Subject Name: **Source Code Management**

Subject Code: **CS181**

Cluster: **Beta**

Department: **DCSE**


**Submitted By:**
Abhay Kumar
2110990036
G1-A

**Submitted To:**
Dr. Monit Kapoor

CHITKARA UNIVERSITY

# INDEX

# Introduction

## What is GIT and why is it used?

Git is a source code management technology used by DevOps. Git is a piece of software that allows you to track changes in any group of files. It is a free and open-source version control system that may be used to efficiently manage small to big projects. Git is a version control system that allows numerous developers to collaborate on non-linear development projects.
Git is an example of a distributed version control system (DVCS) (hence Distributed Version Control System).

## What is GITHUB?

GitHub is a version management and collaboration tool for programming. It allows you and others to collaborate on projects from any location.

## What is the difference between GIT and GITHUB?

| Git | GitHub |
|---|---|
| 1. It is a software | 1. It is a service |
| 2. It is installed locally on the system | 2. It is hosted on Web |
| 3. It is a command line tool | 3. It provides a graphical interface |
| 4. It is a tool to manage different versions of edits, made to files in a git repository | 4. It is a space to upload a copy of the Git repository |
| 5. It provides functionalities like Version Control System Source Code Management | 5. It provides functionalities of Git like VCS, Source Code Management as well as adding few of its own features |

## What is Repository?

A repository stores all of your project's files, as well as the revision history for each one. Within the repository, you may discuss and monitor your project's progress. The.git/ subdirectory within a project is a Git repository. This repository keeps track of any changes made to files in your project over time, creating a history. That is, if you delete the.git/ subdirectory, you are also deleting the history of your project.

## What is Version Control System (VCS)?

Version Control Systems are the software tools for tracking/managing all the changes made to the source code during the project development. It keeps a record of every single change made to the code. It also allows us to turn back to the previous version of the code if any mistake is made in the current version. Without a VCS in place, it would not be possible to monitor the development of the project.

## Types of VCS

- Local Version Control System
- Centralized Version Control System
- Distributed Version Control System

I. **Local Version Control System:** Local Version Control System is located in your local machine. If the local machine crashes, it would not be possible to retrieve the files, and all the information will be lost. If anything happens to a single version, all the versions made after that will be lost.

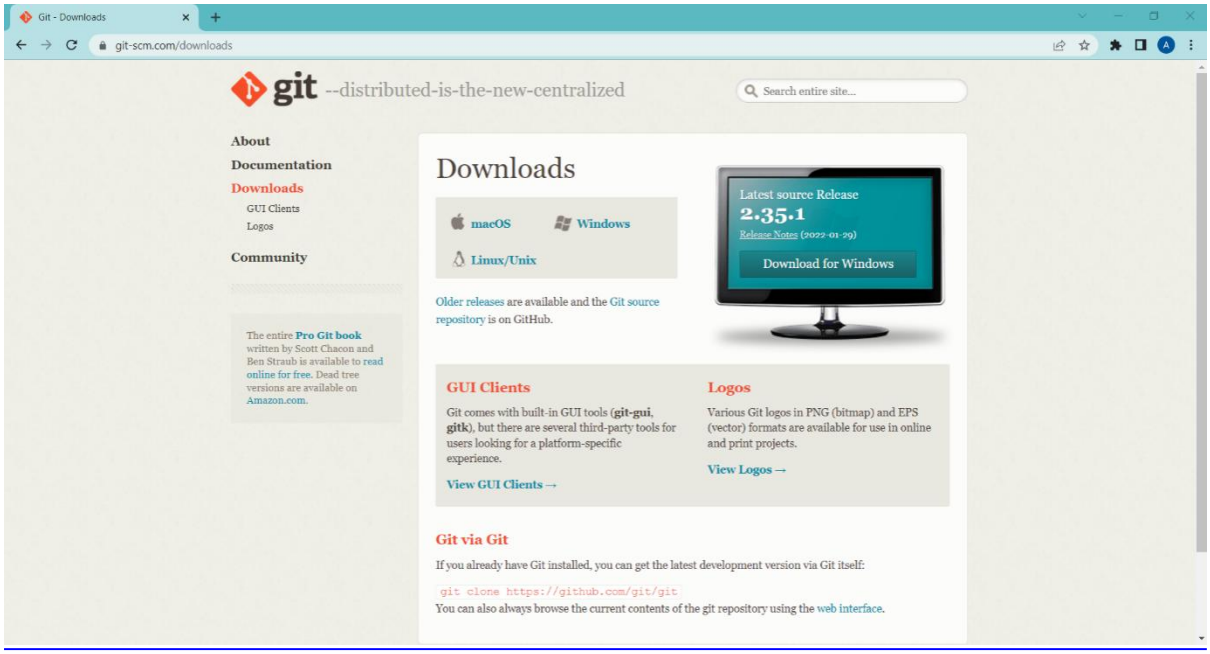
II. **Centralized Version Control System:** In the Centralized Version Control Systems, there will be a single central server that contains all the files related to the project, and many collaborators checkout files from this single server (you will only have a working copy). The problem with the Centralized Version Control Systems is if the central server crashes, almost everything related to the project will be lost.

III. **Distributed Version Control System:** In a distributed version control system, there will be one or more servers and many collaborators similar to the centralized system. But the difference is, not only do they check out the latest version, but each collaborator will have an exact copy of the main repository on their local machines. Each user has their own repository and a working copy. This is very useful because even if the server crashes we would not lose everything as several copies are residing in several other computers.

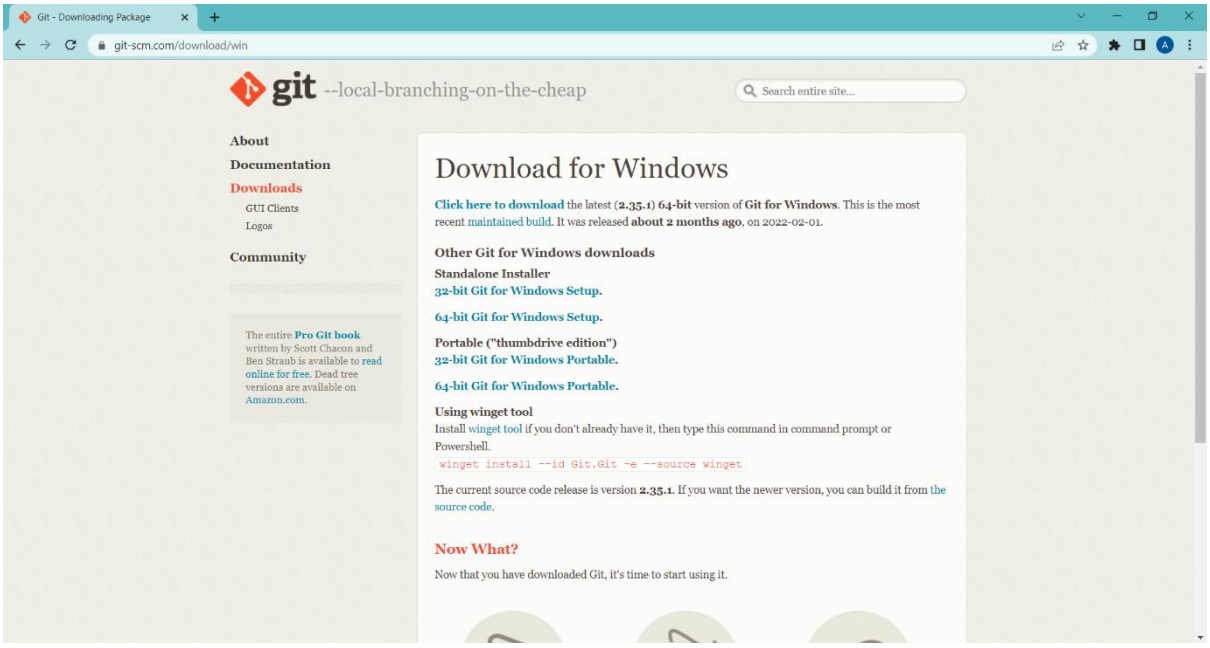
# Experiment No. 01

**Aim:** Setting up of Git Client

✧ For git installation on your system, go to the linked URL.
https://git-scm.com/downloads




✧ You must first access this webpage and then choose your operating system by clicking on it. I'll walk you through the processes for the Windows operating system in this article.



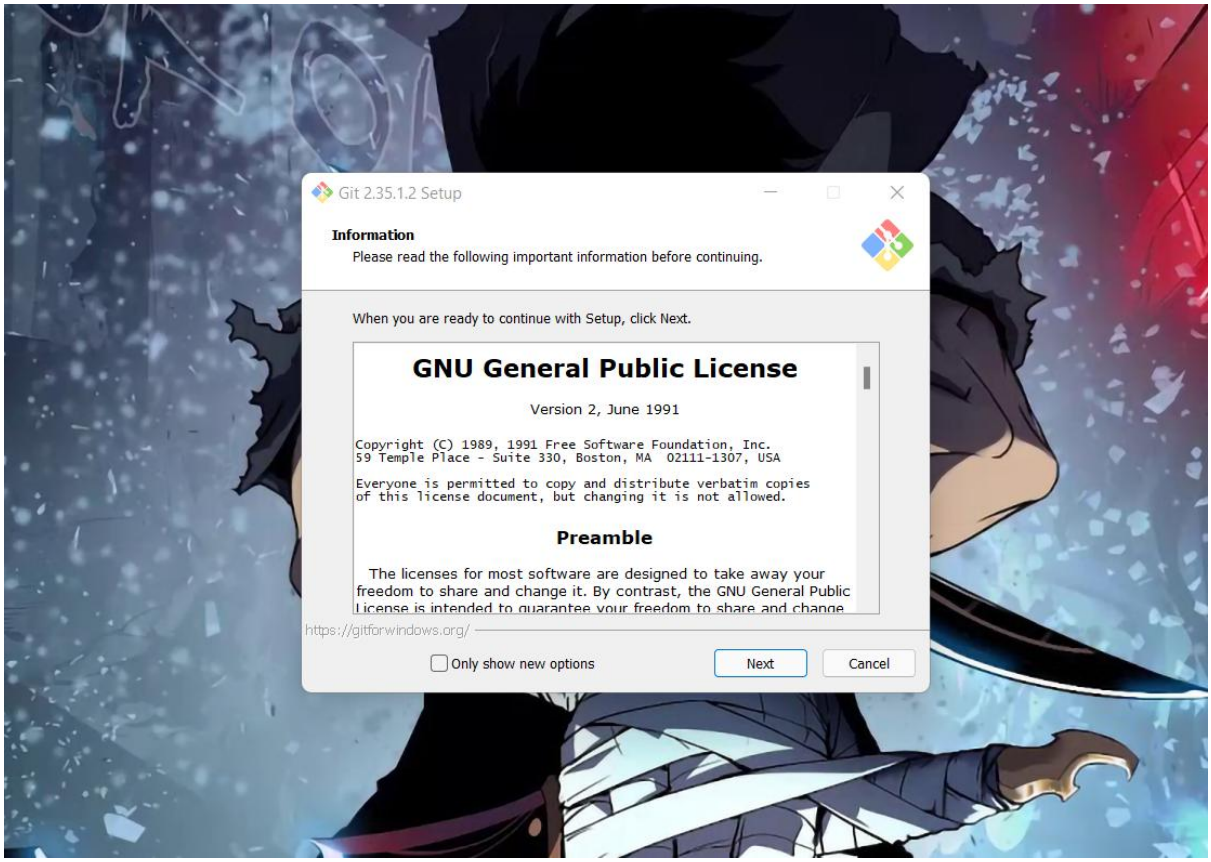

✧ Select the CPU for your system now. (Most of the system now runs on

64-bit processors.) Your download will begin when you pick a processor.

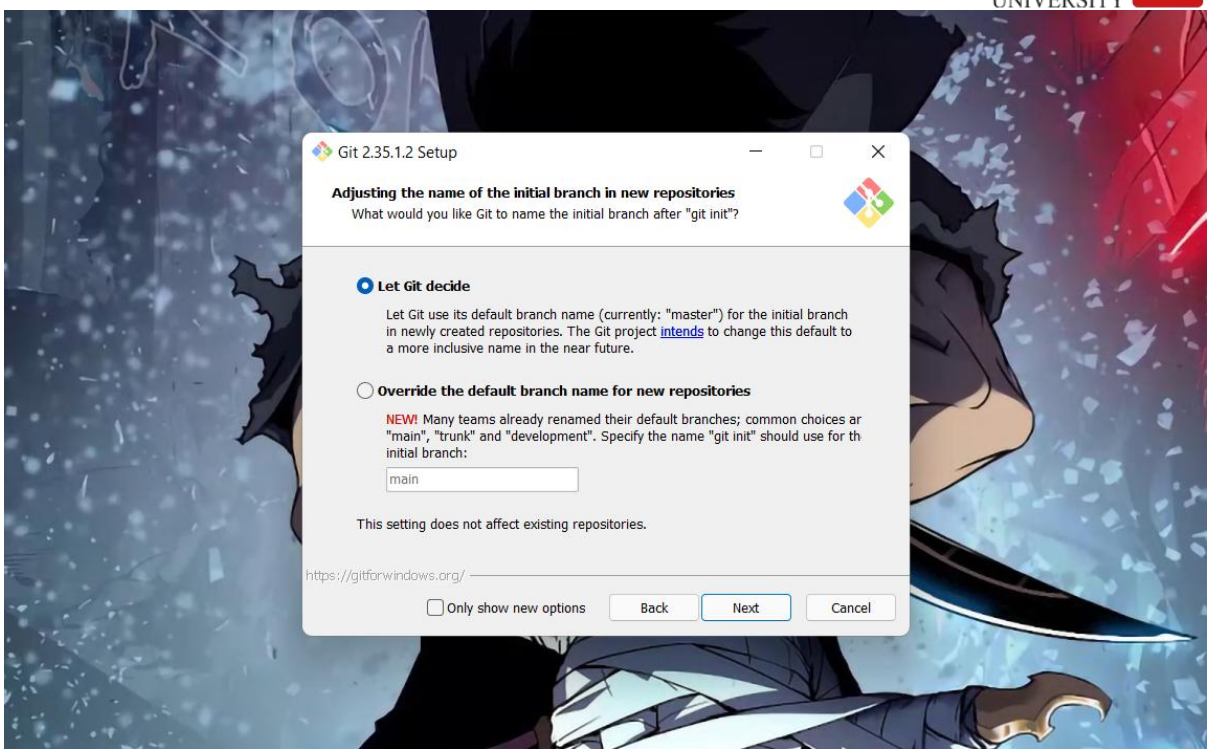

✧ You must now open the Git folder.

✧ You will be asked if you want to enable this programe to make modifications to your PC once you launch it.

