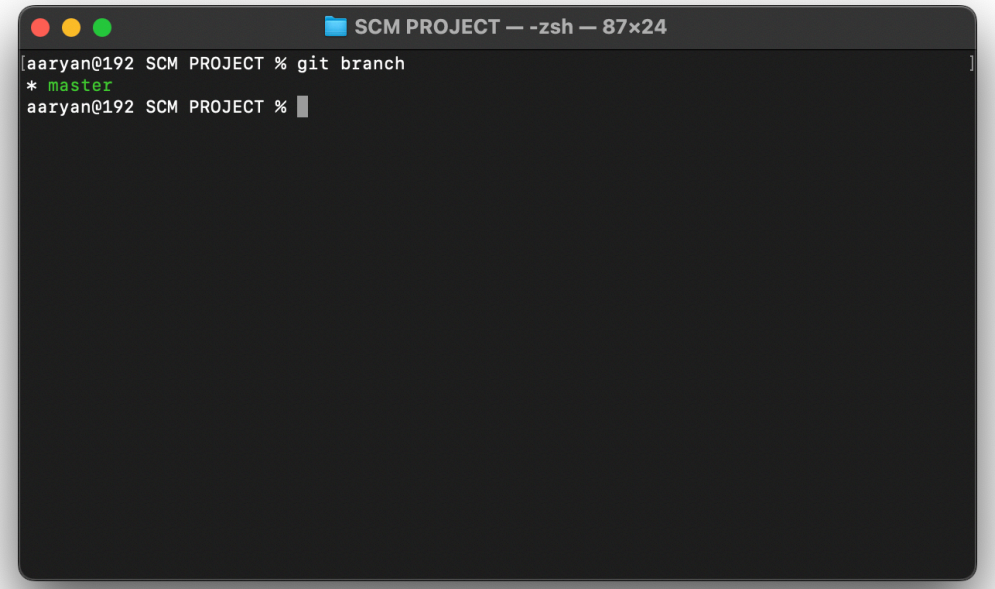


Suppose your team is making a website on the master. You did your work and you think that you can make the website better and you want to do some experiments. Without messing with the website, you can experiment with it using this amazing feature git provides us i.e., branching.

Whenever we create a child branch from a parent branch, all the data will be copied in child branch. But any changed after committing in child will not reflect in parent branch. Similarly, any changes in parent branch after creating child branch, won't be reflected in child branch.

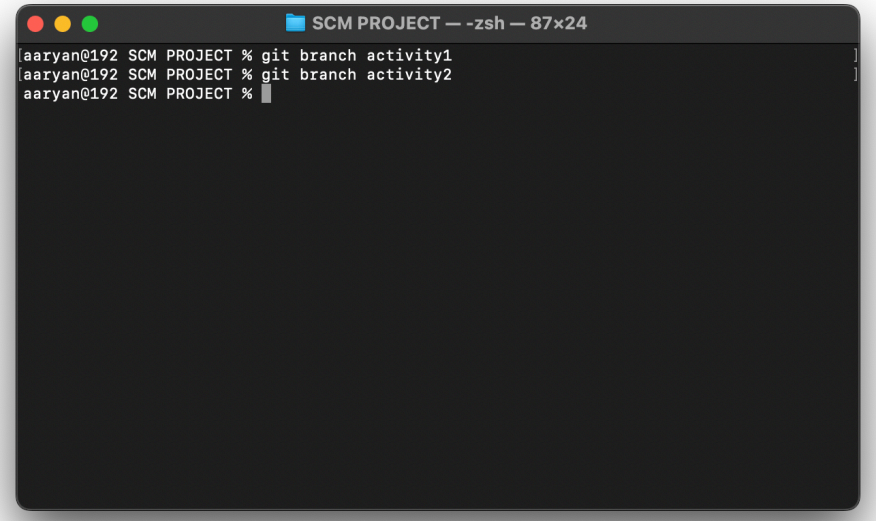
Procedure:

1. Open the folder(**SCM PROJECT**)in terminal and type the command ***git branch***

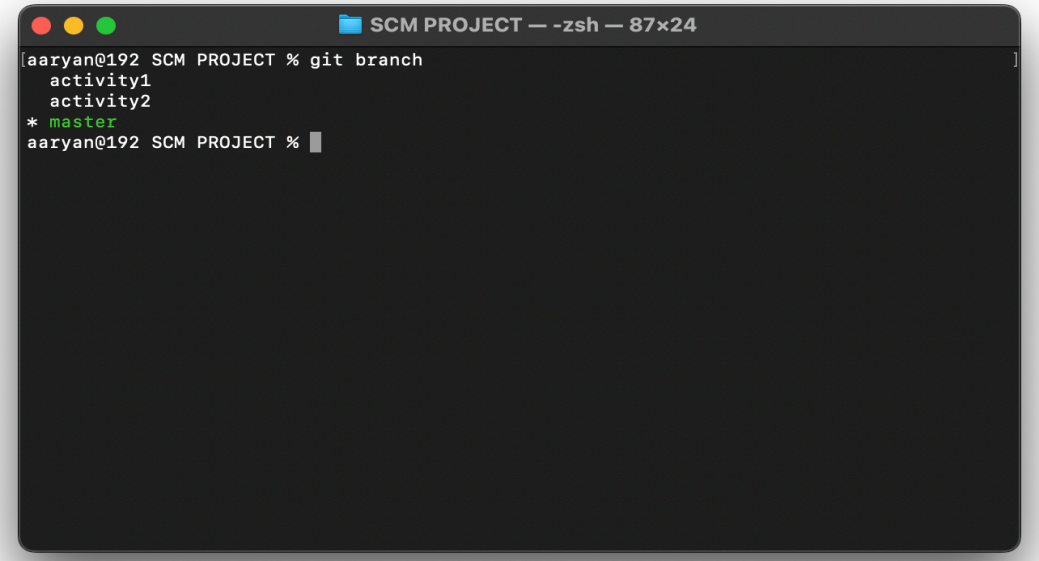


Master branch is the default and main branch. Green colour means that this branch active at the moment. ***git branch*** command shows the list of all the branches.

