

Subject Name: Source Code Management

Subject Code: CS181

Cluster: Beta

Department: DCSE



Submitted By:

Abhay Kumar

2110990036

G1-A

Submitted To:

Dr. Monit Kapoor

INDEX

S. NO	Experiment Name	Page No.
1.	Introduction	1-2
2.	Setting up of Git Client	3-6
3.	Setting up GitHub Account	7-8
4.	Program to Generate logs	9-10
5.	Create and visualize branches	11-12
6.	Git lifecycle description	13

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.

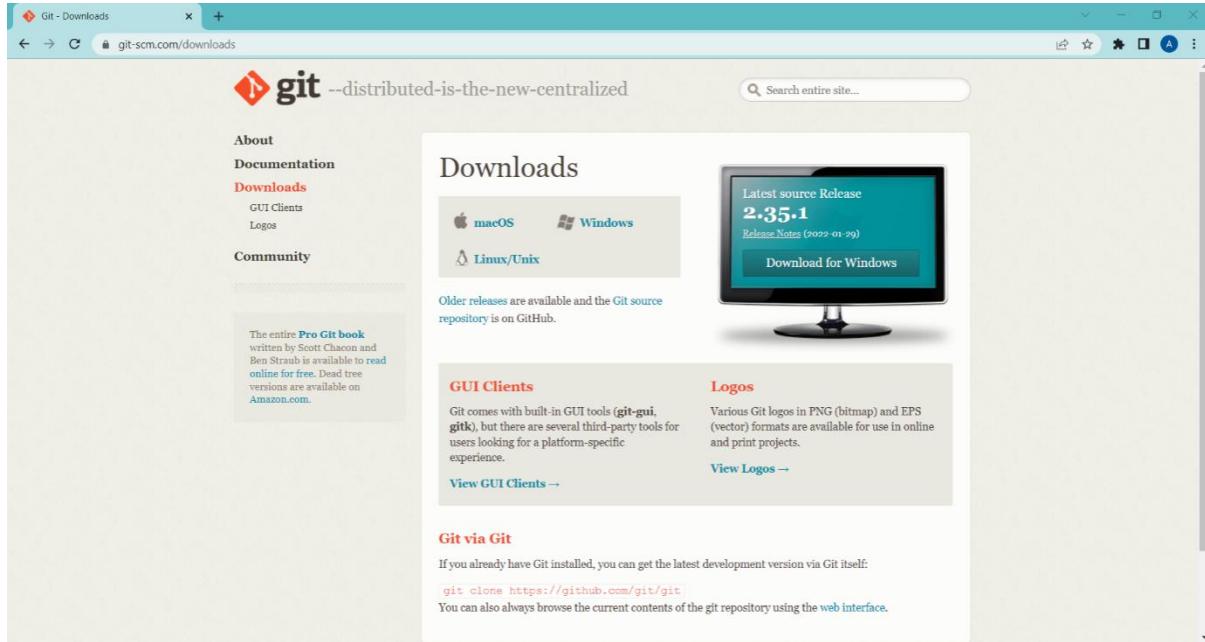
Types of VCS

- 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.

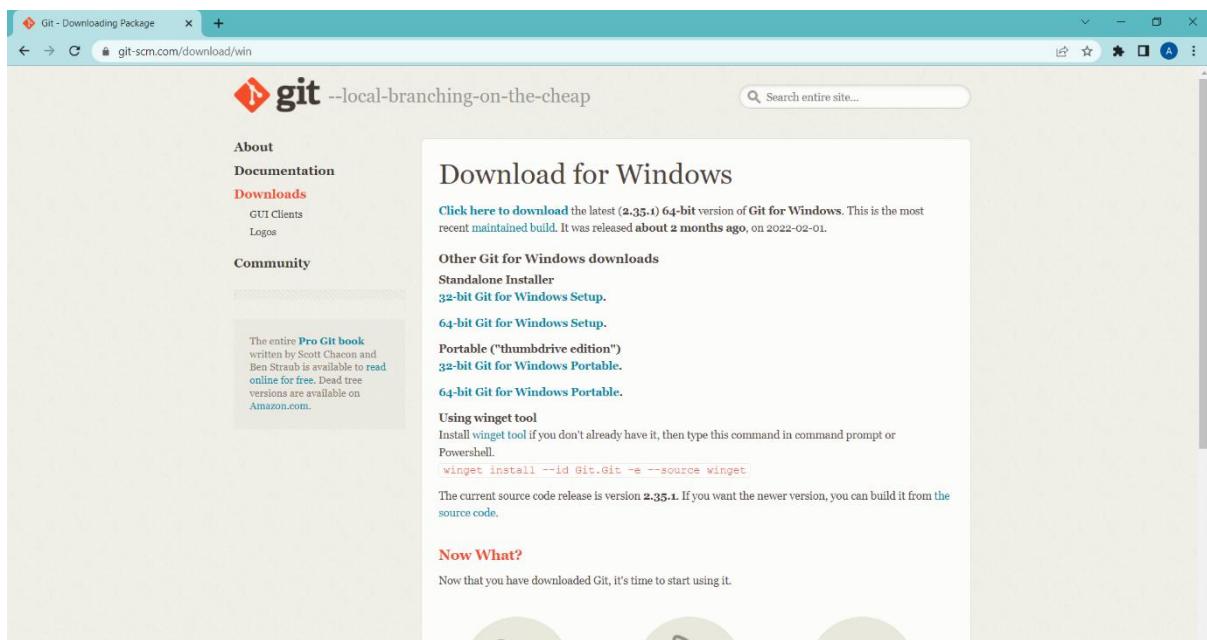
Experiment No. 01

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.