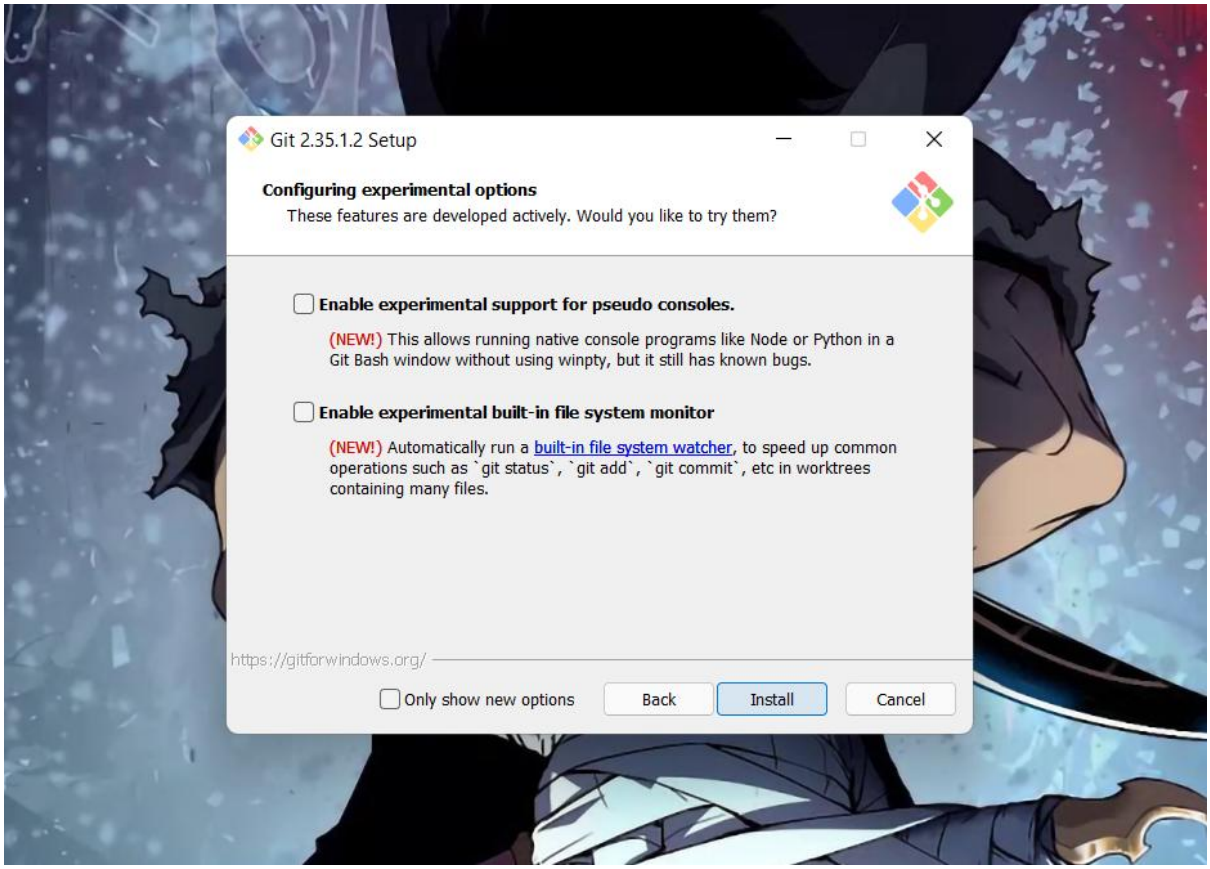
✧ YES should be selected.



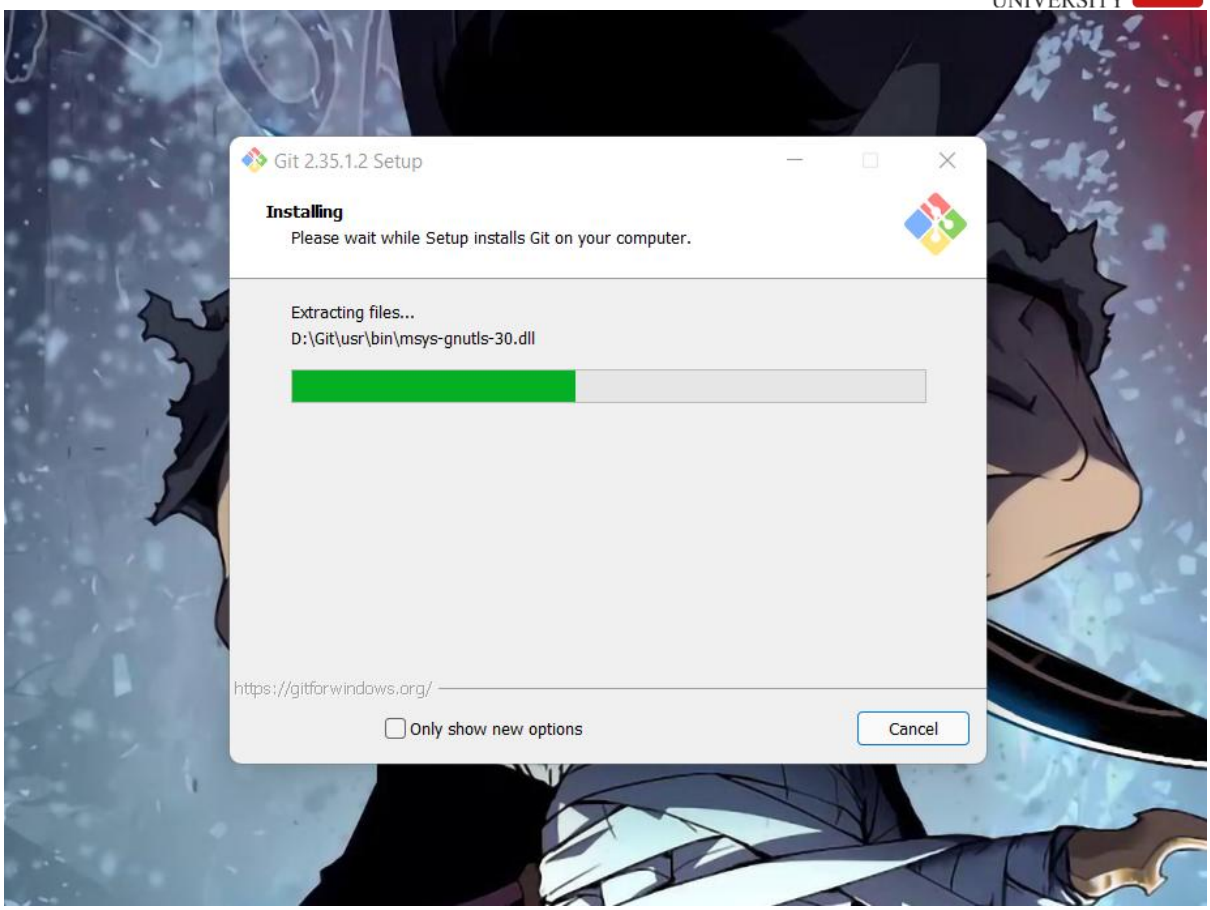

✧ Click on Next




✧ Continue clicking on next few times more




✧ Now select the Install option.




✧ Click on Finish after the installation is finished.

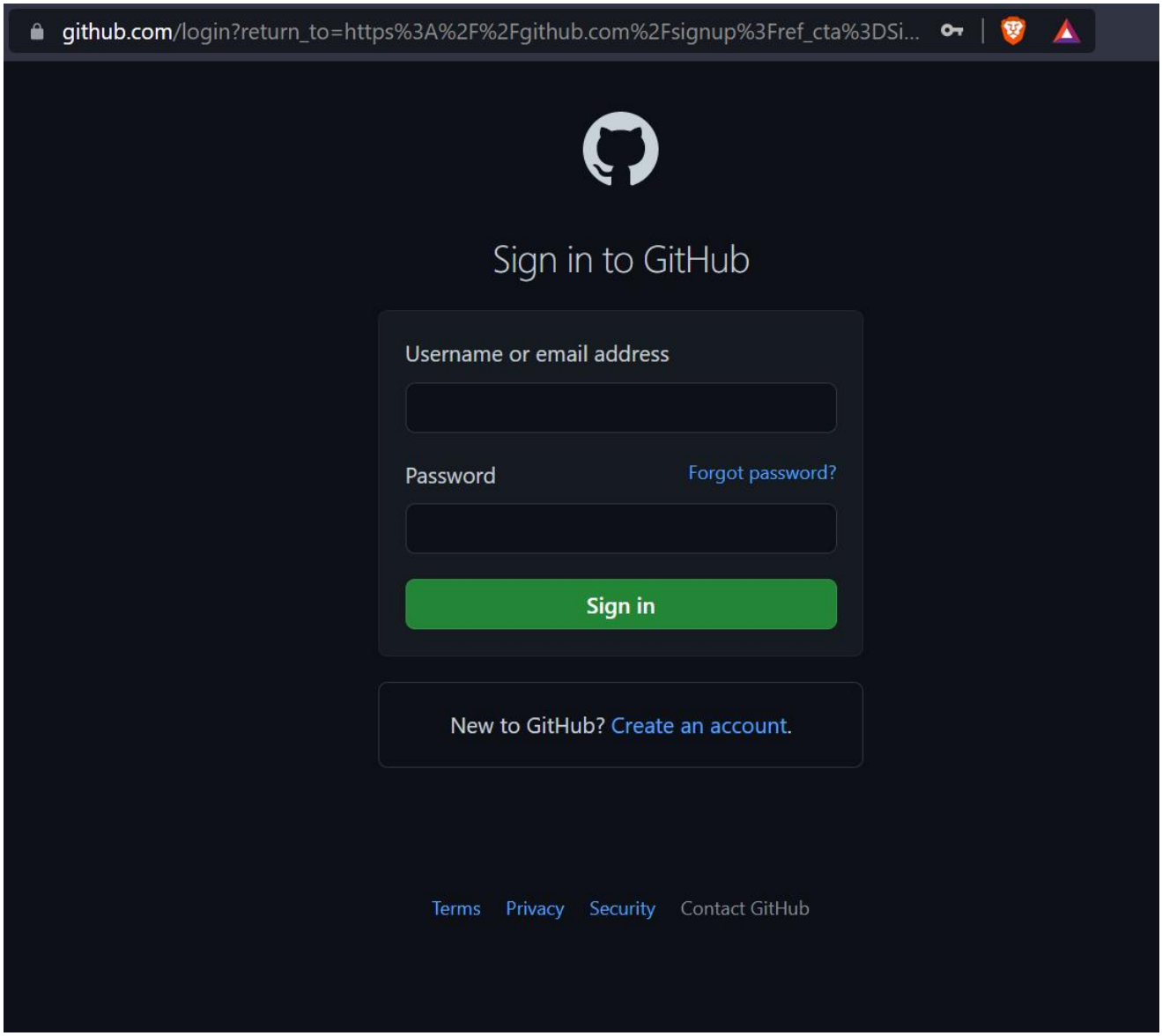
The installation of the git is finished and now we have to setup git client and GitHub account.

# Experiment No. 02

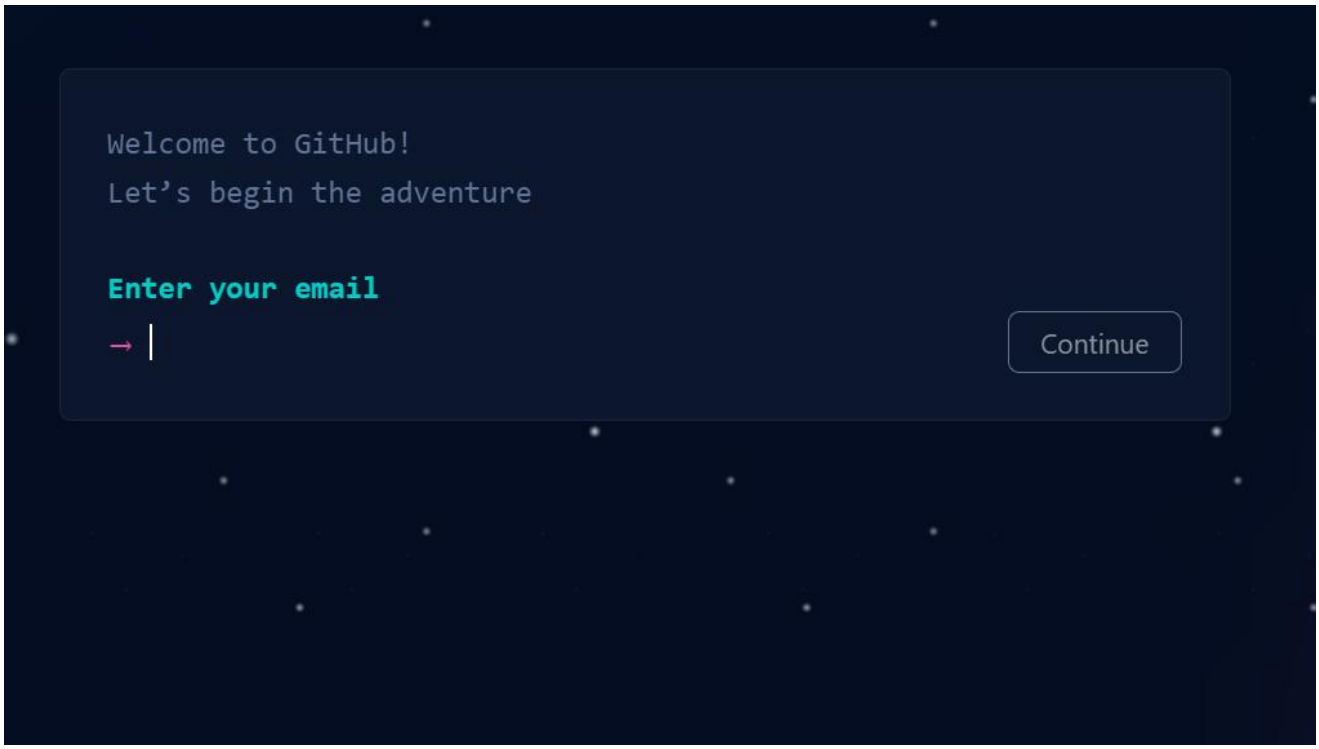
**Aim: Setting up GitHub Account**

✧ Open your web browser search GitHub login.

✧ Click on Create an account if you are a new user or if you have already an account, please login.




✧ After clicking on "Create a New Account," you will be sent to a new page where you must enter your email address for your account. Now type in the password you'd want to use for your GitHub account. Then you'll be prompted to enter your username.