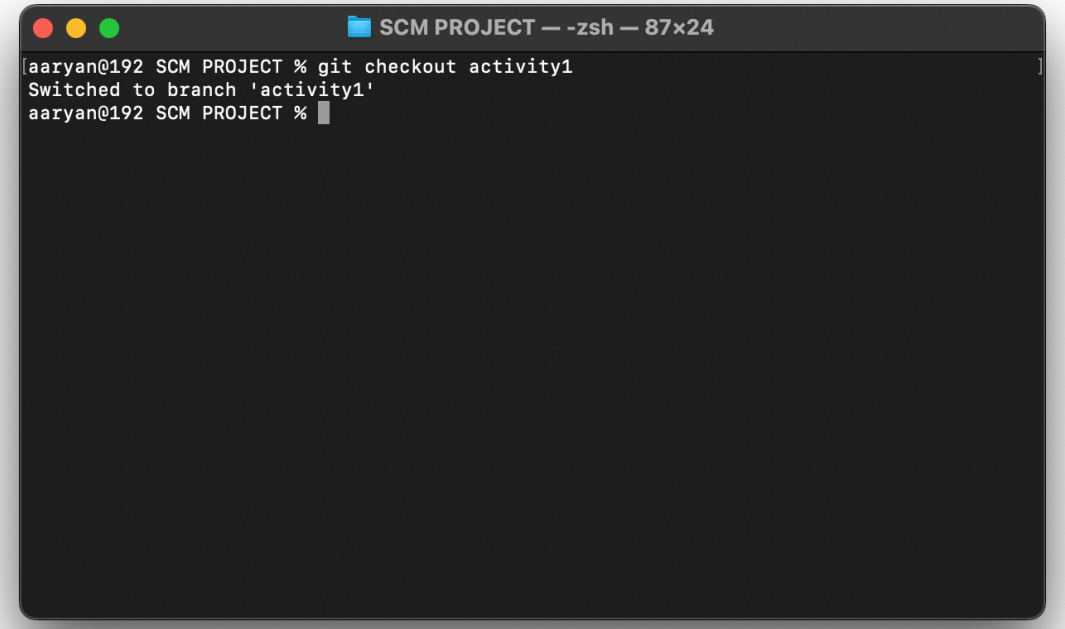
2. To create a new branch from master, use the command ***git branch <Name of the branch>***
   Let's take for example create two branches from master branch ***git branch activity1*** and ***git branch activity2***