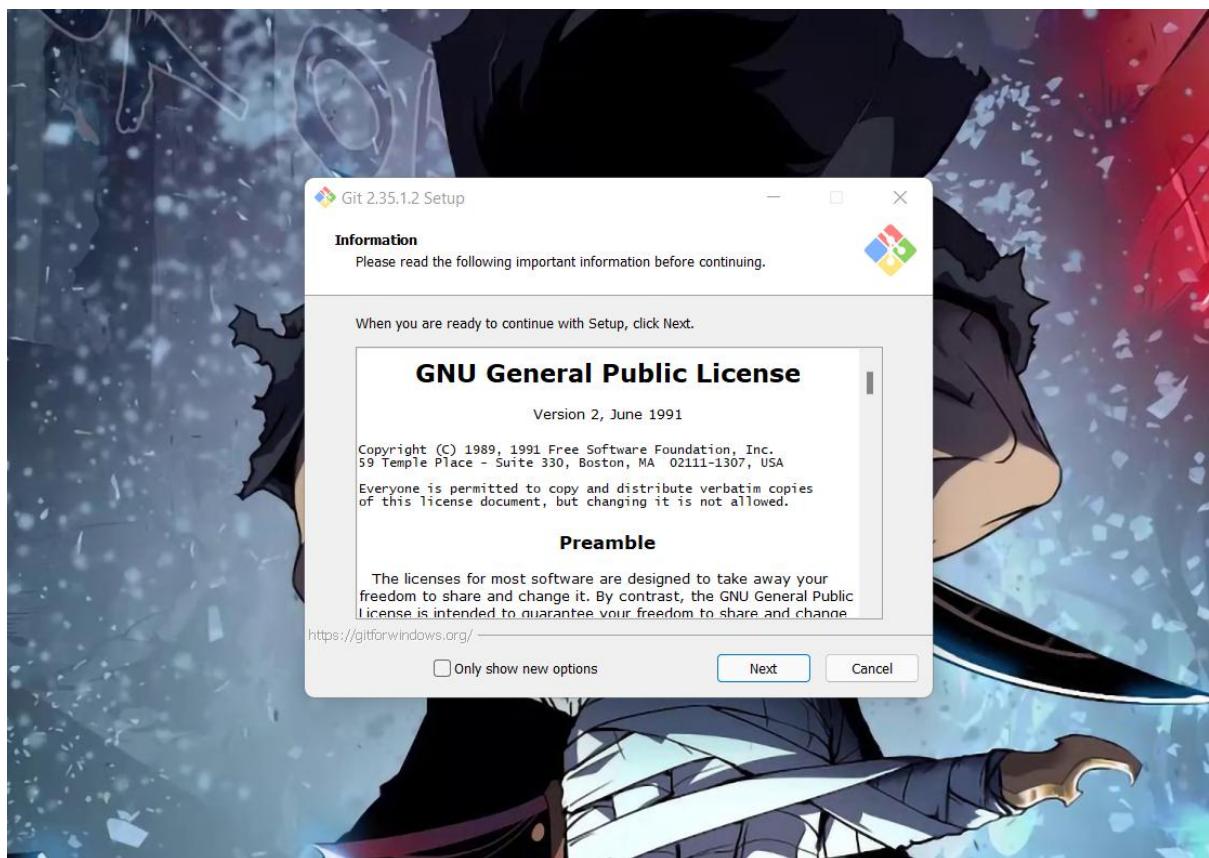


- ✧ Select the CPU for your system now. (Most of the system now runs on

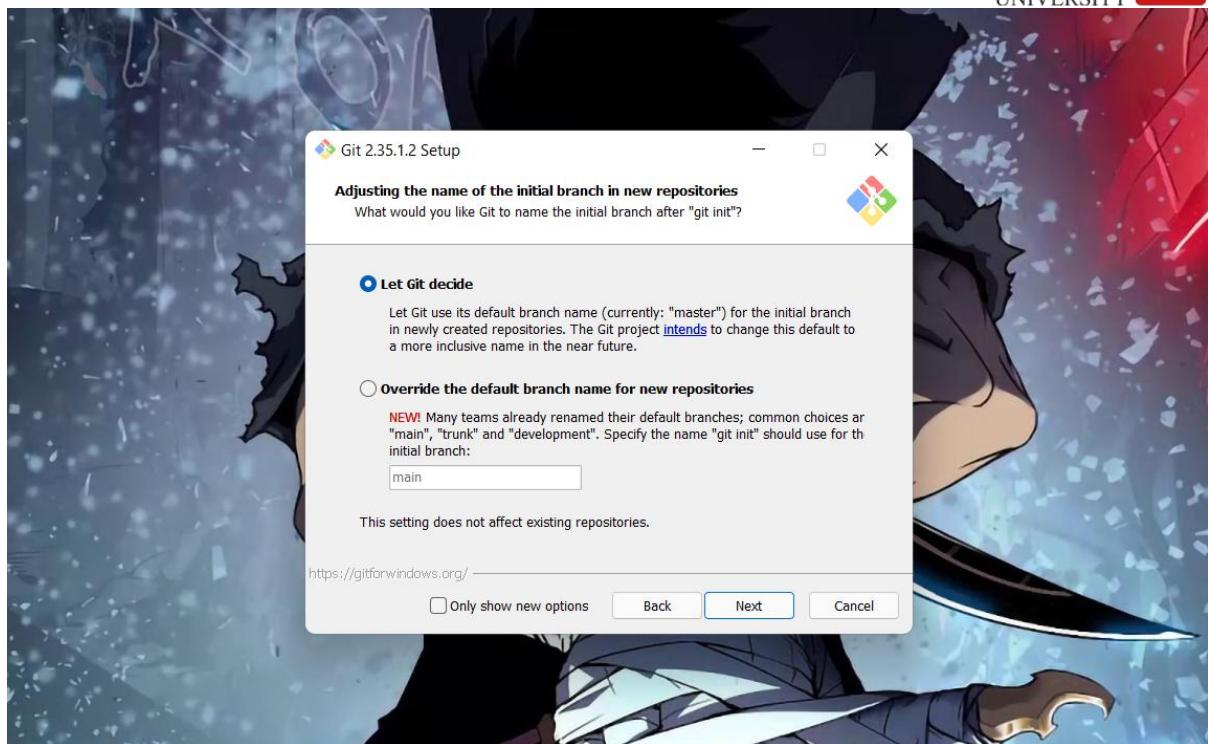
64-bit processors.) Your download will begin when you pick a processor.