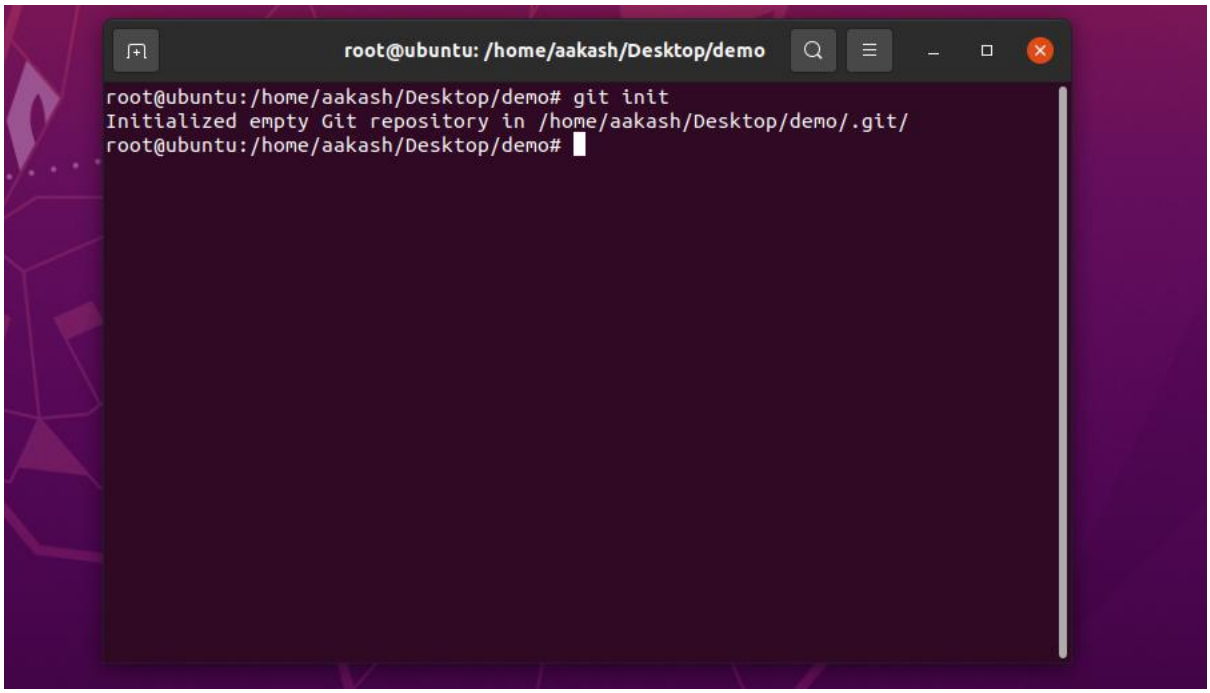

✧ Now Click on Create Account.

✧ Verify it from your email and you are all set to go.

## Experiment No. 03

**Aim:** Program to Generate logs

✧ First of all create a local repository using Git. For this, you have to make a folder in your device, right click and select "**Git Bash Here**". This opens the Git terminal. To create a new local repository, use the command "**git init**" and it creates a folder **.git**.




✧ When we use GIT for the first time, we have to give the user name and email so that if I am going to change in project, it will be visible to all.

For this, we use command →

"git config --global user.name *Name*"
"git config --global user.email *email*"

For verifying the user's name and email, we use →

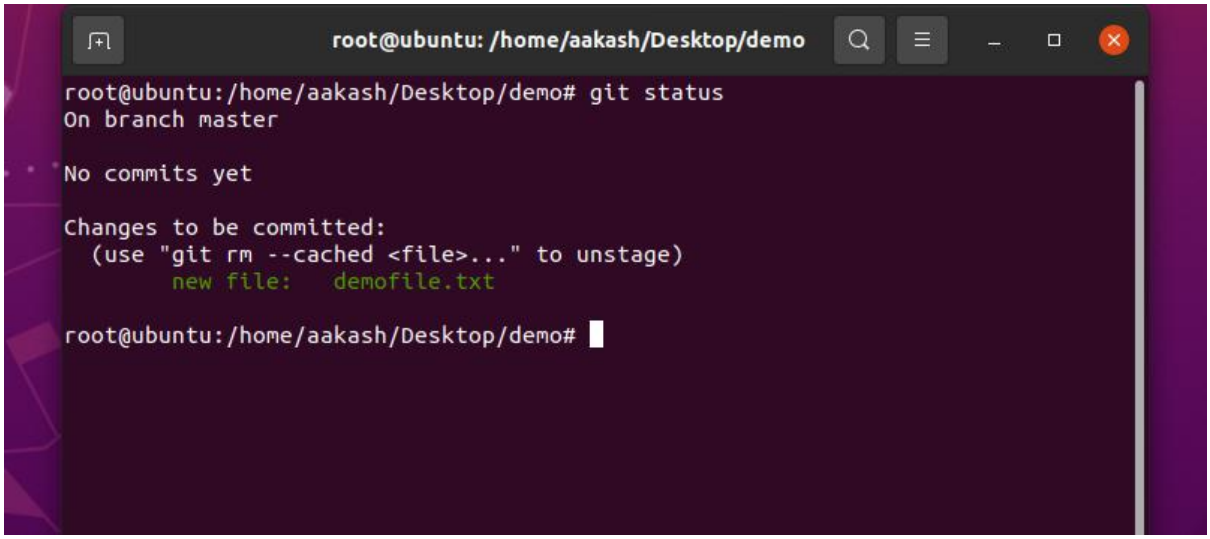
"git config --global user.name"
"git config --global user.email"

## Some Important Commands:

- **ls** → It gives the file names in the folder.
- **ls -lart** → Gives the hidden files also.
- **git status** → Displays the state of the working directory and the staged snapshot.

- **touch filename →** This command creates a new file in the repository.
- **Clear →** It clears the terminal.
- **rm -rf .git →** It removes the repository.
- **git log →** displays all of the commits in a repository's history
- **git diff →** It compares my working tree to staging area.

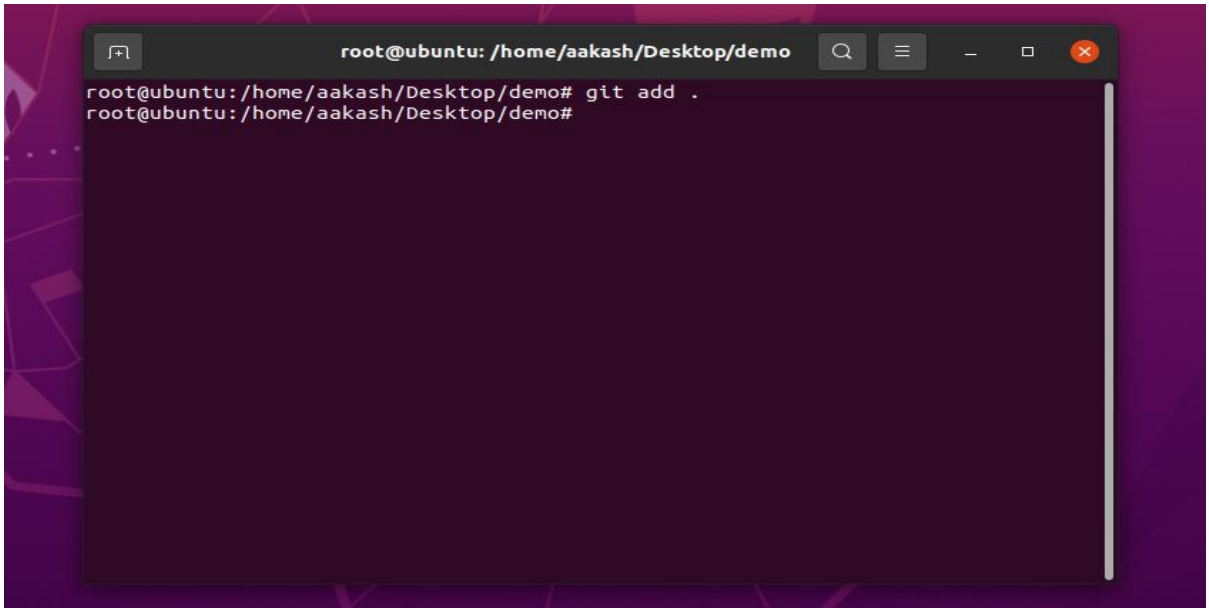
Now, we have to create some files in the repository. Suppose we created index.html Now type git status:

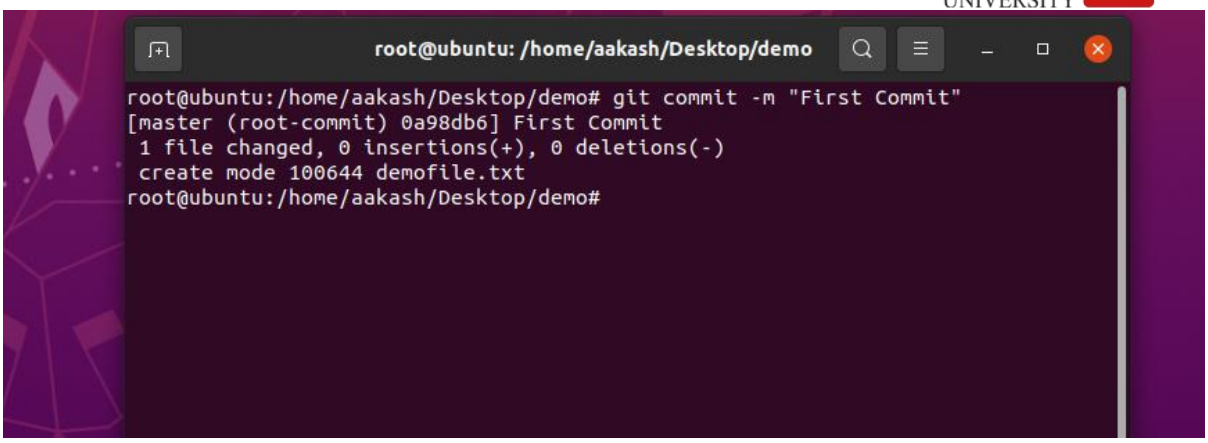



You can see that *index.html* is in red colour that means it is an untracked file.
Now firstly add the file in staging area and then commit the file.

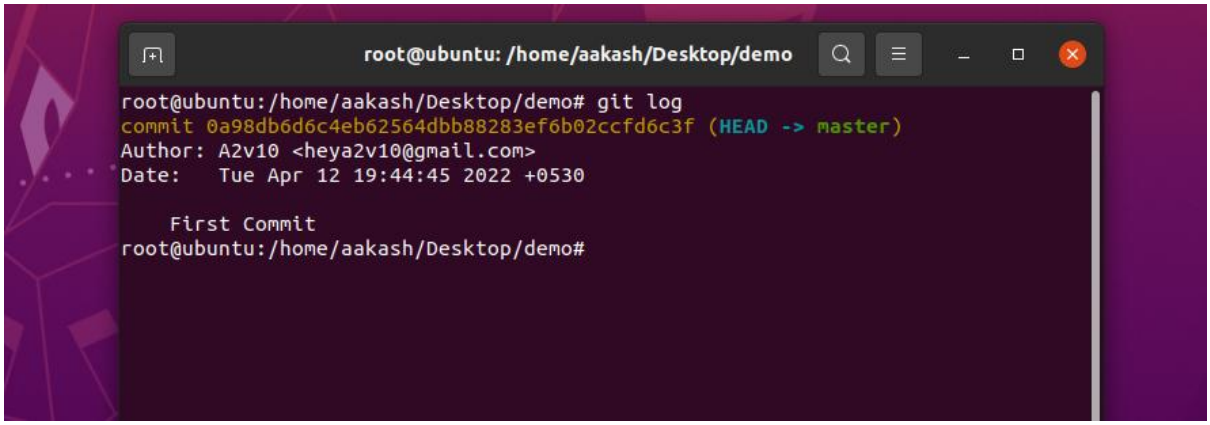
For this, use command →

**git add -A** [ For add all the files in staging area. ]
**git commit -m "write any message"** [ For commit the file ]





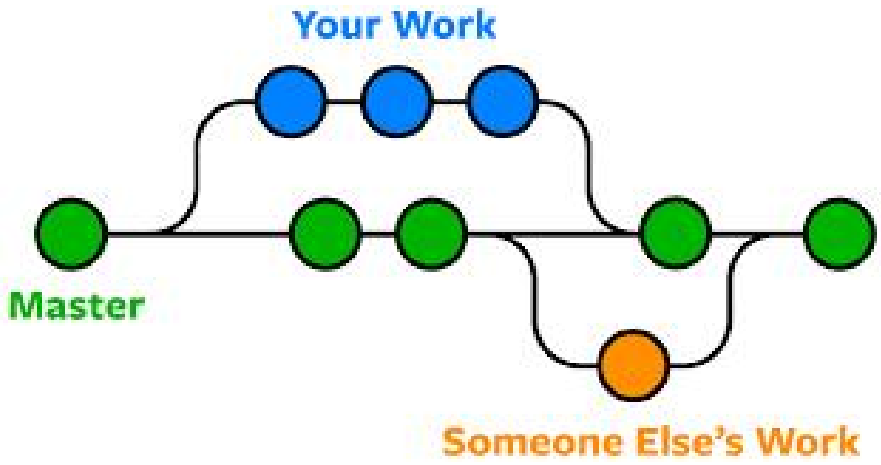

✧ git log: *The git log command displays a record of the commits in a Git repository. By default, the git log command displays a commit hash, the commit message, and other commit metadata.*



## Experiment No. 04

**Aim:** Create and visualize branches

✧ **Branching:** A branch in Git is an independent line of work(a pointer to a specific commit). It allows users to create a branch from the original code (master branch) and isolate their work. Branches allow you to work on different parts of a project without impacting the main branch.




Let us see the command of it:
Firstly, add a new branch, let us suppose the branch name is activity1.
For this use command →

- **git branch name** [adding new branch]
- **git branch** [use to see the branch's names]
- **git checkout *branch name*** [use to switch to the given branch]

```
root@ubuntu:/home/aakash/Desktop/demo# git branch
* master
root@ubuntu:/home/aakash/Desktop/demo# git branch act1
root@ubuntu:/home/aakash/Desktop/demo# git branch
  act1
* master
root@ubuntu:/home/aakash/Desktop/demo#
```

In this you can see that firstly ‘git branch’ shows only one branch in green colour but when we add a new branch using ‘git branch act1’, it shows 2 branches but the green colour and star is on master. So, we have to switch to act1 by using ‘git checkout act1’. If we use ‘git branch’, now you can see that the green colour and star is on act1. It means you are in activity1 branch and all the data of master branch is also on act1 branch. Use “ls” to see the files.

Now add a new file in activity1 branch, do some changes in file and commit the file.

If we switched to master branch, ‘contact.html’ file is not there. But he file is in activity1 branch.