3. To see all the branches, use command ***git branch***

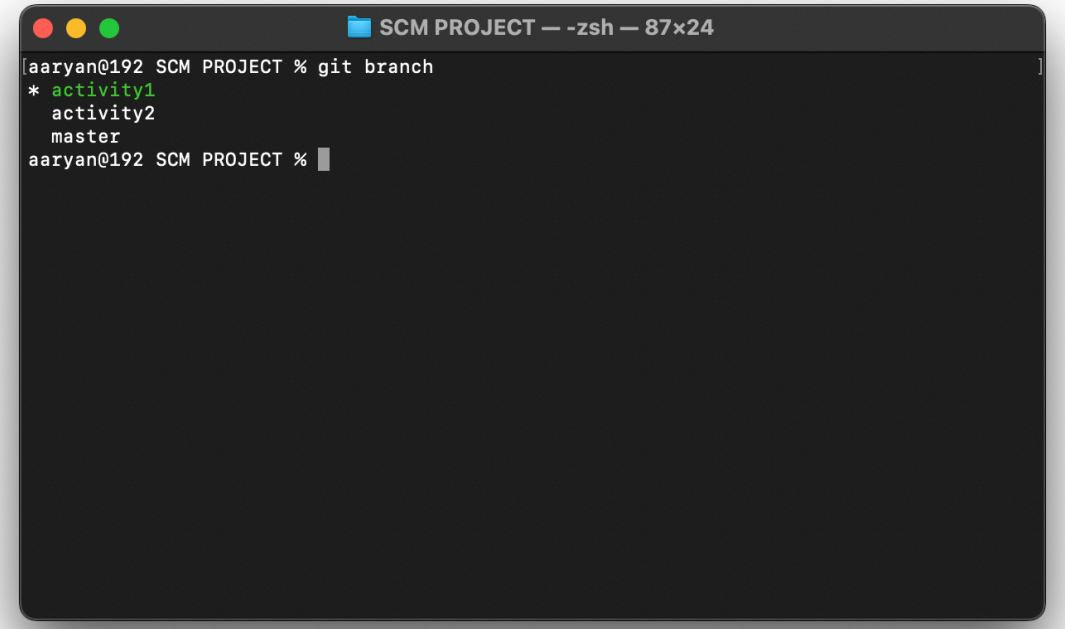


4. Now, to shift or go to the newly created branches use the command ***git checkout <branch name>***



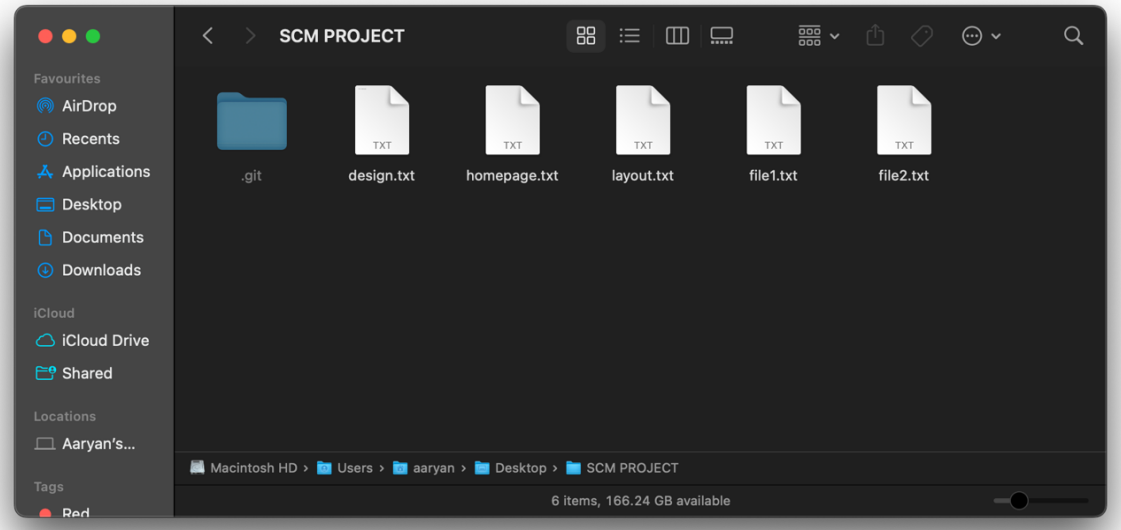
Now, we can clearly see that it is switched to the branch activity1.

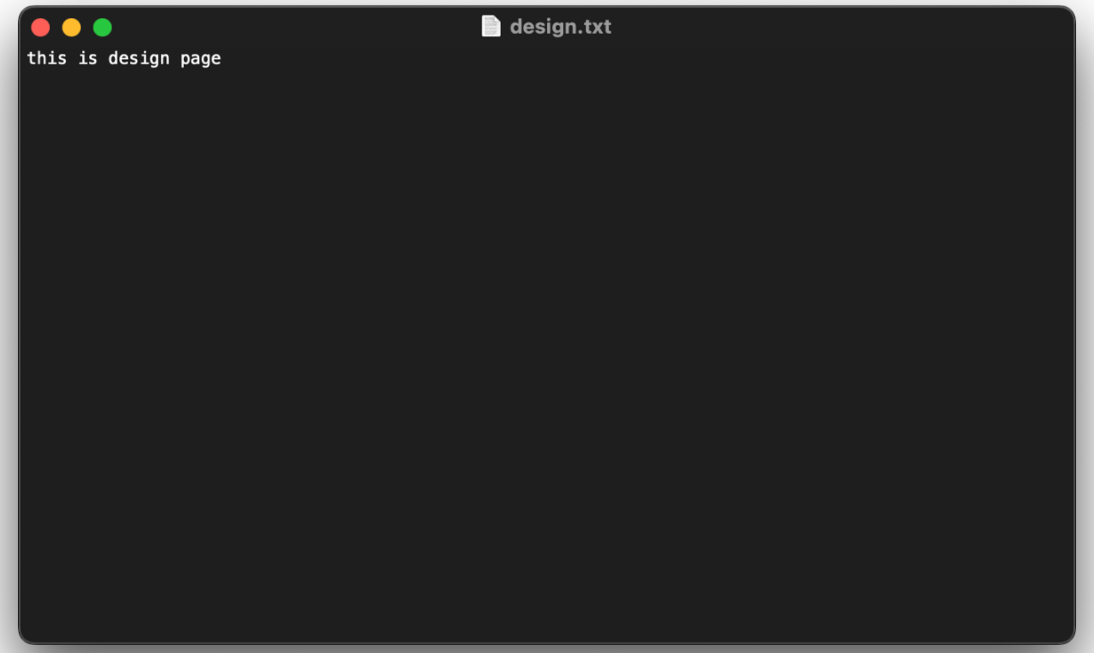
To double verify we can use the command ***git branch***