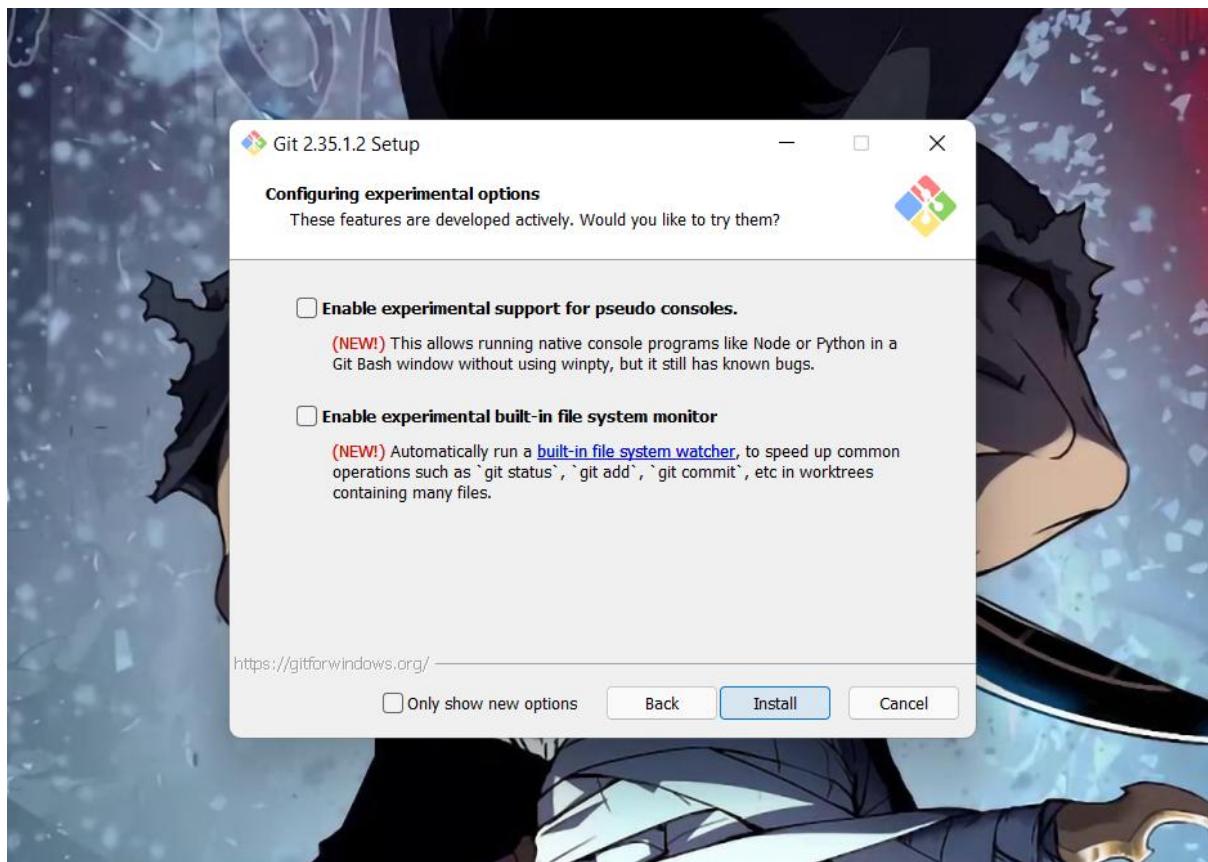
- ❖ You must now open the Git folder.
- ❖ You will be asked if you want to enable this program to make modifications to your PC once you launch it.
- ❖ YES should be selected.



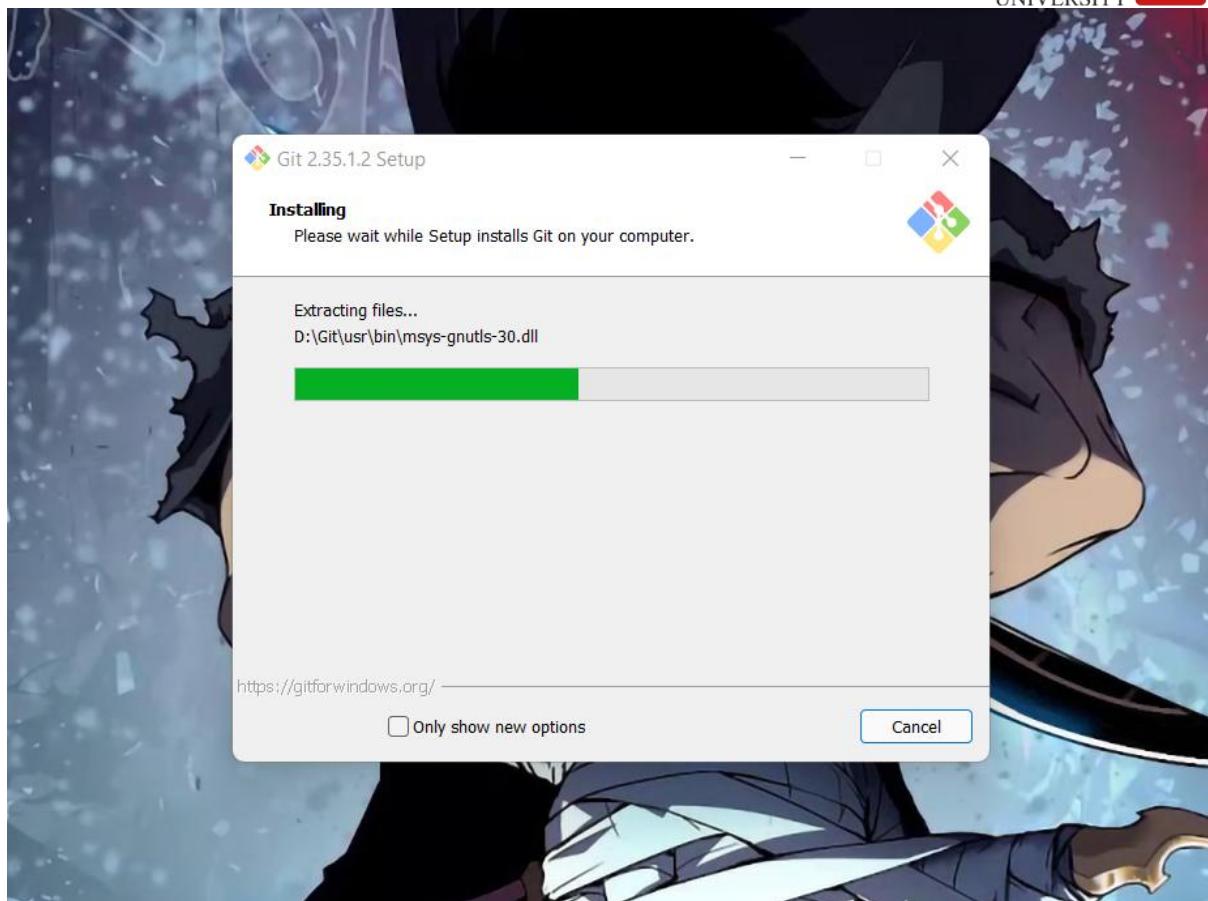
- ❖ Click on Next



✧ Continue clicking on next few times more



✧ Now select the Install option.