- To add these files in master branch, we have to do merging. For this firstly switch to master branch and then use command →

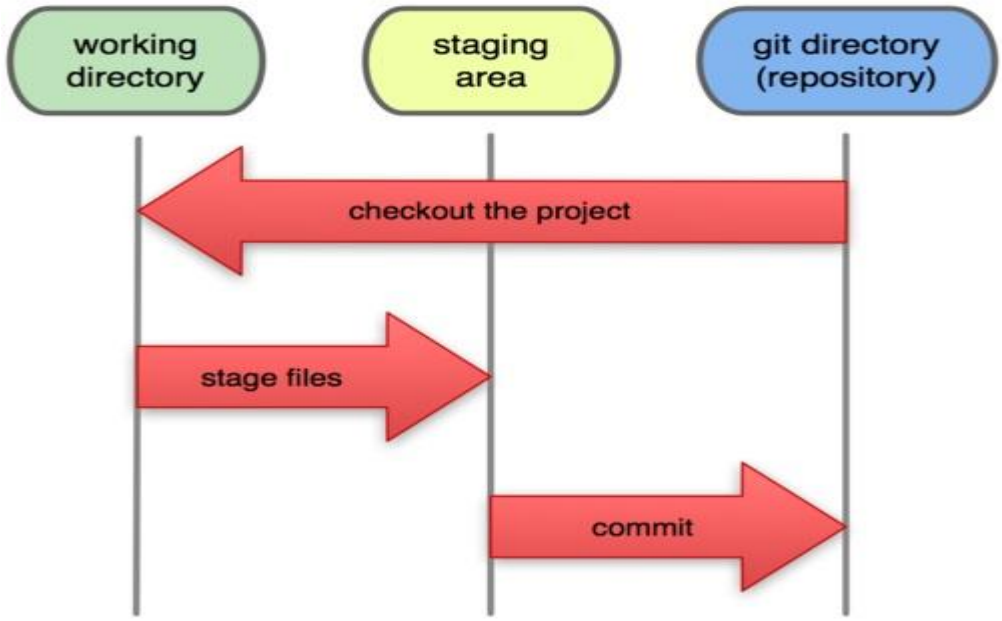
  **git merge branchname** [use to merge branch]

## Experiment No. 05

**Aim:** Git lifecycle description

**Stages in GIT Life Cycle:**

Now let's understand the three-stage architecture of Git:




- **Working Directory:** This is the directory that we've initialized, and here all the changes are made to commit on GitHub.

- **Staging Area:** This is where we first put out code or files of the working repository. The command that we use to stage code is, "git add --a", "git add FileName" or "git add -A". In simple terms, staging means telling Git what files we want to commit (new untracked files, modified files, or deleted files).

- **Git directory(repository):** This is where all the commits are stored whenever we make a commit. We can revert to an older version of or project using the "git checkout" command from this directory.

# Source Code Management

# Task 1.2 and 2

(CS181)

Submitted by

Name-Abhay Kumar  Roll No.- 2110990036


**Department of Computer Science & Engineering**

Chitkara University Institute of Engineering and Technology, Punjab

Jan- June
(2021-22)

CHITKARA UNIVERSITY

| | | | |
|---|---|---|---|
| Institute/School Name | **Chitkara University Institute of Engineering and Technology** | | |
| Department Name | **Department of Computer Science & Engineering** | | |
| Programme Name | **Bachelor of Engineering (B.E.), Computer Science & Engineering** | | |
| Course Name | **Source Code Management** | Session | **2021-22** |
| Course Code | **CS181** | Semester/Batch | **2nd/2021** |
| Vertical Name | **Beta** | Group No | G01 |
| Course Coordinator | **Dr. Neeraj Singla** | | |
| Faculty Name | **Mr. Monit Kapoor** | | |

CHITKARA UNIVERSITY

## INDEX

CHITKARA UNIVERSITY

# ***ADD COLLABORATORS ON GITHUB REPO***

In GitHub, we can invite other GitHub users to become collaborators to our private repositories (which expires after 7 days if not accepted, restoring any unclaimed licenses). Being a collaborator, of a personal repository you can pull (read) the contents of the repository and push (write) changes to the repository. You can add unlimited collaborators on public and private repositories.

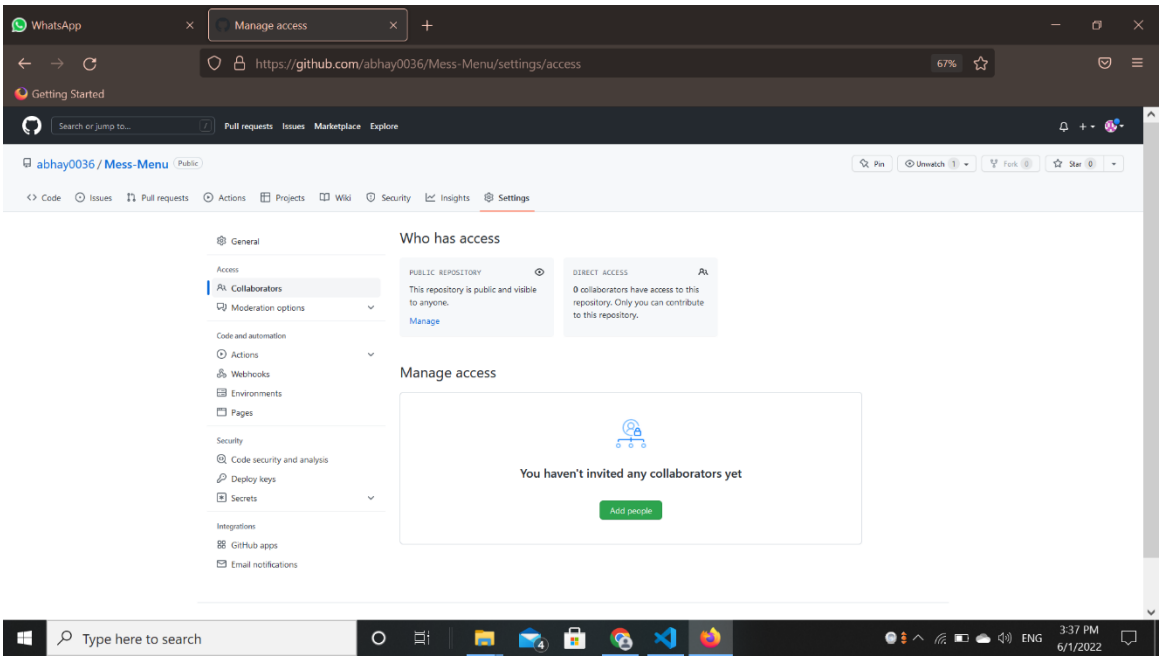
Collaborators can perform a number of actions into someone else's personal repositories, they have gained access to. Some of them are,

1. Create, merge, and close pull requests in the repository
2. Publish, view, install the packages
3. Fork the repositories
4. Make the changes on the repositories as suggested by the Pull requests.
5. Mark issues or pull requests as duplicate
6. Create, edit, and delete any comments on commits, pull requests, and issues in the repository
7. Removing themselves as collaborators on the repositories.
8. Manage releases in the repositories.

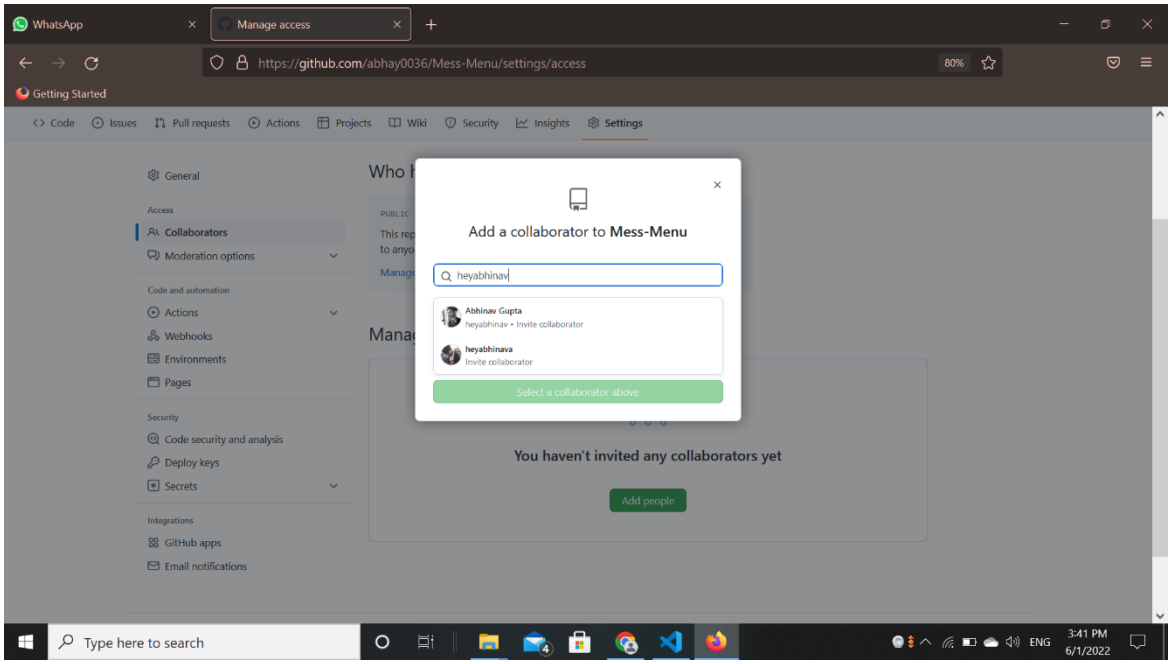
STEPS TO ADD COLLABORATORS:

1. Navigate to the repository on Github you wish to share with your collaborator.
2. Click on the "Settings" tab on the right side of the menu at the top of the screen.

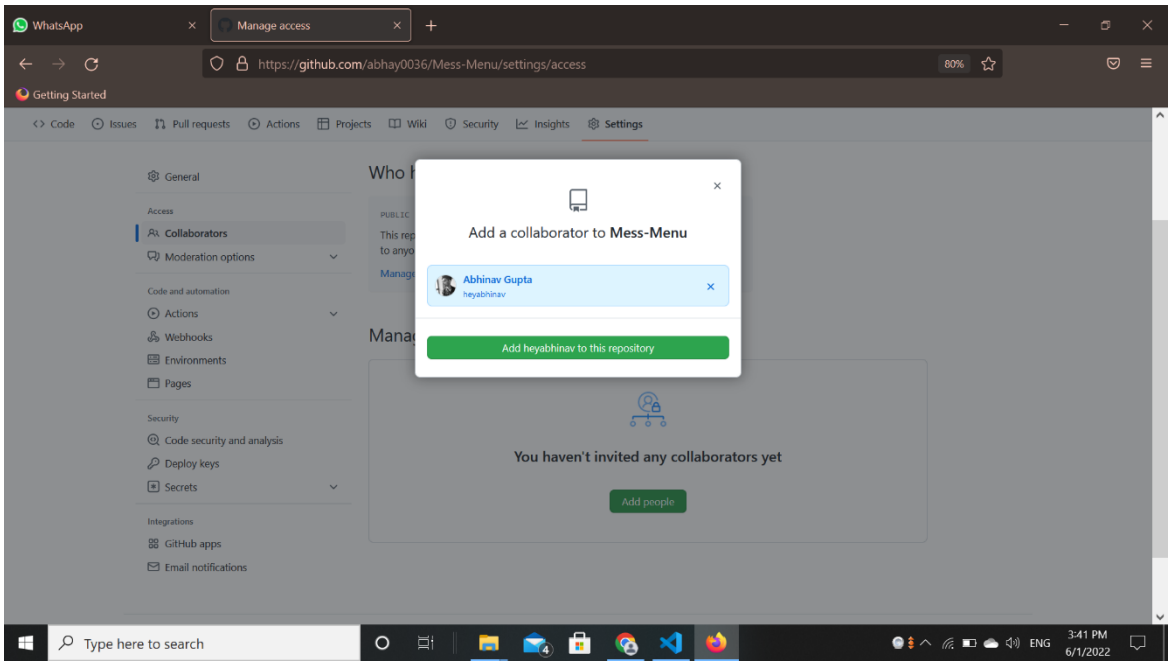
3.On the new page, click the "Collaborators" menu item on the left side of the page.




4. Start typing the new collaborator's GitHub username into the text box.



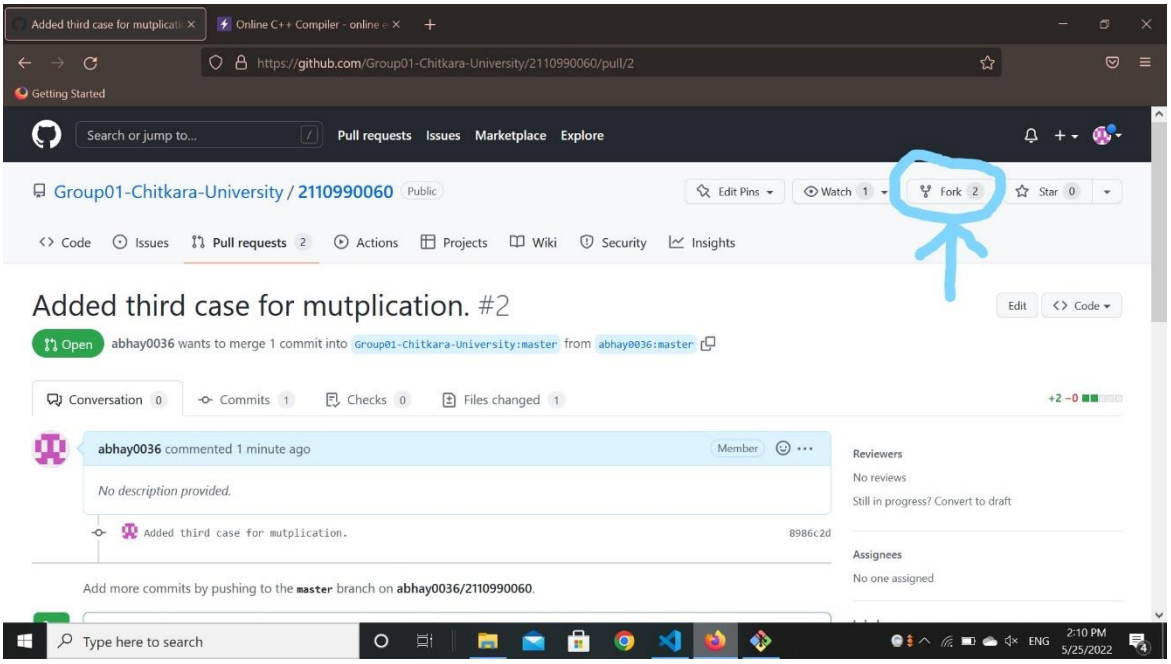

5. Select the GitHub user from the list that appears below the text box.
6. Click the "Add" button.



# FORK AND COMMIT

A fork is a copy of a repository. Forking a repository allows you to freely experiment with changes without affecting the original project. Most commonly, forks are used to either propose changes to someone else's project to which you do not have write access, or to use someone else's project as a starting point for your own idea.