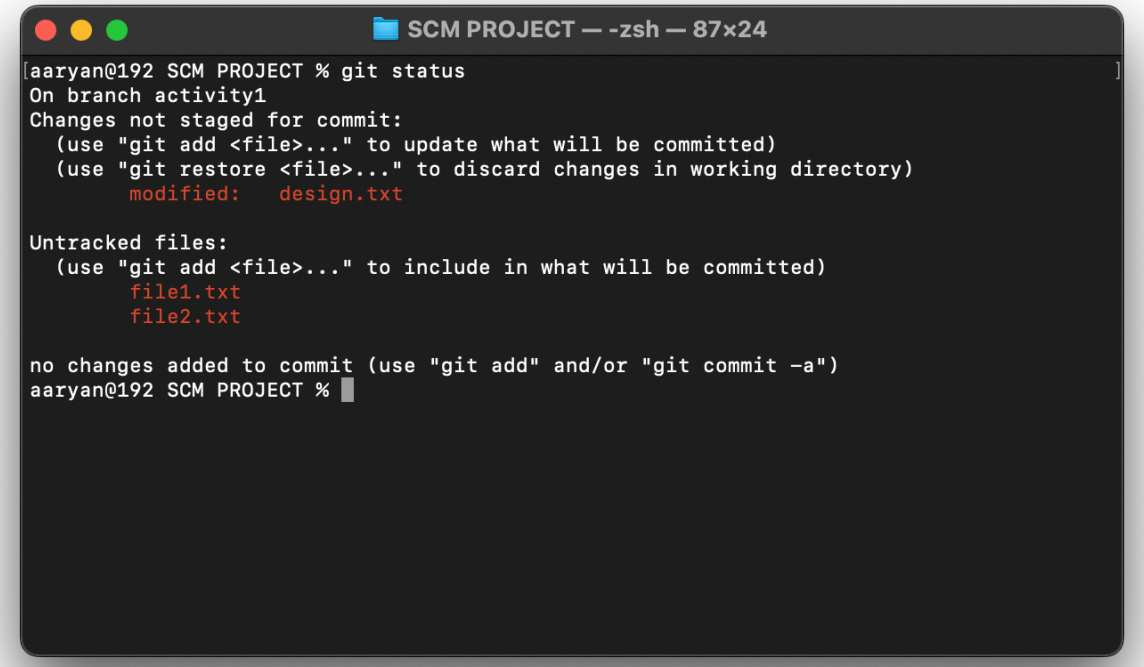


It can be clearly seen that activity has turned green

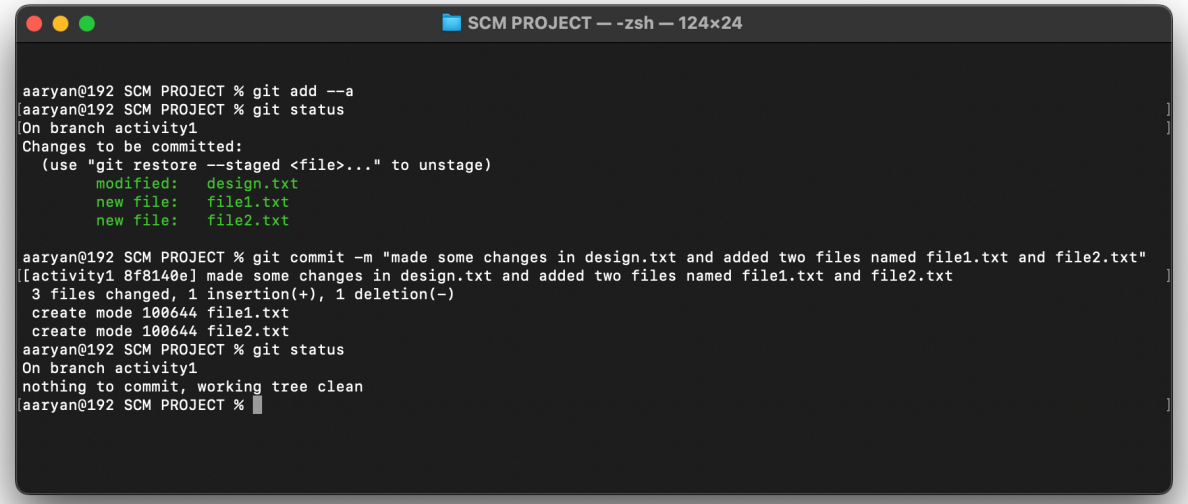
5. Now make some changes in the folder. For example: Make two new txt files named file1.txt and file2.txt
   And make some changes in design.txt



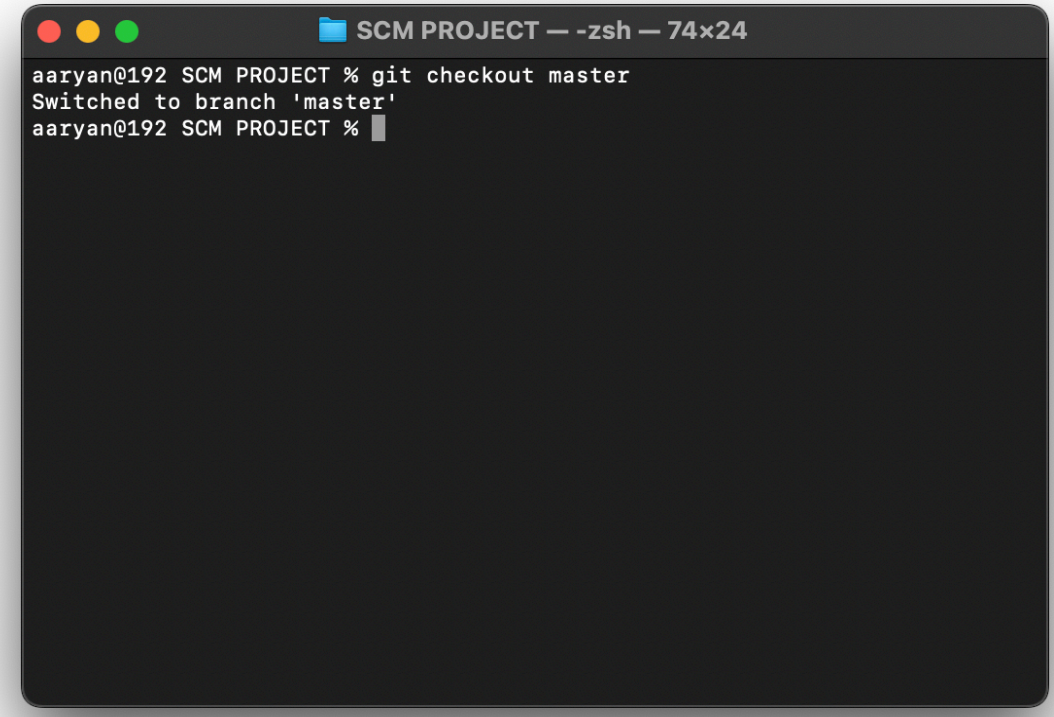

design.txt
this is design page


SCM PROJECT — -zsh — 87×24
aaryan@192 SCM PROJECT % git status
On branch activity1
Changes not staged for commit:
(use "git add <file>..." to update what will be committed)
(use "git restore <file>..." to discard changes in working directory)
modified: design.txt
Untracked files:
(use "git add <file>..." to include in what will be committed)
file1.txt
file2.txt
no changes added to commit (use "git add" and/or "git commit -a")
aaryan@192 SCM PROJECT %