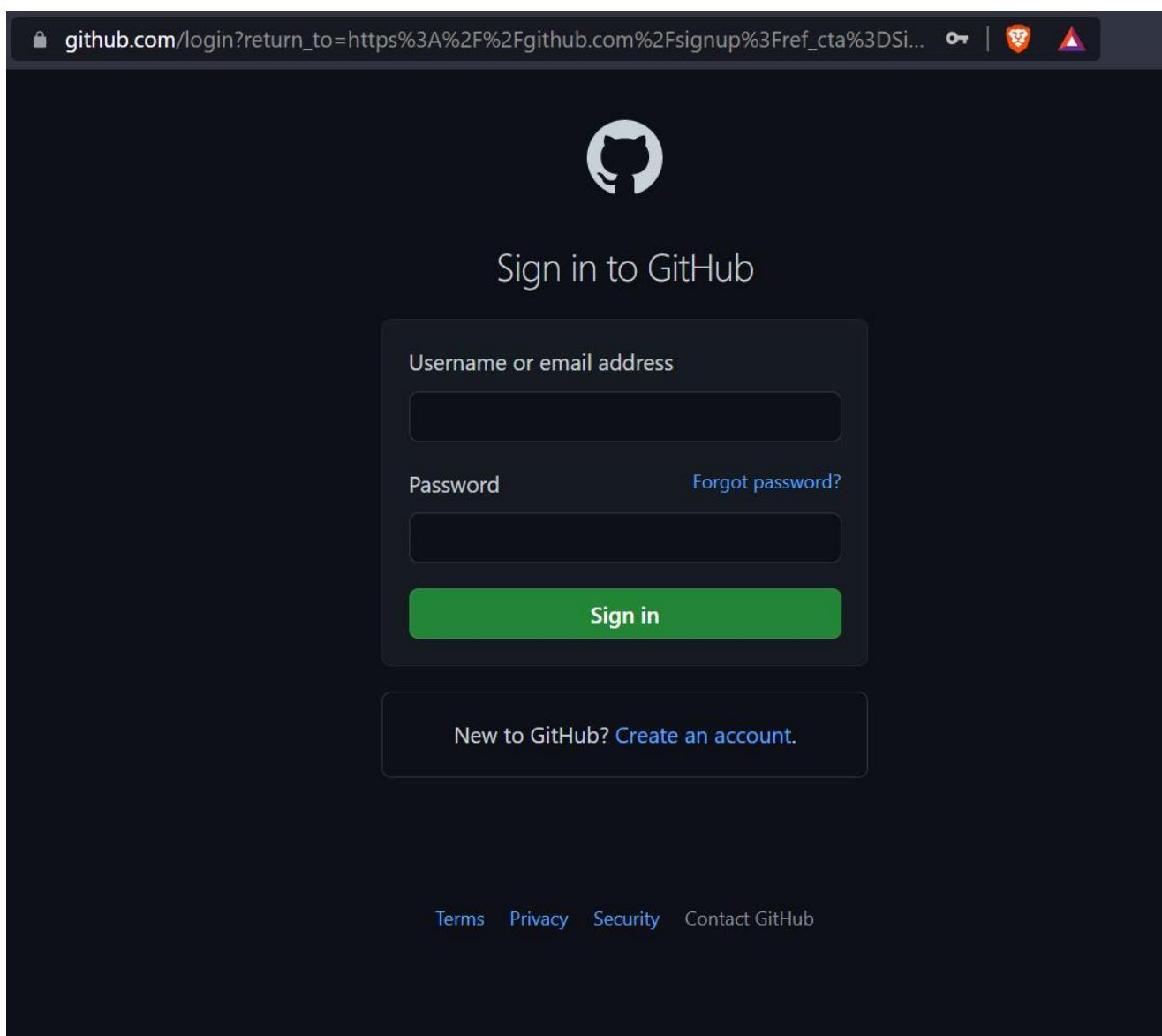


- ✧ Click on Finish after the installation is finished.

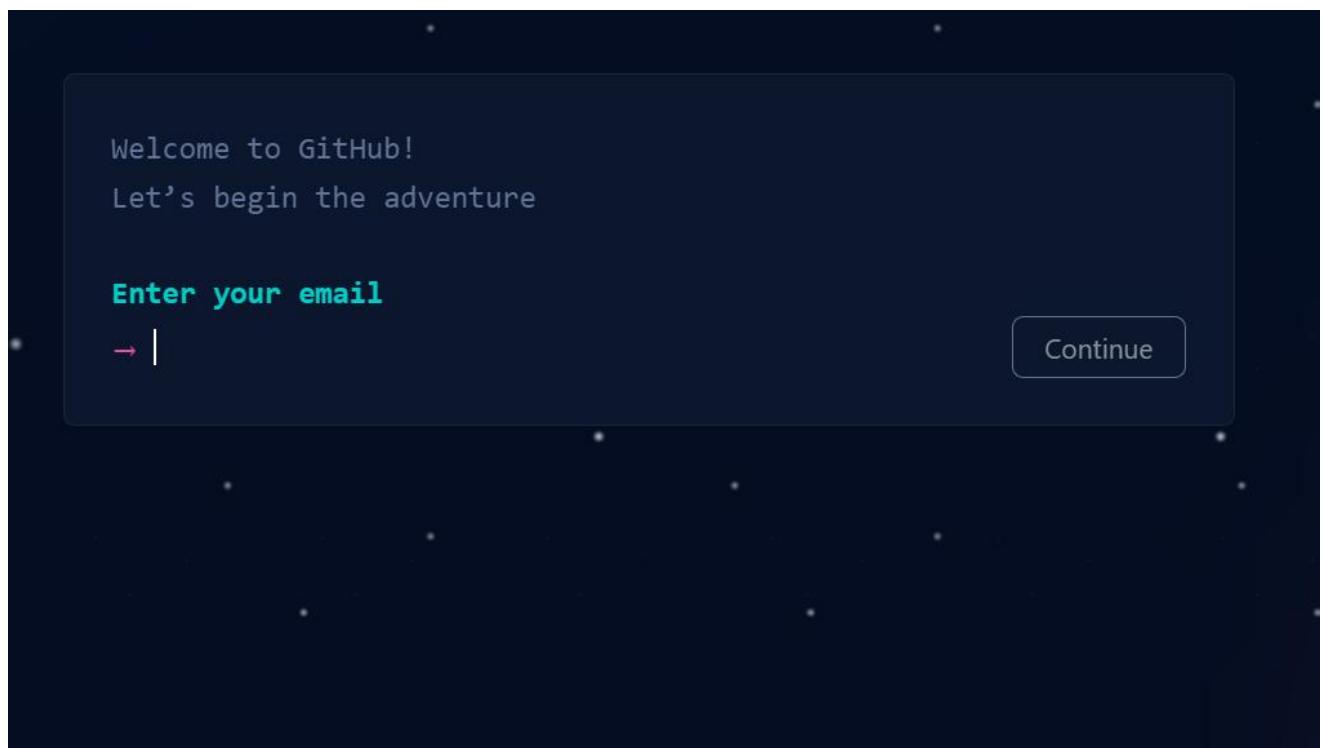
The installation of the git is finished and now we have to setup git client and GitHub account.

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.