## STEPS TO FORK A REPO-




1. Go to the repository that you wish to fork.

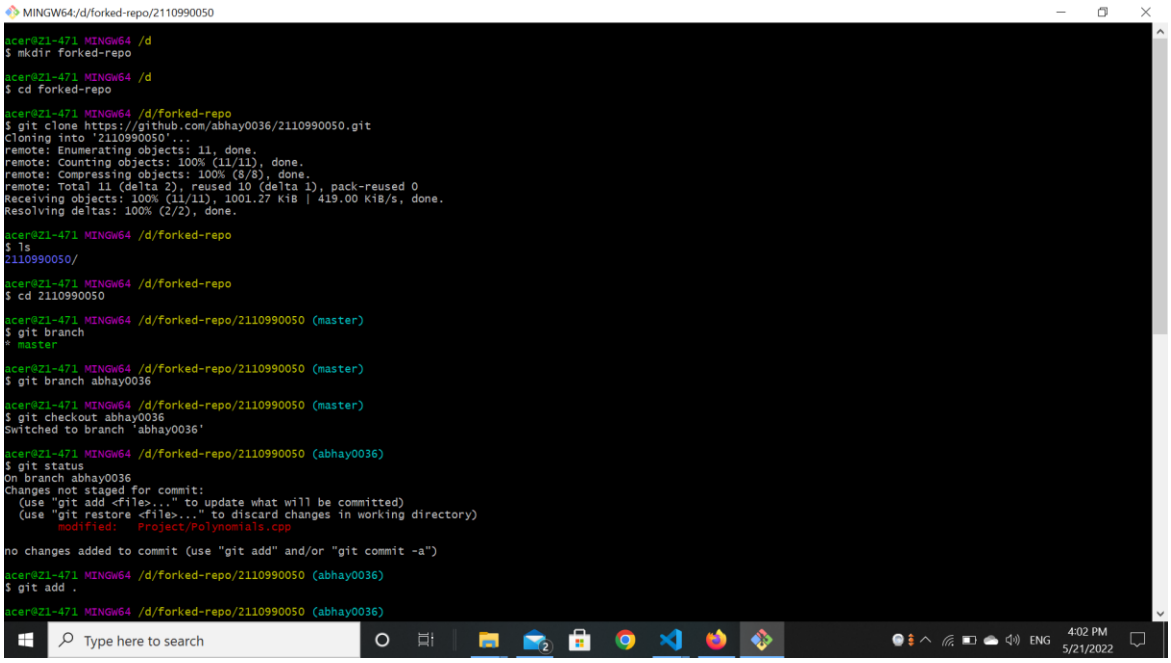
2.Click on the option ‘Fork’ in the top right corner.

3.You now have a forked repository.

## CLONING THE REPO INTO YOUR DEVICE

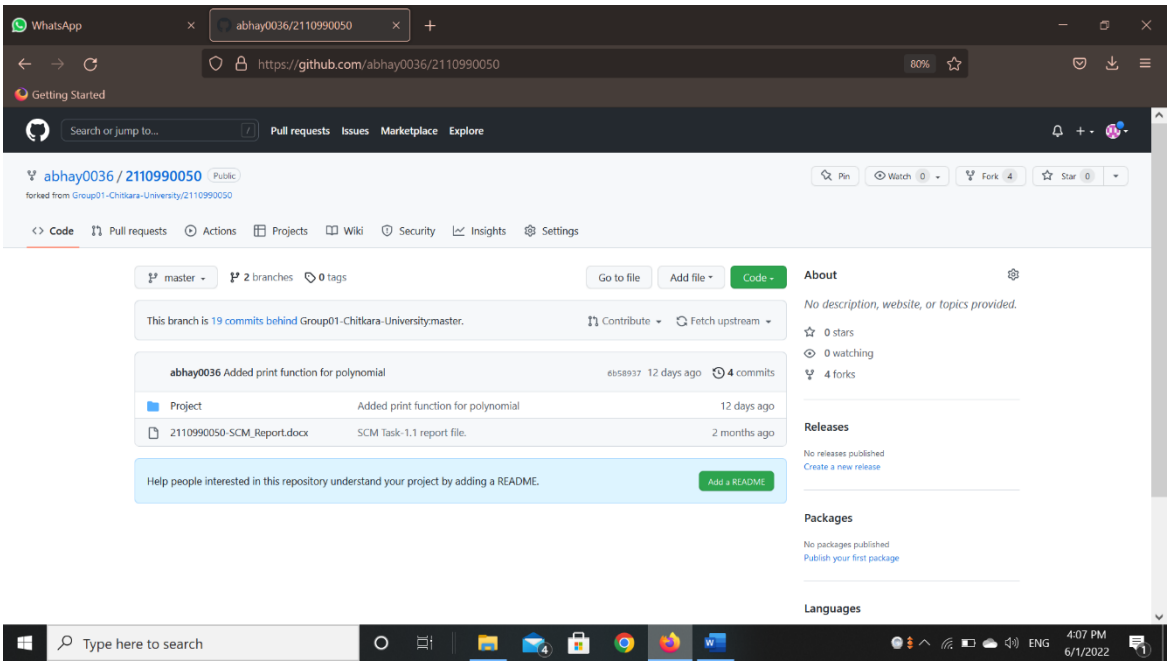
When you create a repository on GitHub.com, it exists as a remote repository. You can clone your repository to create a local copy on your computer and sync between the two locations.

1. Once you have forked the repository, you can clone it into your computer using directly the option given on github or through running git clone command in git bash.



2. Copy the URL of the forked repository

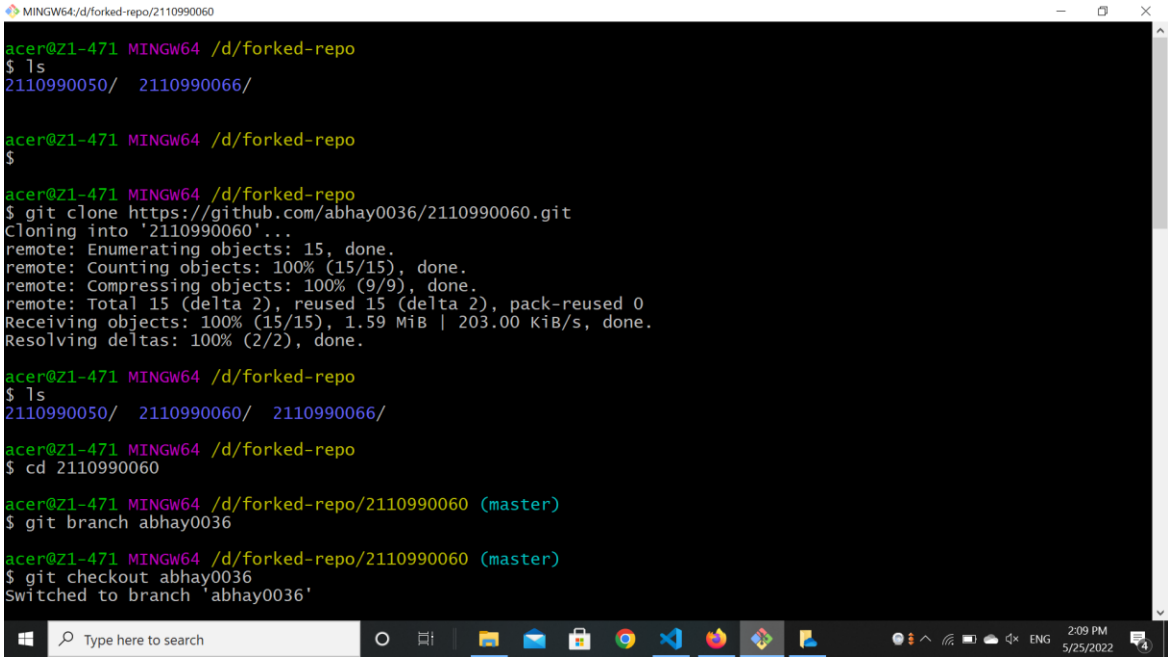
3. Open git bash and type the command " git clone <url of the forked repository>"

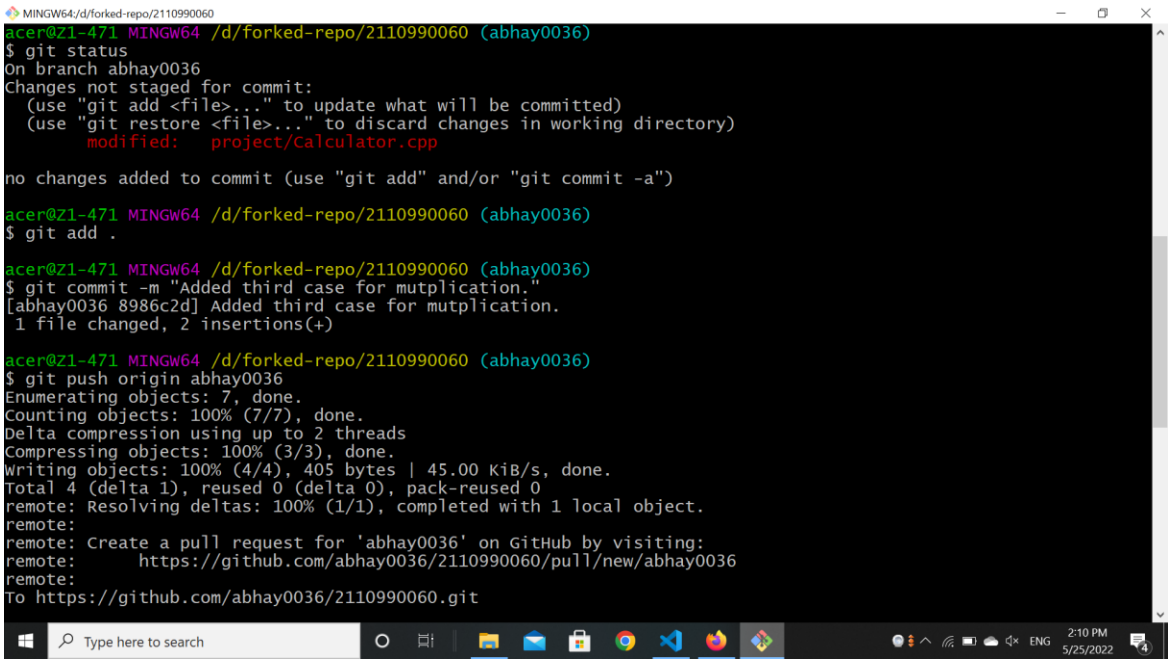



## COMMITING CHANGES TO THE FORKED REPOSITORY

1. Once you have cloned the repository you can introduce changes to it as per your wish.
2. After changing it you have to stage the file and then commit it.

3. After committing changes push it to your remote repository.

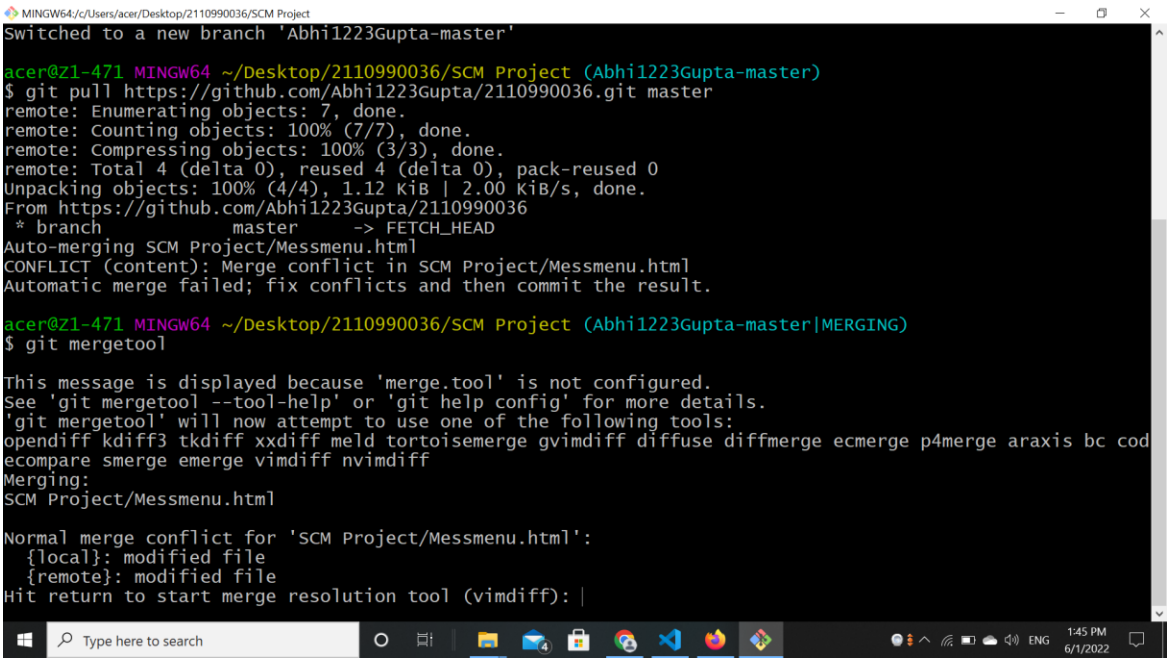


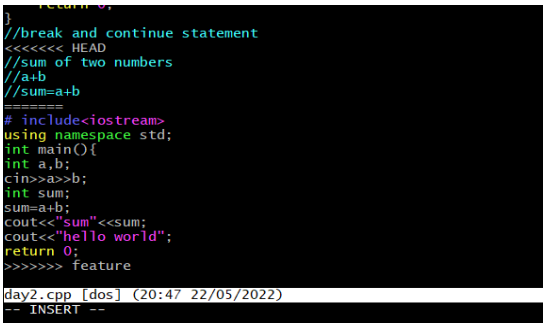



# MERGE AND RESOLVE CONFLICTS CREATED DUE TO OWN ACTIVITY AND COLLABORATORS ACTIVITY

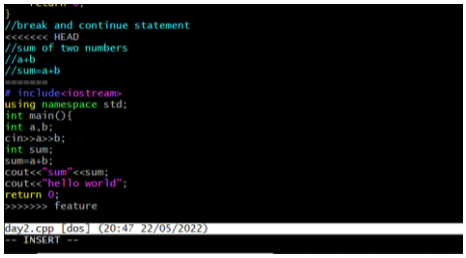
Merging and conflicts are a common part of the Git experience. Conflicts generally arise when two people have changed the same lines in a file, or if one developer deleted a file while another developer was modifying it. In these cases, Git cannot automatically determine what is correct. Conflicts only affect the developer conducting the merge, the rest of the team is unaware of the conflict. Git will mark the file as being conflicted and halt the merging process. It is then the developers' responsibility to resolve the conflict.

1.To understand the merging concept of branches, create a branch named “feature” in your repository.










2.Here, there is a file called ‘day2.cpp’. Make changes to it, add and commit them.