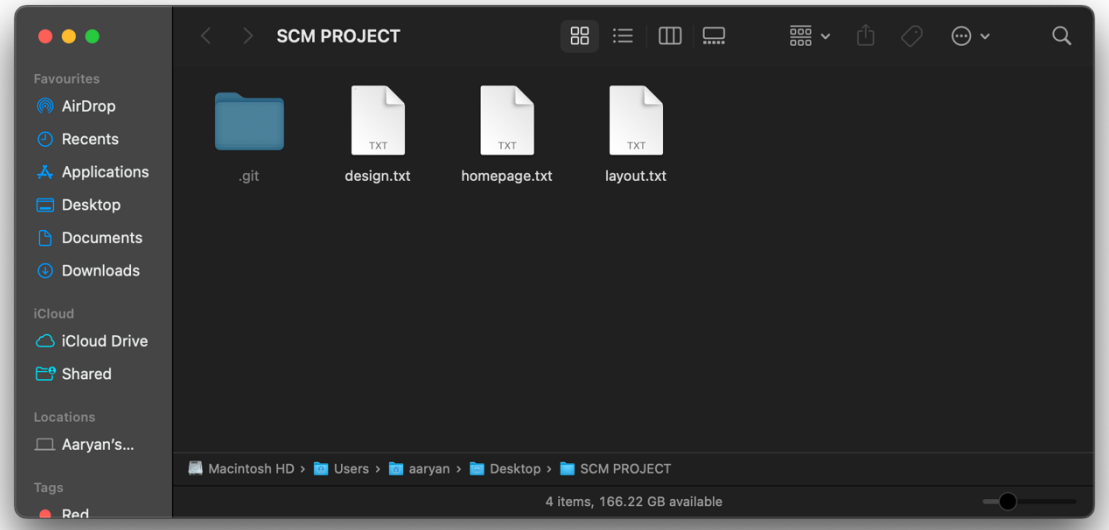
6. Now we have to add the changes or new files to staging area and then commit them.

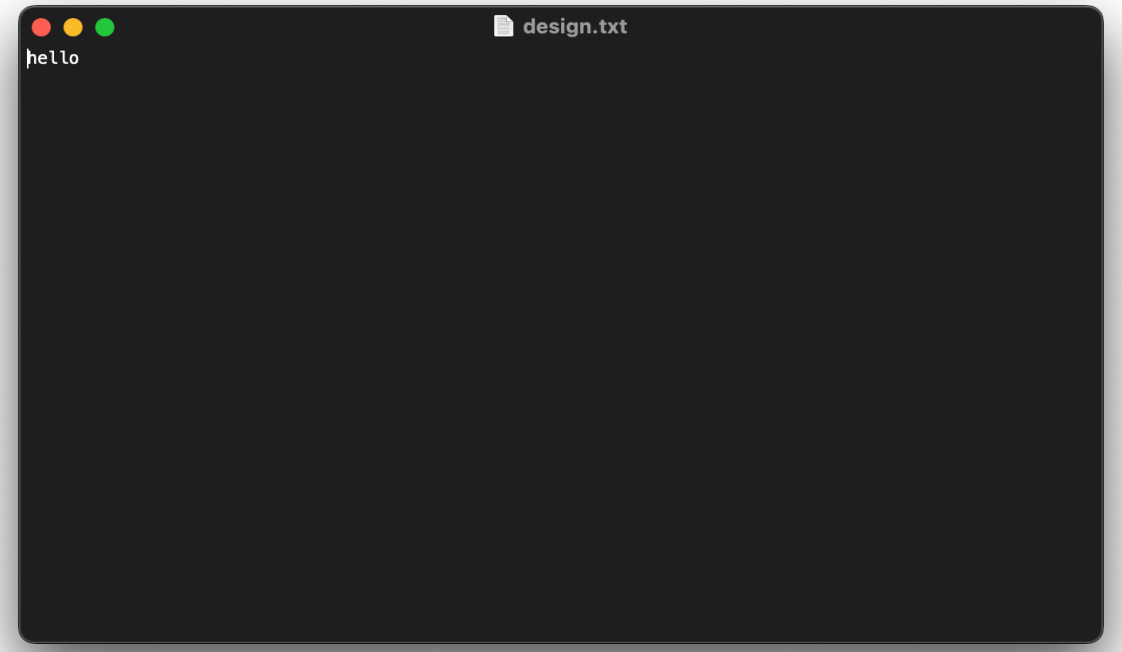


7. Now let's go back to the master branch using the command ***git checkout master***



8. Check the SCM Project folder, there are no file1.txt and file2.txt and design file has not changed like we did in activity1 branch.