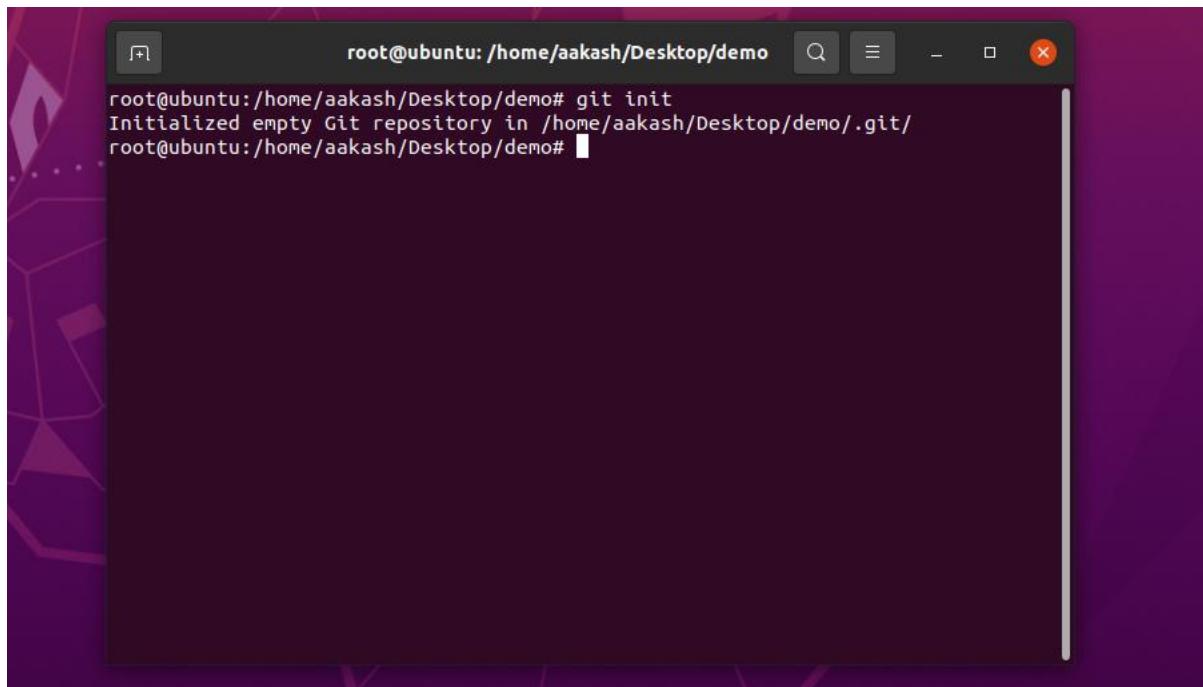
- ✧ After clicking on "Create a New Account," you will be sent to a new page where you must enter your email address for your account. Now type in the password you'd want to use for your GitHub account. Then you'll be prompted to enter your username.



- ✧ Now Click on Create Account.
- ✧ Verify it from your email and you are all set to go.

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.



The screenshot shows a terminal window titled "root@ubuntu:/home/aakash/Desktop/demo". The command "git init" is entered, and the output "Initialized empty Git repository in /home/aakash/Desktop/demo/.git/" is displayed. The terminal has a dark background and light-colored text. It is running on an Ubuntu desktop environment.

- ❖ 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 →

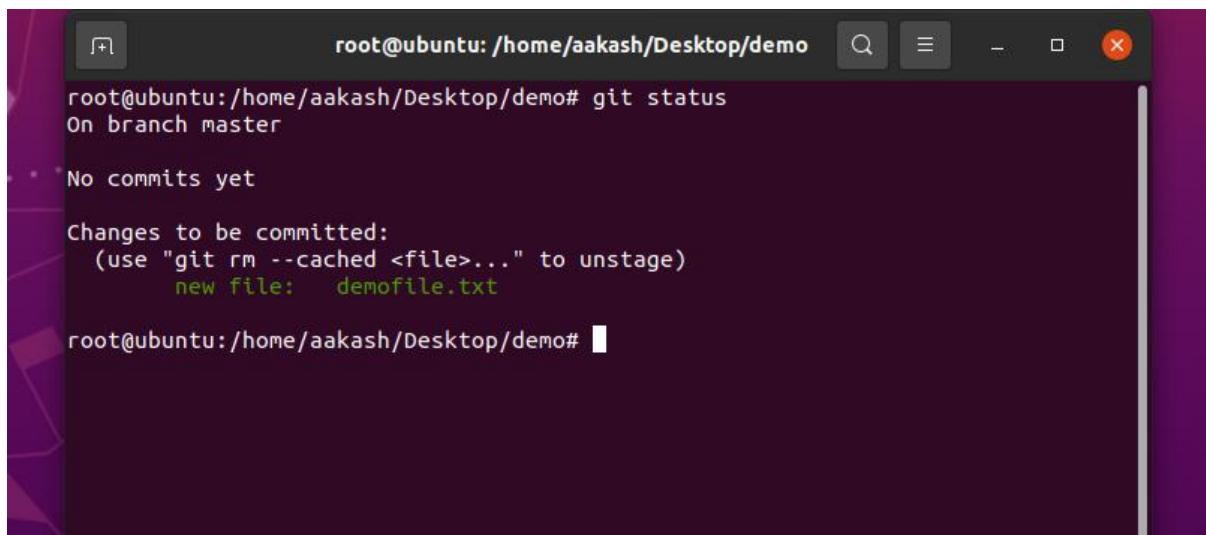
“git config --global user.name”
“git config --global user.email”

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:



```
root@ubuntu:/home/aakash/Desktop/demo# git status
On branch master

No commits yet

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

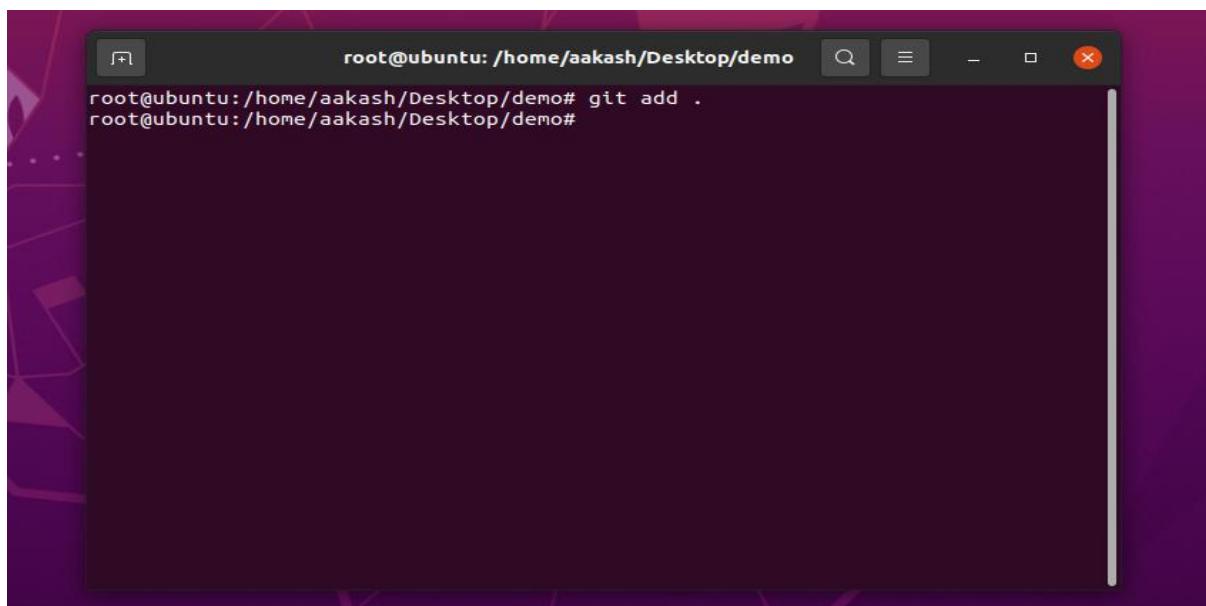
root@ubuntu:/home/aakash/Desktop/demo#
```