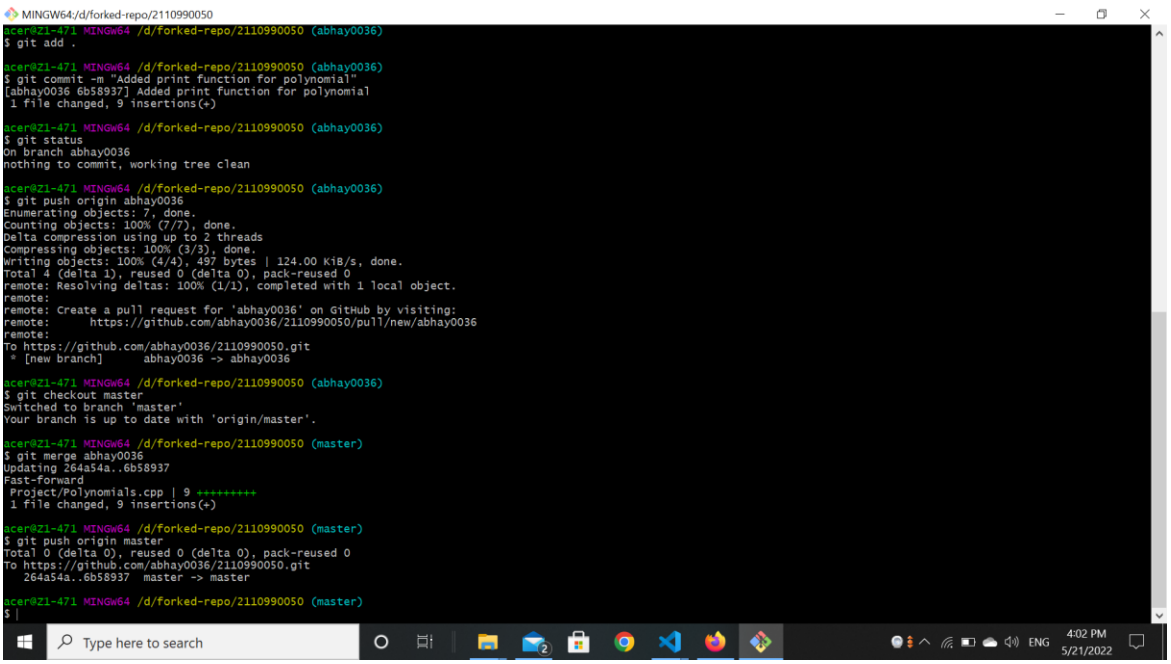
3.Similarly, change the same lines of day2.cpp file in the master branch.

4.If you are not already on the branch that you want the other one to merged in (in this example master branch), then switch to it.

5.Using the command try merging feature branch into master branch using the “git merge <branch name>”

6.Auto merging fails and conflict arises. In order to resolve it we make use of the mergetool by running the command “git mergetool”. The mergetool editor will open.

7.Make changes as per requirement in order to resolve the conflicts and exit the editor.



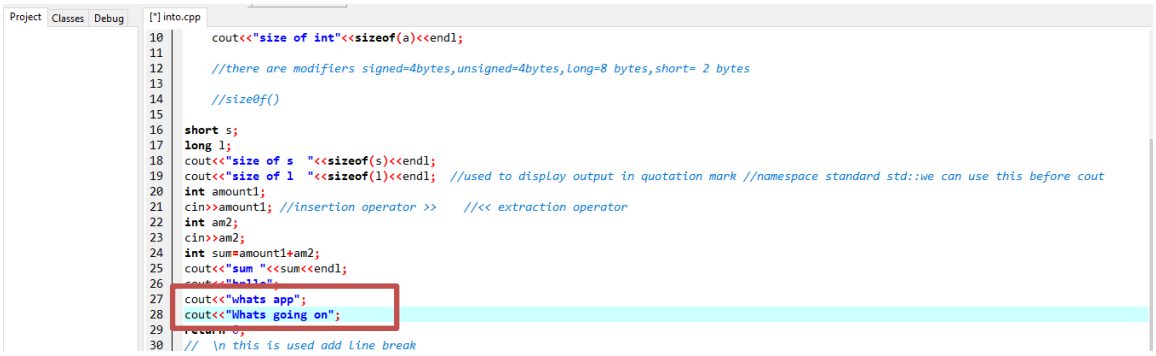

# RESET AND REVERT

While Working with Git in certain situations we want to undo changes in the working area or index area, sometimes remove commits locally or remotely and we need to reverse those changes. We can do it by using the git reset, git revert, git checkout commands.

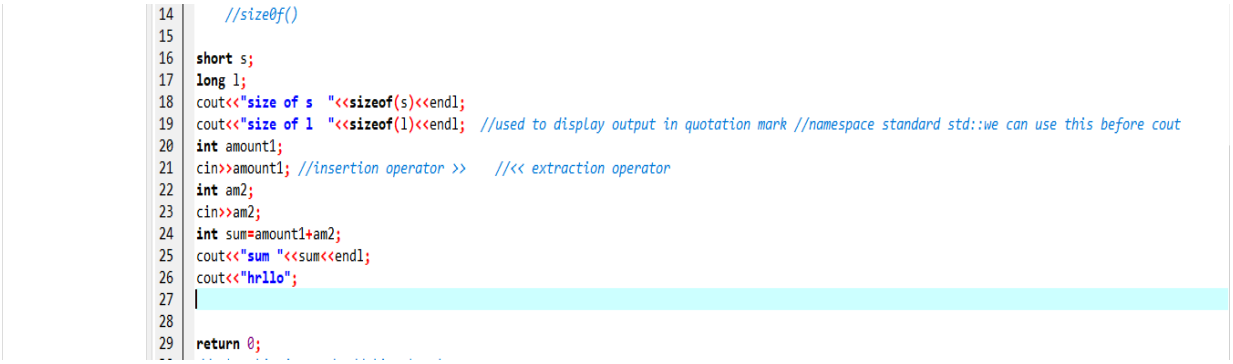
**RESET**-

git reset is used when we want to unstage a file and bring our changes back to the working directory. Git reset can also be used to remove commits from the local repository.

Suppose we make edits to a file, stage it and commit it





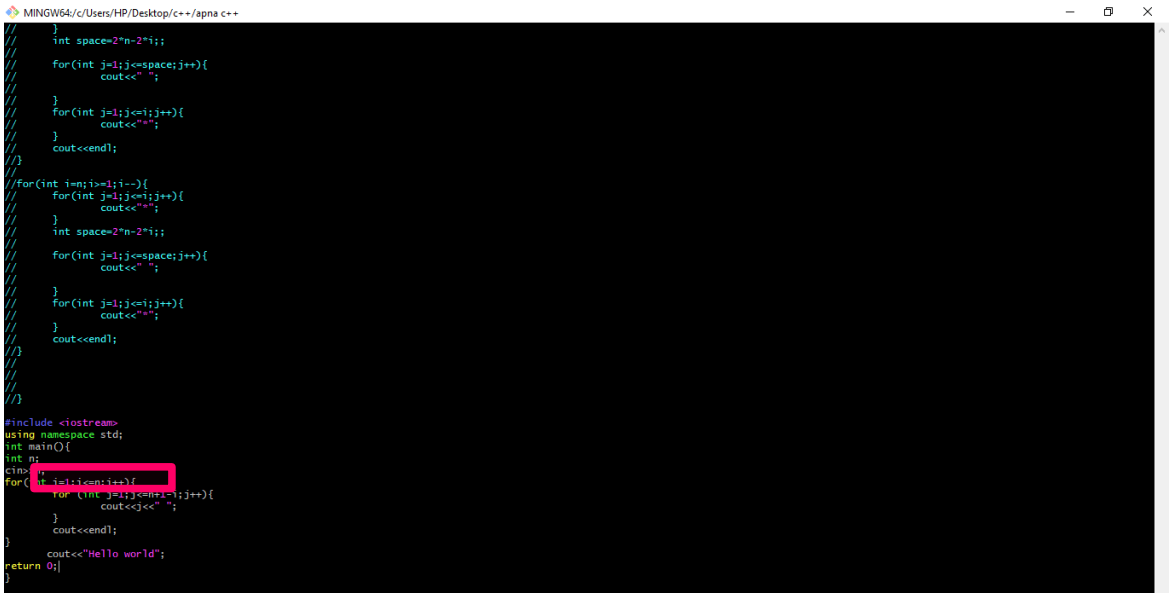



In order to reset the changes made in the recent commit, run the “git reset --hard HEAD~1” command.
Or a command git “reset commit no.”

The HEAD returns to the previous commit and the changes made are reset.

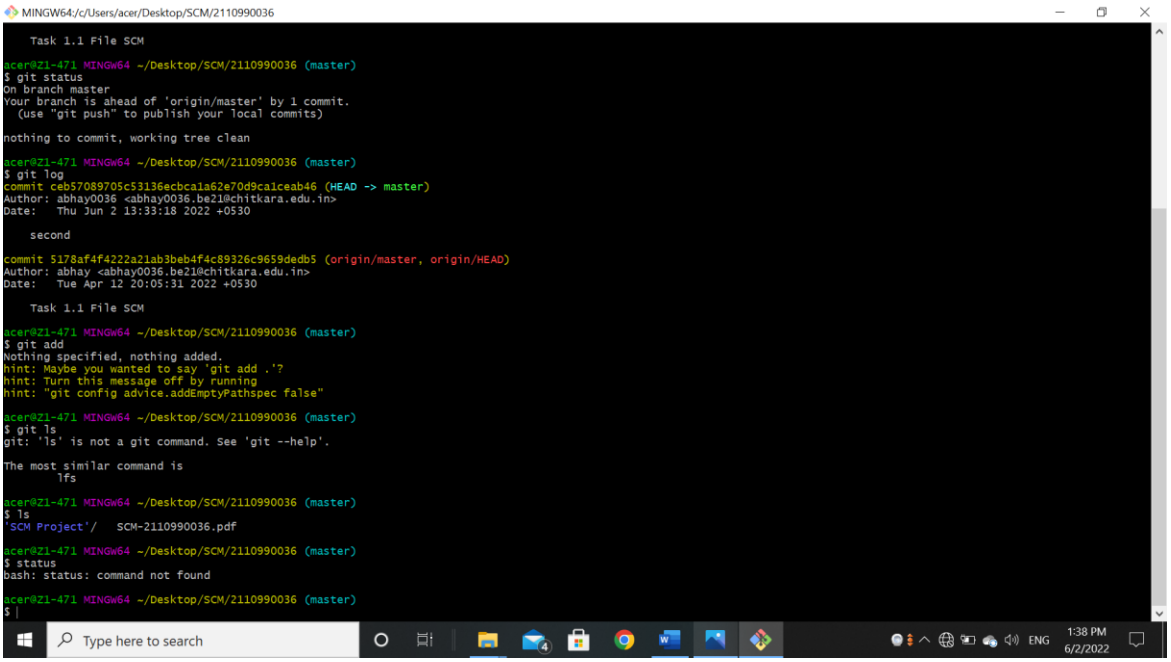

**REVERT**-

git revert is used to remove the commits from the remote repository. git revert removes the commit that we have done but adds one more commit which tells us that the revert has been done.




In order to understand it add changes to a file, stage and commit it.


Now to revert the changes made in the commit run the “git revert <commit id>” command.

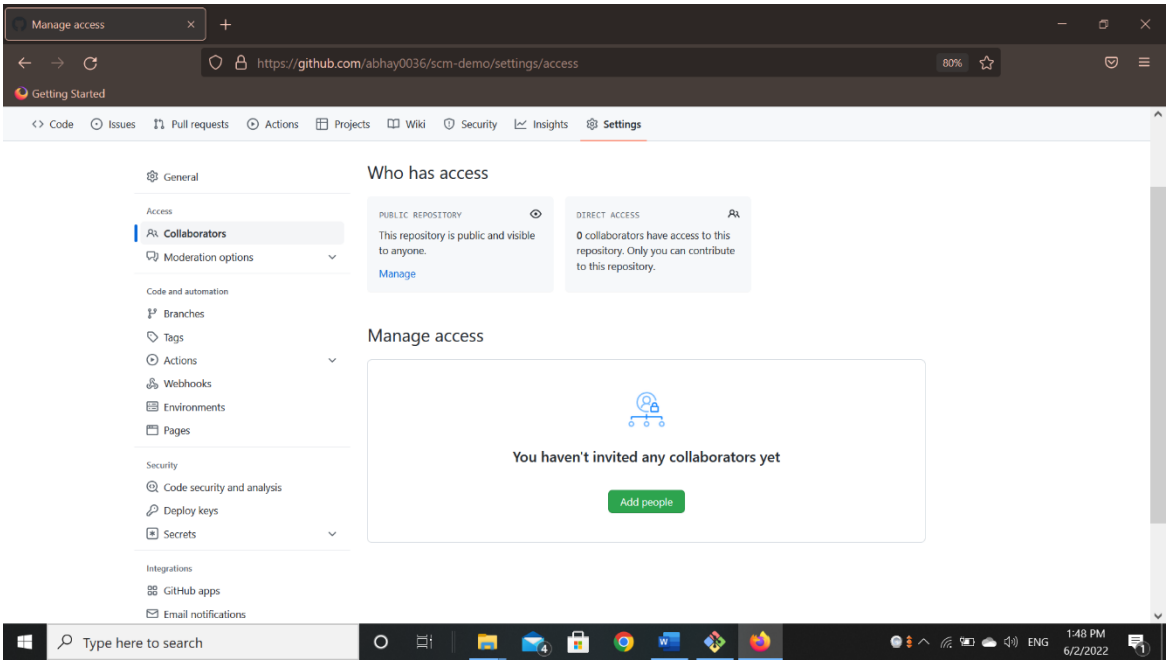


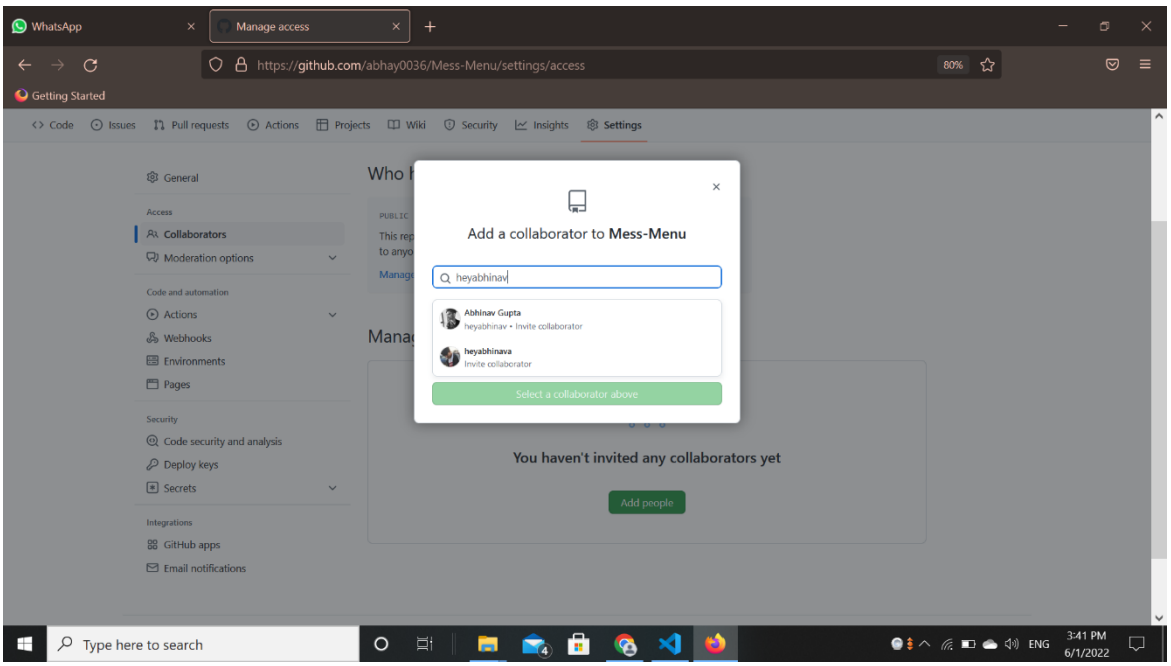

You can see that a new commit as ‘revert “changes made”’ is there and the file has returned to its previous state.