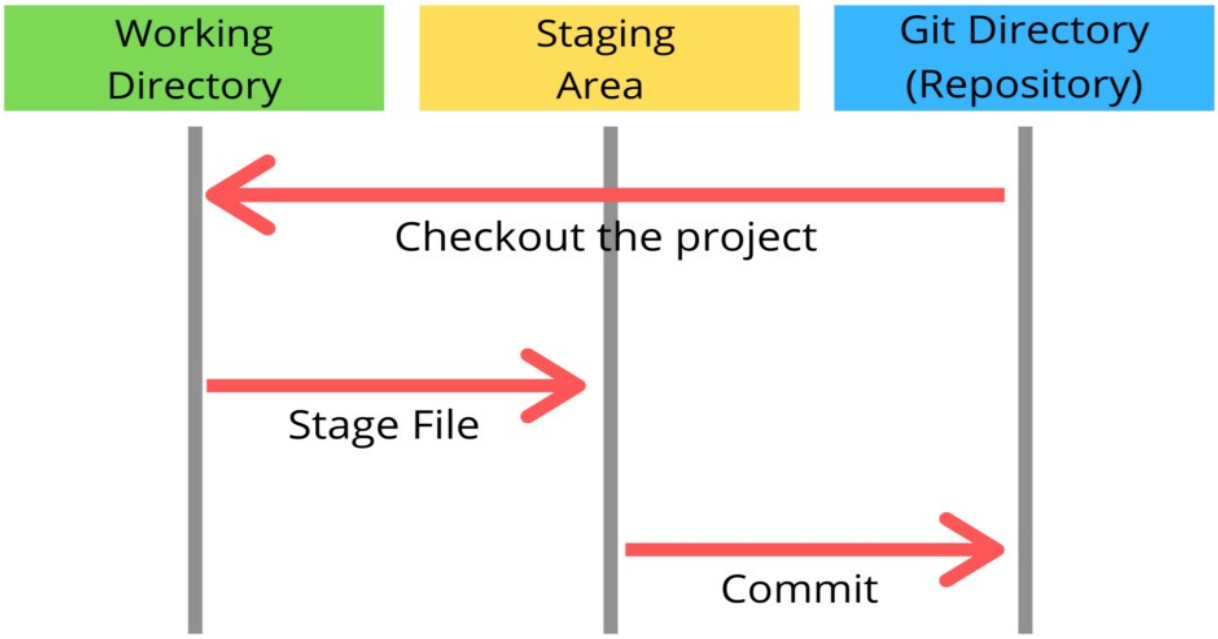


So, this proves that any changes done in child branch will not reflect in parent branch after committing the changes.

Experiment No. 05

Aim: To describe Git Lifecycle.

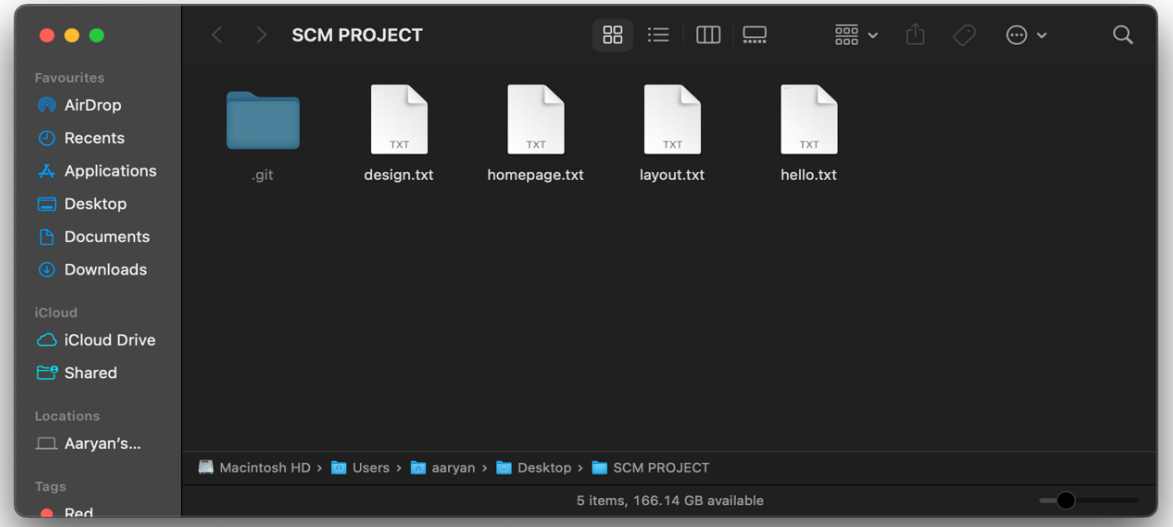
Many VCS's use a two-stage architecture i.e., a repository and a working copy. Git uses three-stage architecture i.e., a working directory, staging area and local repository.
Working Directory is the folder which we created i.e., SCM Project.
Working Directory was converted into Repository where we used ***git init*** command to initial the repository.



Working directory specifies the file explorer's folder where your files are stored, Staging area is an area where those files are present which you want to send to commit (to create snapshot of files). After commit is fired, files which are in staging area will move to Git Repository.

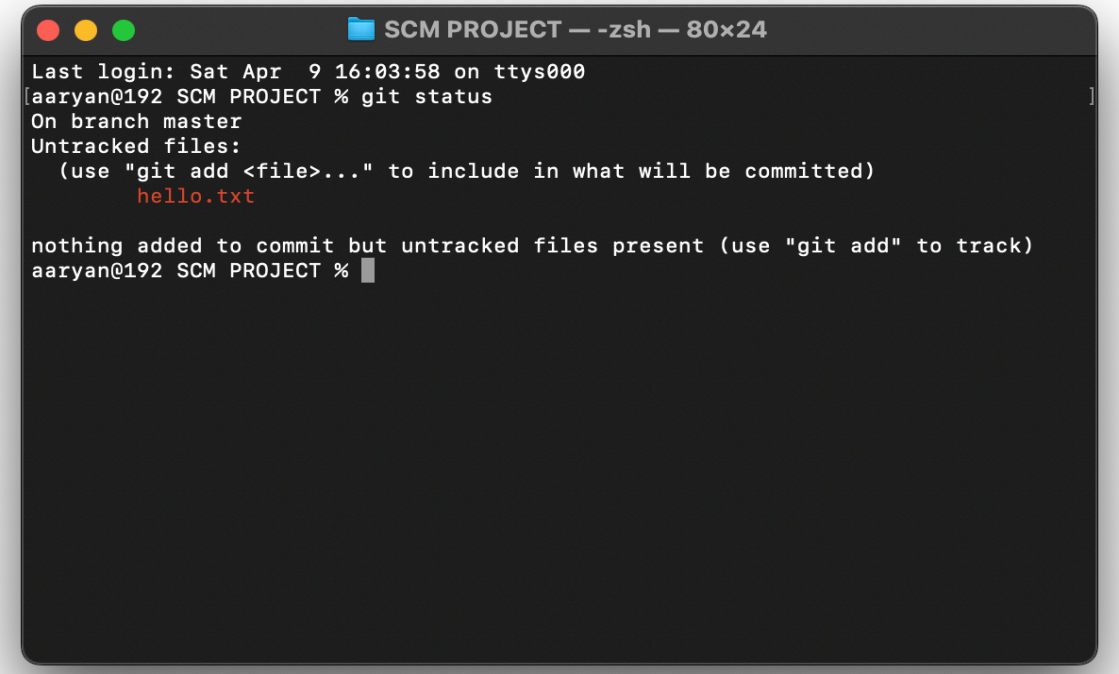
Now if you made any changes in your files which are in Git repository, those have to be added into the Staging area and Committed.