You can see that *index.html* is in red colour that means it is an untracked file.

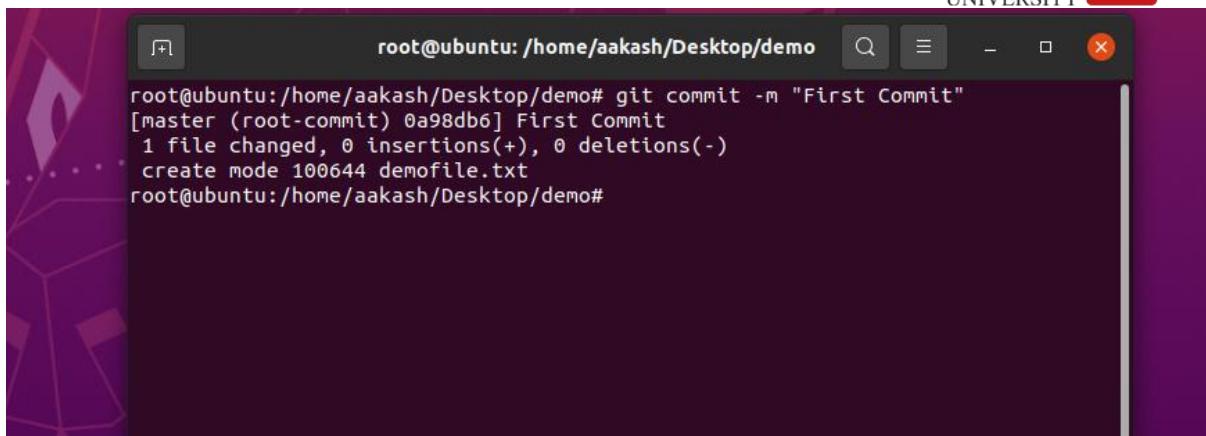
Now firstly add the file in staging area and then commit the file.

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 ]
```

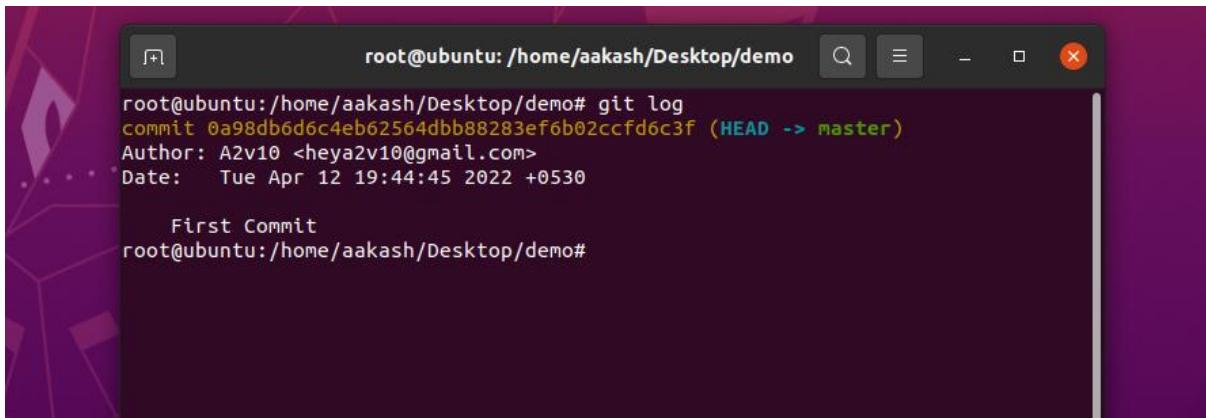


```
root@ubuntu:/home/aakash/Desktop/demo# git add .
root@ubuntu:/home/aakash/Desktop/demo#
```



```
root@ubuntu:/home/aakash/Desktop/demo# git commit -m "First Commit"
[master (root-commit) 0a98db6] First Commit
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 demofile.txt
root@ubuntu:/home/aakash/Desktop/demo#
```

- ❖ **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.



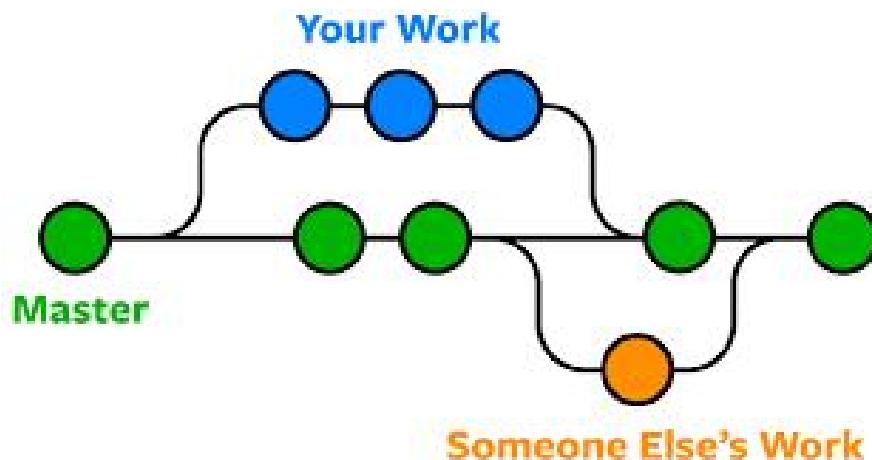
```
root@ubuntu:/home/aakash/Desktop/demo# git log
commit 0a98db6d6c4eb62564dbb88283ef6b02ccfd6c3f (HEAD -> master)
Author: A2v10 <heya2v10@gmail.com>
Date:   Tue Apr 12 19:44:45 2022 +0530