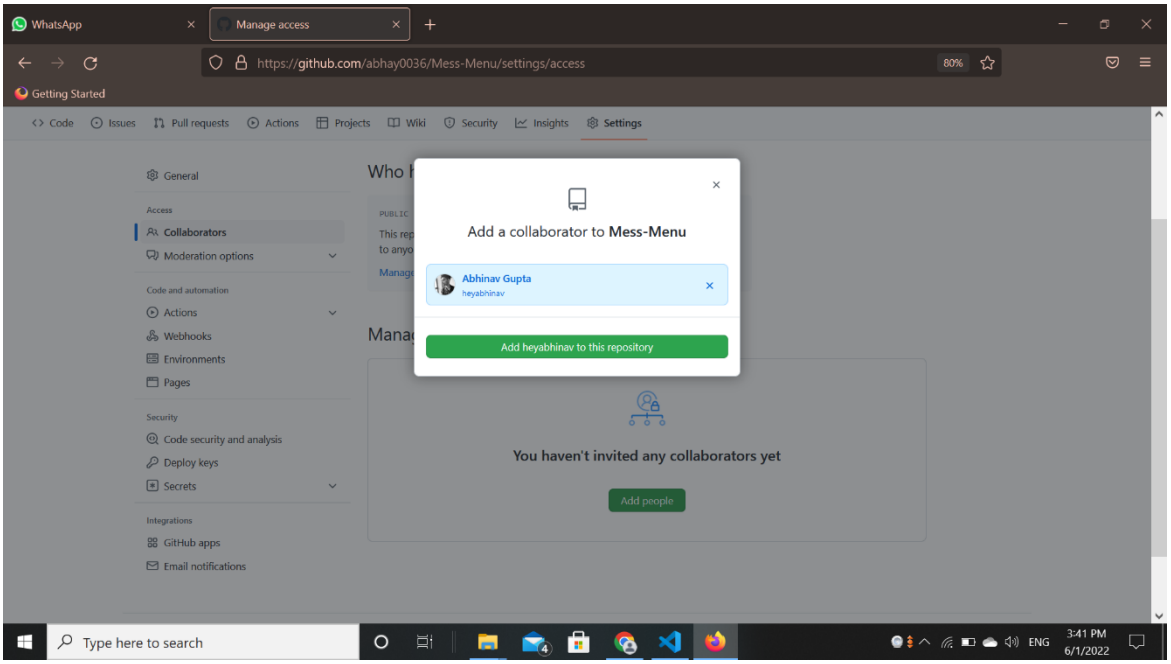
## ***CREATE A DISTRIBUTED REPOSITORY AND ADD MEMBERS IN PROJECT TEAM***

1. On the homepage of your GitHub account, click on Repositories option in the menu bar.
2. Click on the ‘New’ button in the top right corner.

3. Enter the Repository name and add the description of the repository.
4. To add members to your repository, open your repository and select settings option in the navigation bar.
5. Click on Collaborators option under the access tab.
6. You can manage access and add/remove team members to your project.
7. To add members, click on the add people option and search the id of your respective team member.

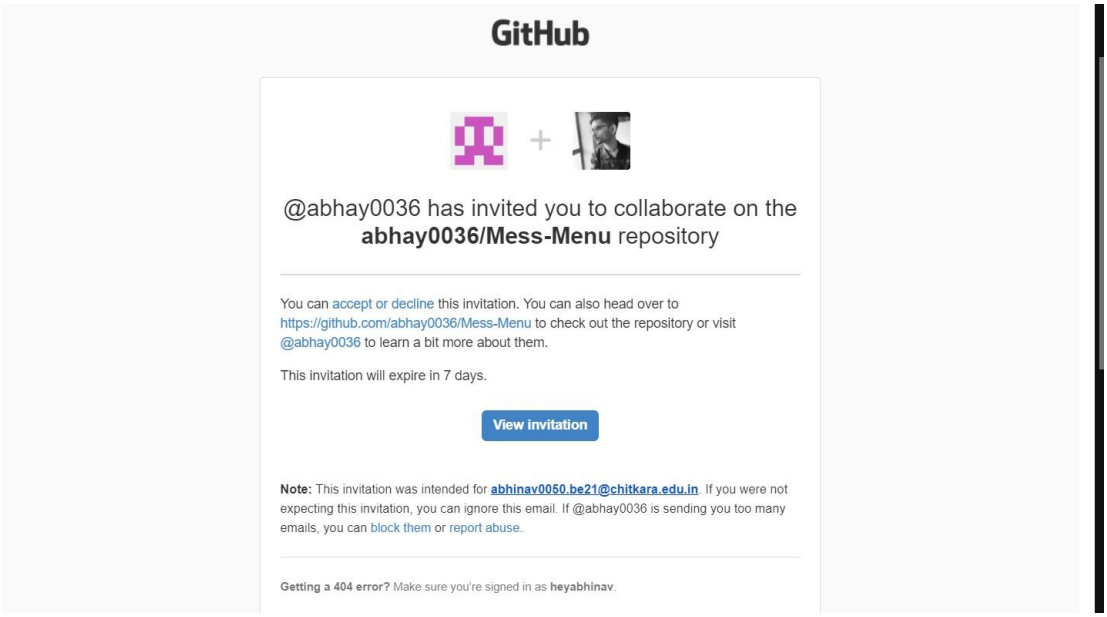

Manage access
https://github.com/abhay0036/scm-demo/settings/access
Getting Started
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
General
Access
Collaborators
Moderation options
Code and automation
Branches
Tags
Actions
Webhooks
Environments
Pages
Security
Code security and analysis
Deploy keys
Secrets
Integrations
GitHub apps
Email notifications
Who has access
PUBLIC REPOSITORY
This repository is public and visible to anyone.
Manage
DIRECT ACCESS
0 collaborators have access to this repository. Only you can contribute to this repository.
Manage access
You haven't invited any collaborators yet
Add people
Type here to search
ENG
1:48 PM
6/2/2022


WhatsApp
Manage access
https://github.com/abhay0036/Mess-Menu/settings/access
Getting Started
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
General
Access
Collaborators
Moderation options
Code and automation
Actions
Webhooks
Environments
Pages
Security
Code security and analysis
Deploy keys
Secrets
Integrations
GitHub apps
Email notifications
Add a collaborator to Mess-Menu
heyabhinav
Abhinav Gupta
heyabhinav • Invite collaborator
heyabhinava
Invite collaborator
Select a collaborator above
You haven't invited any collaborators yet
Add people
Type here to search
ENG
3:41 PM
6/1/2022




8. To accept the invitation from your team member, open your email registered with GitHub.

9. You will receive an invitation mail from the repository owner. Open the email and click on accept invitation.




10. You will be redirected to GitHub where you can either select to accept or decline the invitation.

Similarly, you can add more collaborators to your project.

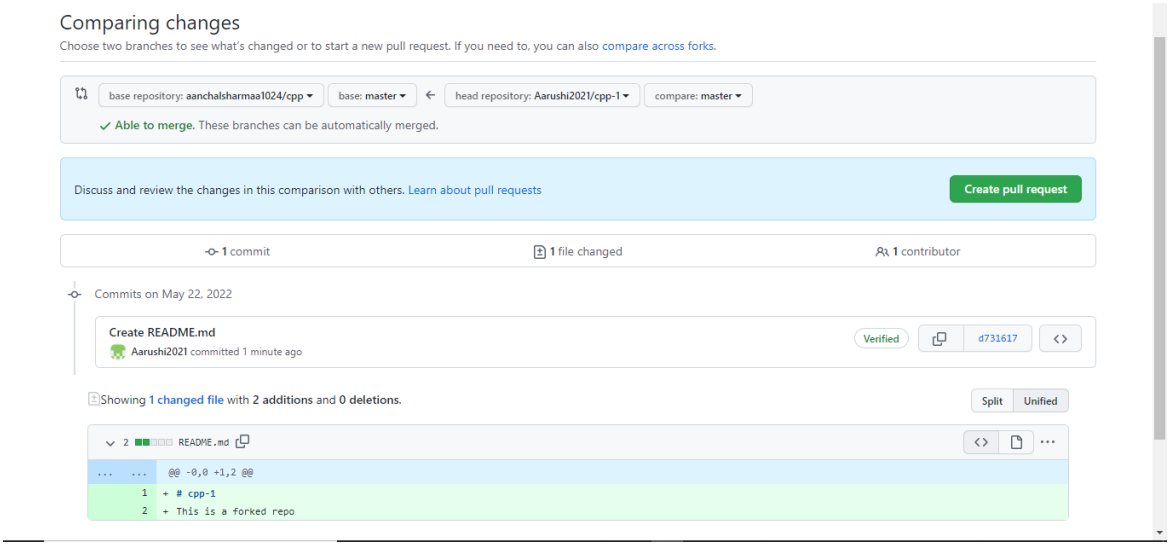
# OPEN AND CLOSE A PULL REQUEST

1. First, select a repository of the other person in which you want to make changes and create a pull request.
2. Clone it into your local storage.
3. To open a pull request we first have to make a new branch, by using git checkout -b *branch name* option.
4. After making new branch we add a file to the branch or make changes in the existing file.
5. Add and commit the changes to the local repository.

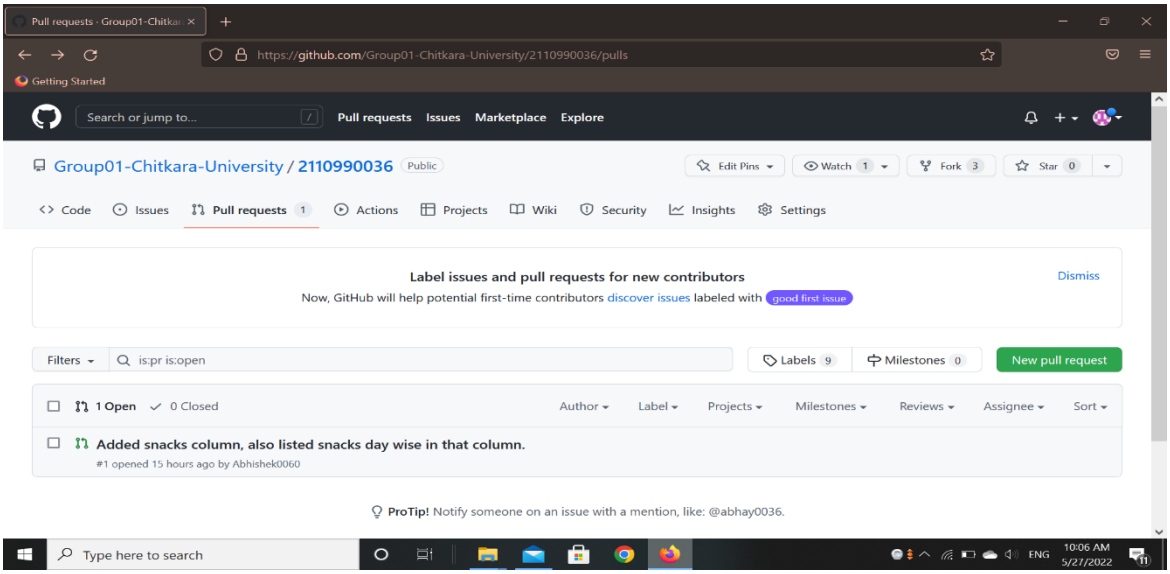

6. Use git push origin *branch name* option to push the new branch to the main repository.

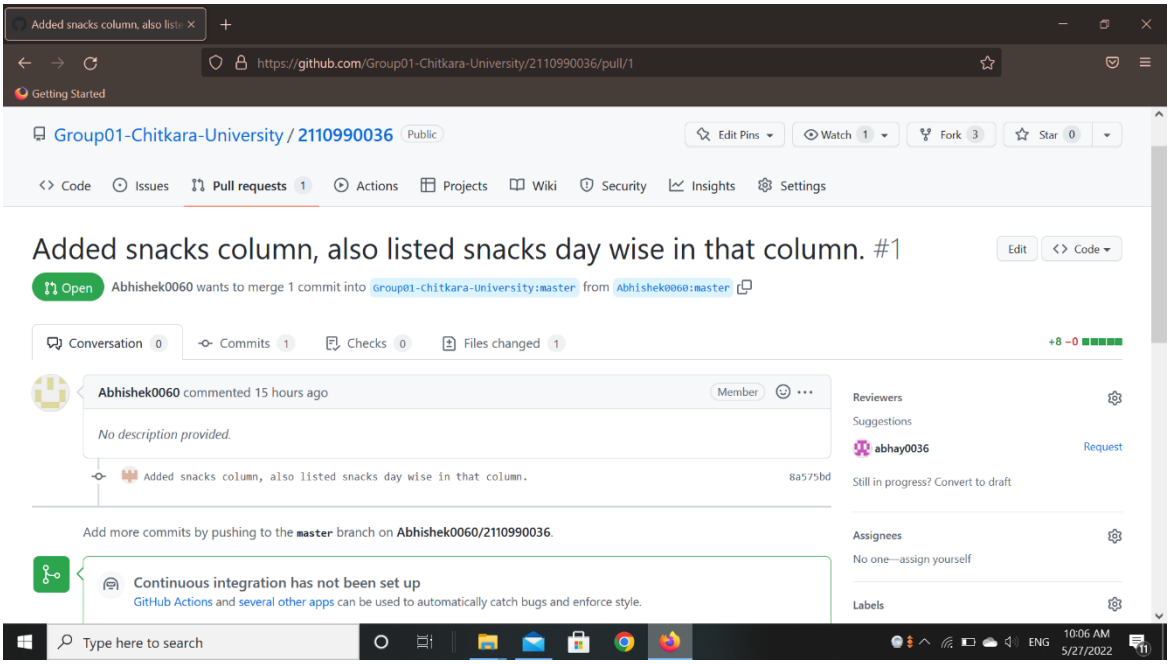
```
acer@Z1-471 MINGW64 /d/forked-repo/2110990050 (abhay0036)
$ git push origin abhay0036
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 2 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (4/4), 497 bytes | 124.00 KiB/s, done.
Total 4 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
remote:
remote: Create a pull request for 'abhay0036' on GitHub by visiting:
remote:      https://github.com/abhay0036/2110990050/pull/new/abhay0036
remote:
To https://github.com/abhay0036/2110990050.git
 * [new branch]      abhay0036 -> abhay0036

acer@Z1-471 MINGW64 /d/forked-repo/2110990050 (abhay0036)
$ git checkout master
Switched to branch 'master'
Your branch is up to date with 'origin/master'.
```

7. After pushing new branch GitHub will either automatically ask you to create a pull request or you can create your own pull request.