1. Create a file in the Git Repository (SCM Project folder). Let's create a .txt file and name it as hello.txt.



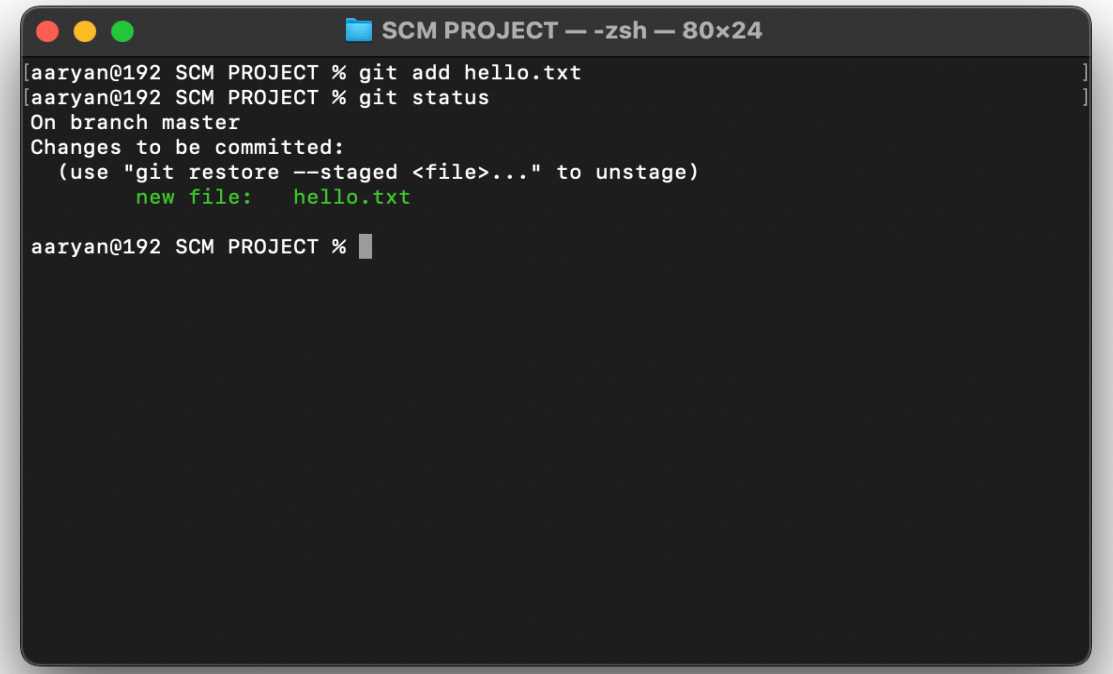
2. Now, open Git Bash and type command *git status*

We can clearly see that the **hello.txt** file is in Untracked files and it is coloured in red.



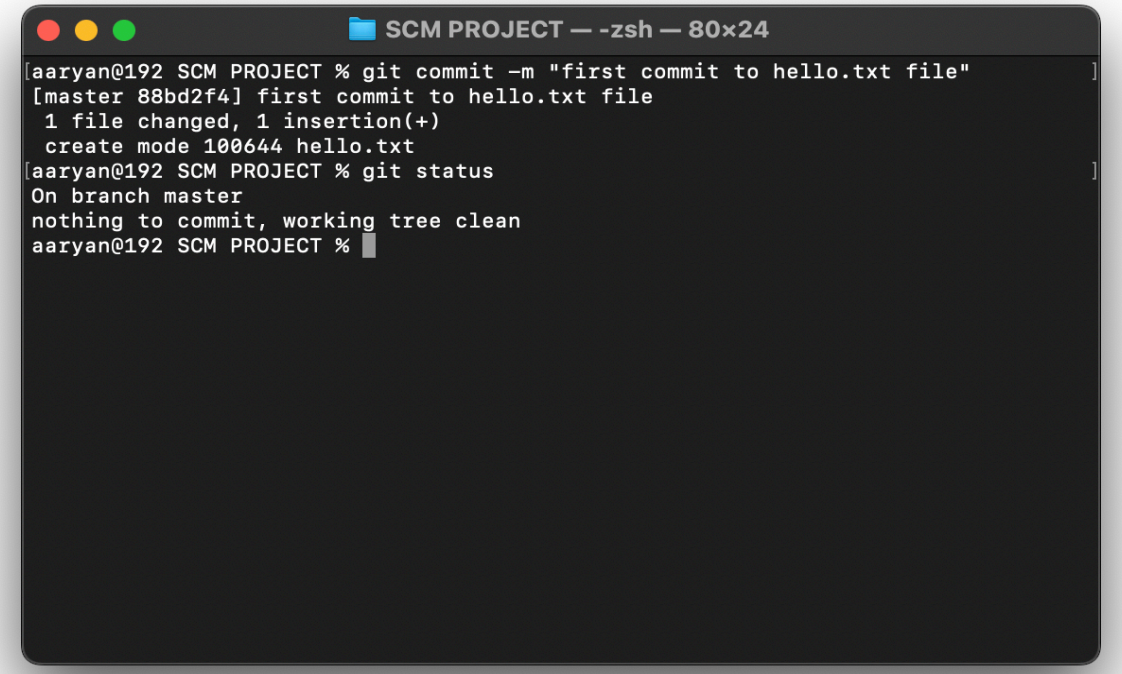
3. Now type ***git add --a*** which will move or add the files to the Staging area. Here --a means all the latest files changed or added. You can also specify a file like ***git add hello.txt.***