  First Commit
root@ubuntu:/home/aakash/Desktop/demo#
```

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 →

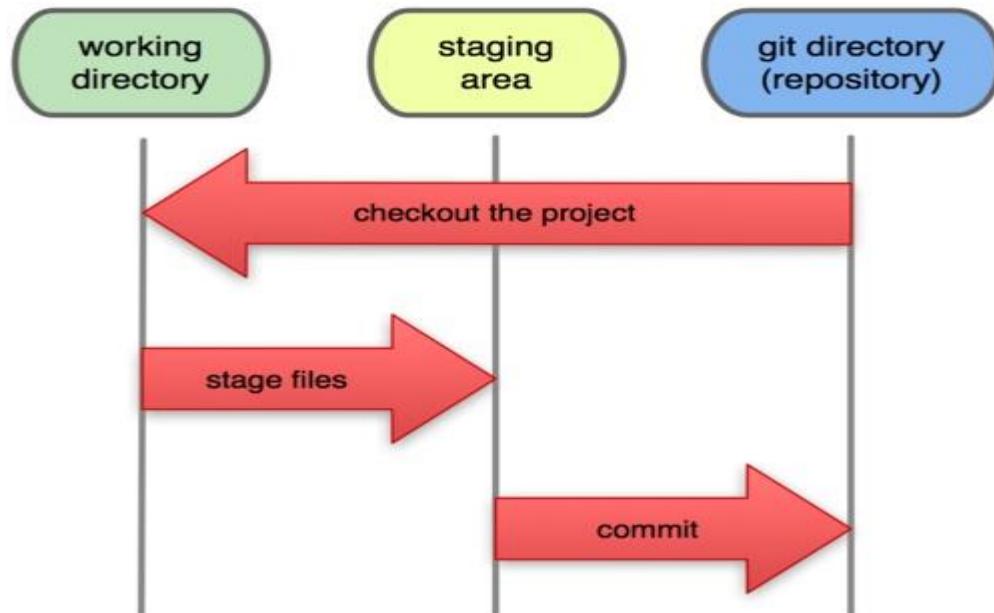
git merge branchname [use to merge branch]

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 our 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)**

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		

INDEX

S. No.	Title
1	Add collaborators on GitHub Repo
2	Fork and Commit
3	Merge and Resolve conflicts created due to own activity and collaborators activity.
4	Reset and Revert

ADD COLLABORATORS ON GITHUB REPO

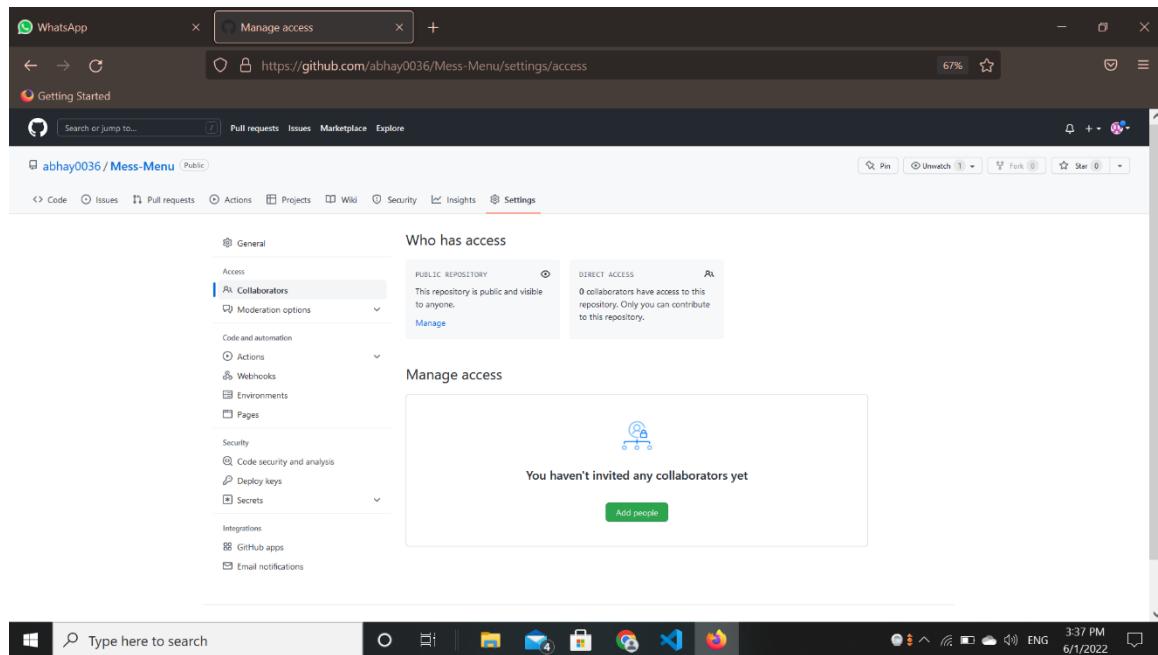
In GitHub, we can invite other GitHub users to become collaborators to our private repositories (which expires after 7 days if not accepted, restoring any unclaimed licenses). Being a collaborator, of a personal repository you can pull (read) the contents of the repository and push (write) changes to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