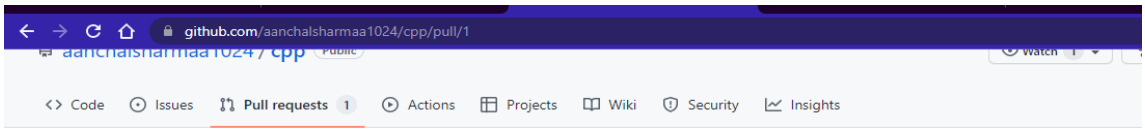



8. To create your own pull request, click on pull request option.

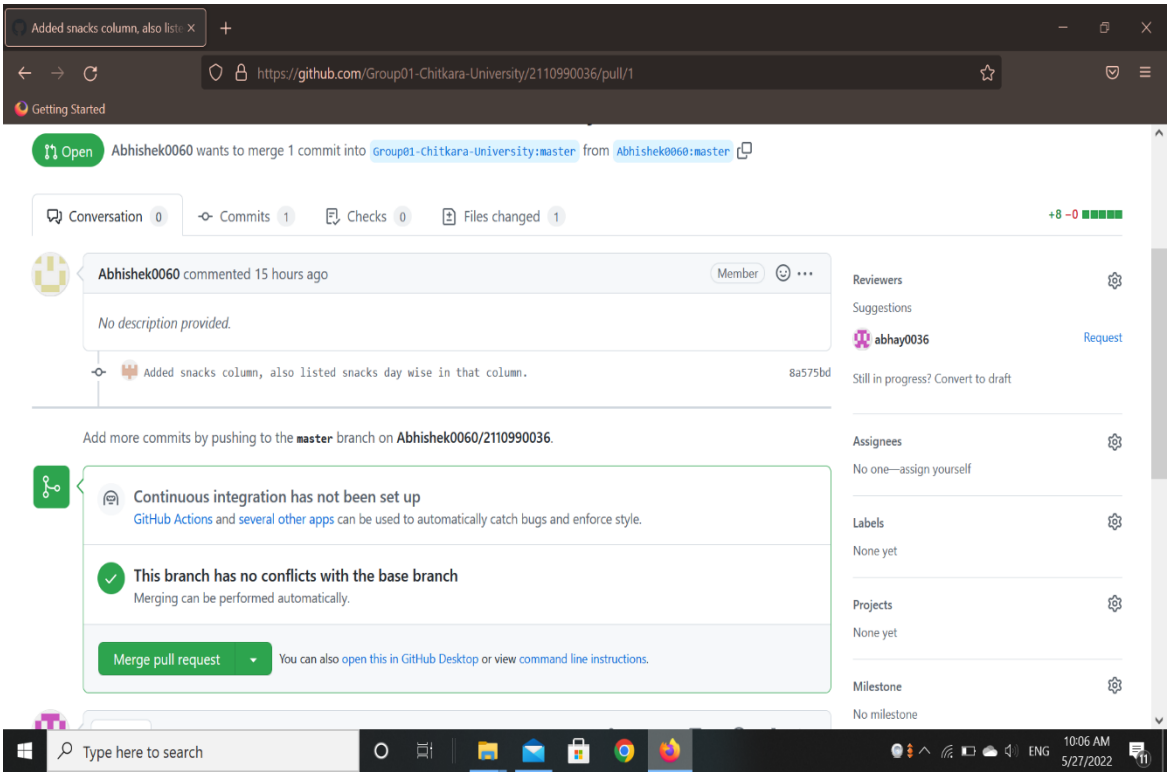





9. GitHub will detect any conflicts and ask you to enter a description of your pull request.
10. After opening a pull request the owner of the original repository will be sent the request if they want to merge or close the request.




11. If the owner chooses not to merge your pull request, they will close it.
12. To close the pull request simply click on close pull request and add comment/ reason why you closed the pull request.
13. If you want to merge it into the original, click on merge pull request.

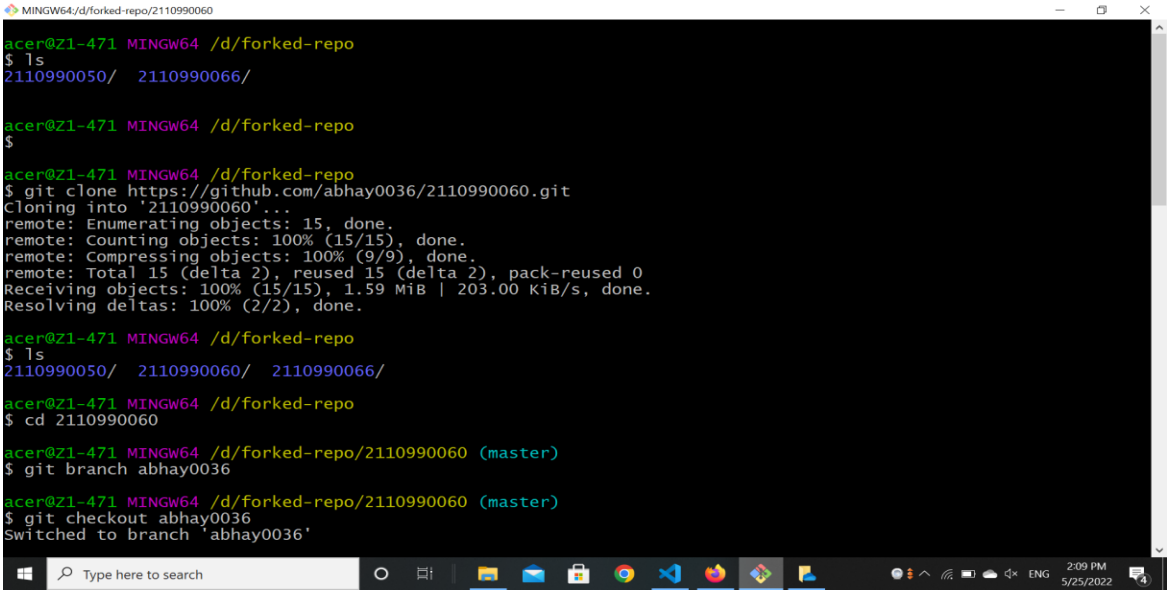


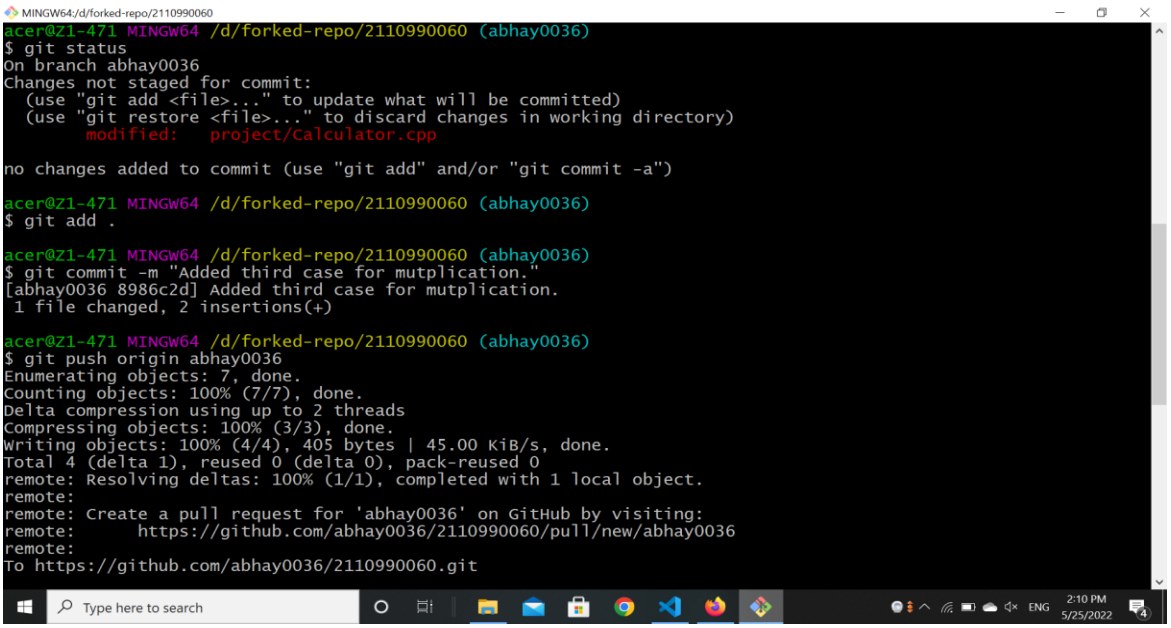

## ***CREATE A PULL REQUEST ON A TEAM MEMBER'S REPO AND CLOSE PULL REQUESTS GENERATED BY TEAM MEMBERS ON OWN REPOSITORY AS A MAINTAINER***

### #OPENING PULL REQUESTS ON TEAM MEMBER'S REPOSITORY

1. Do the required changes in the repository, add and commit these changes in the local repository in a new branch.

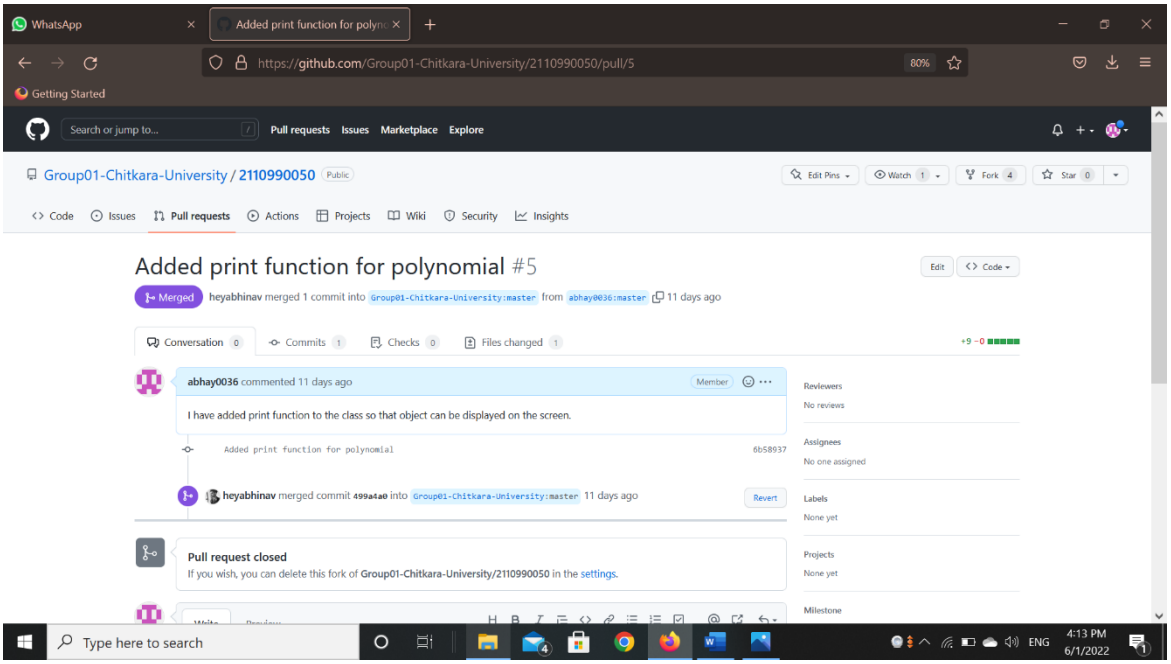
2. Push the modified branch using git push origin branchname.




3. Open a pull request by following the procedure from the above experiment.

4.The pull request will be created and will be visible to all the team members.

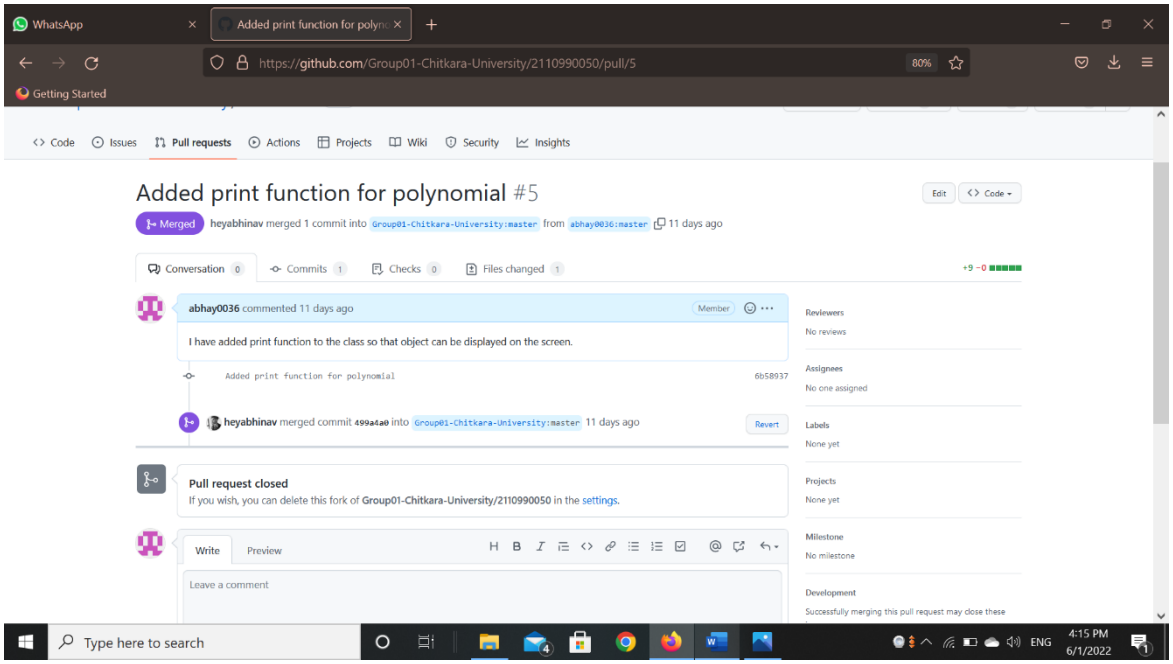



5. Click on compare & pull request option appearing at the top.

6 .After you enter a description, click on the open request button.

7. Pull request has been sent to your team member. They can choose to either close the pull request or merge it.

8. Suppose they merged it, then the changes you made to the forked repository will be introduced into the owner's original repository and you will be notified about merging.


WhatsApp
Added print function for polyno
https://github.com/Group01-Chitkara-University/2110990050/pull/5
80%
Getting Started
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
Added print function for polynomial #5
Edit
Code
Merged
heyabhinav merged 1 commit into Group01-Chitkara-University:master from abhay0036:master 11 days ago
Conversation 0
Commits 1
Checks 0
Files changed 1
+9 −0
abhay0036 commented 11 days ago
Member
I have added print function to the class so that object can be displayed on the screen.
Added print function for polynomial
6b58937
heyabhinav merged commit 499a4a0 into Group01-Chitkara-University:master 11 days ago
Revert
Pull request closed
If you wish, you can delete this fork of Group01-Chitkara-University/2110990050 in the settings.
Write
Preview
Leave a comment
Reviewers
No reviews
Assignees
No one assigned
Labels
None yet
Projects
None yet
Milestone
No milestone
Development
Successfully merging this pull request may close these
Type here to search
ENG
4:15 PM
6/1/2022