4. After that check status using ***git status***

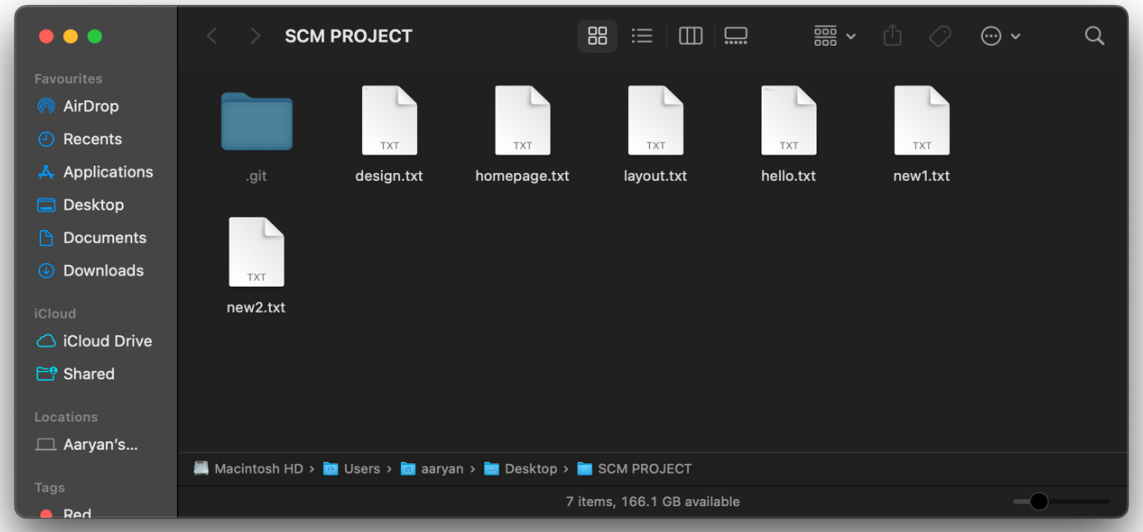


Observe that the **hello.txt** has turned from **red** to **green**. This means that file has been added to the staging area and is ready to be committed.

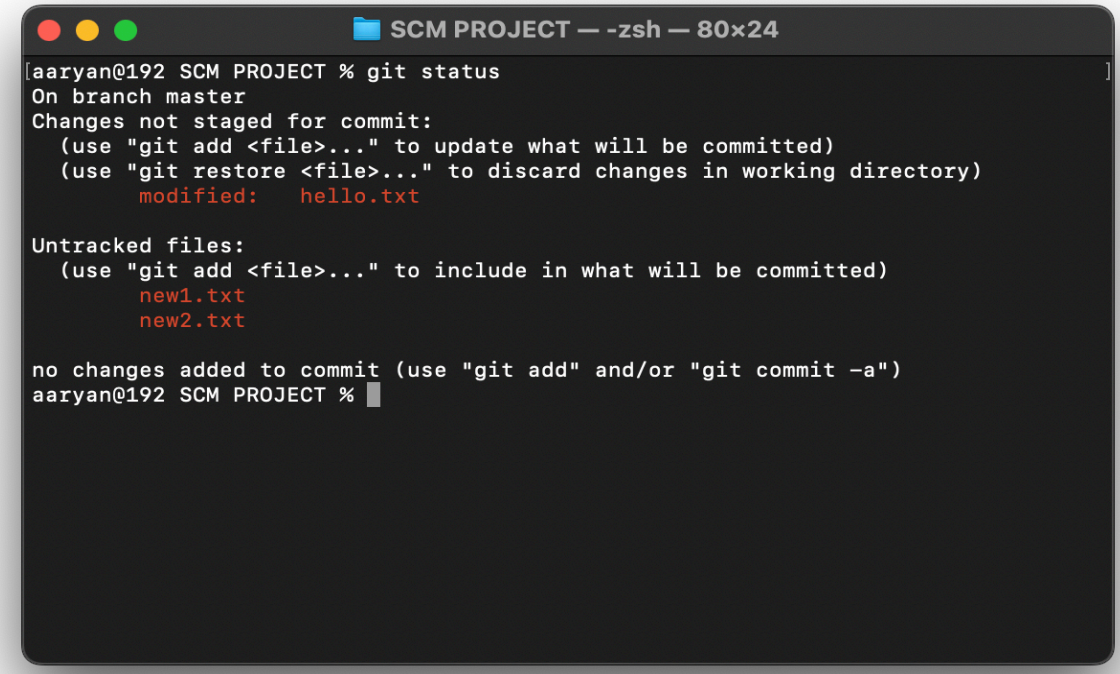
5. Now to commit the files in the staging area, use command ***git commit -m "Any Message"*** and check the status.



6. Now create two new .txt files named **new1.txt** and **new2.txt** and typed something and changed the **hello.txt** file.



7. Now let's check the status. It shows that hello.txt is modified and new1.txt and new2.txt are untracked. All these changes have to be committed.



8. Now we have to commit these changes.