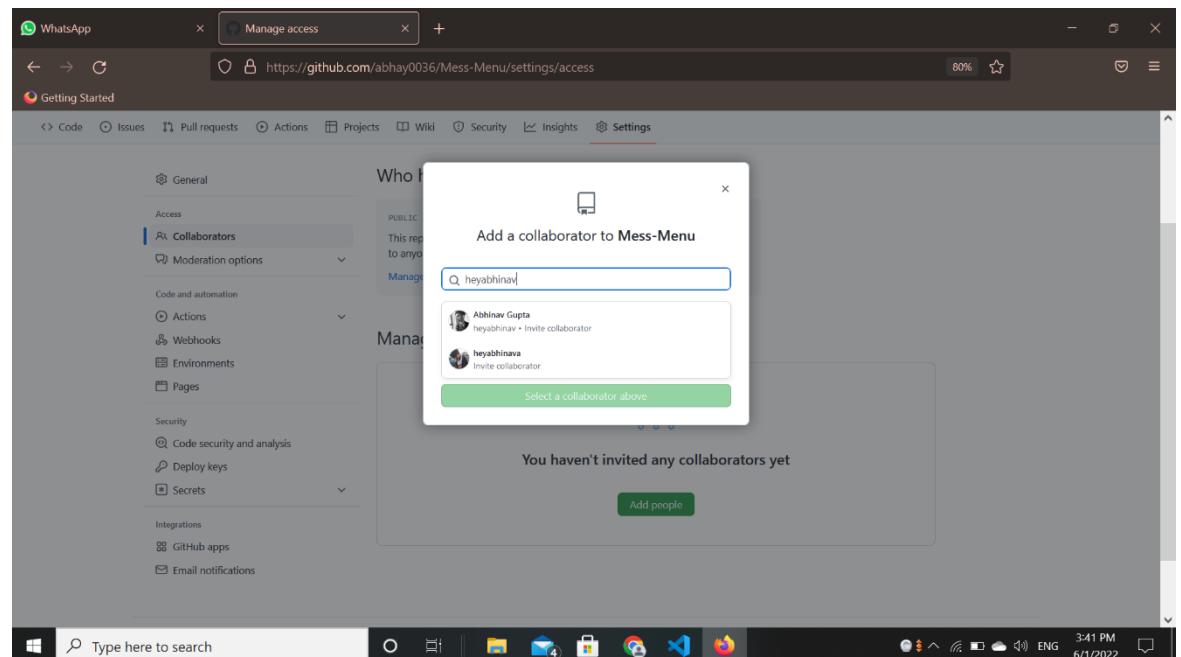
1. Create, merge, and close pull requests in the repository
2. Publish, view, install the packages
3. Fork the repositories
4. Make the changes on the repositories as suggested by the Pull requests.
5. Mark issues or pull requests as duplicate
6. Create, edit, and delete any comments on commits, pull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

STEPS TO ADD COLLABORATORS:

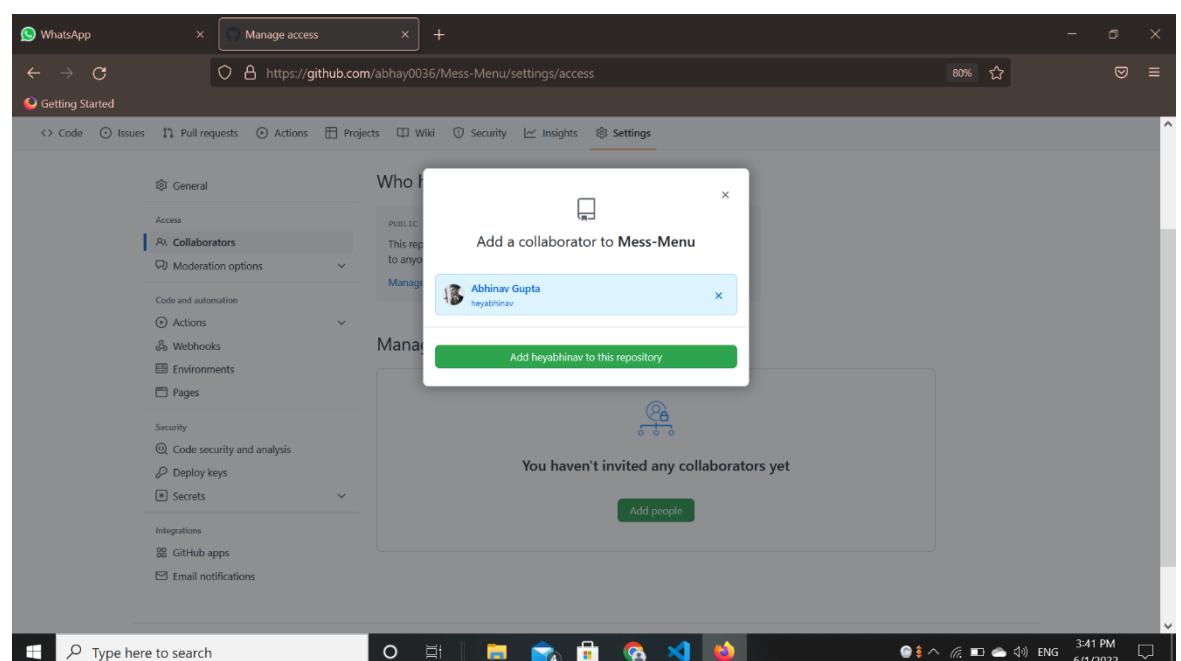
1. Navigate to the repository on Github you wish to share with your collaborator.
2. Click on the "Settings" tab on the right side of the menu at the top of the screen.
3. On the new page, click the "Collaborators" menu item on the left side of the page.



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.

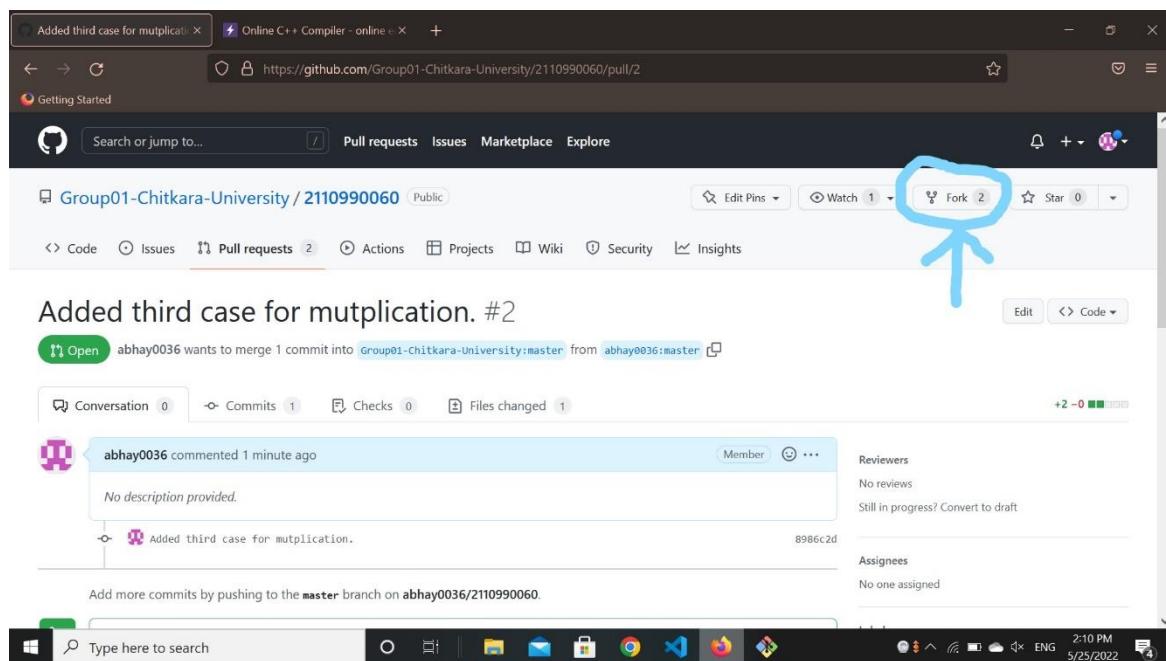




FORK AND COMMIT

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.

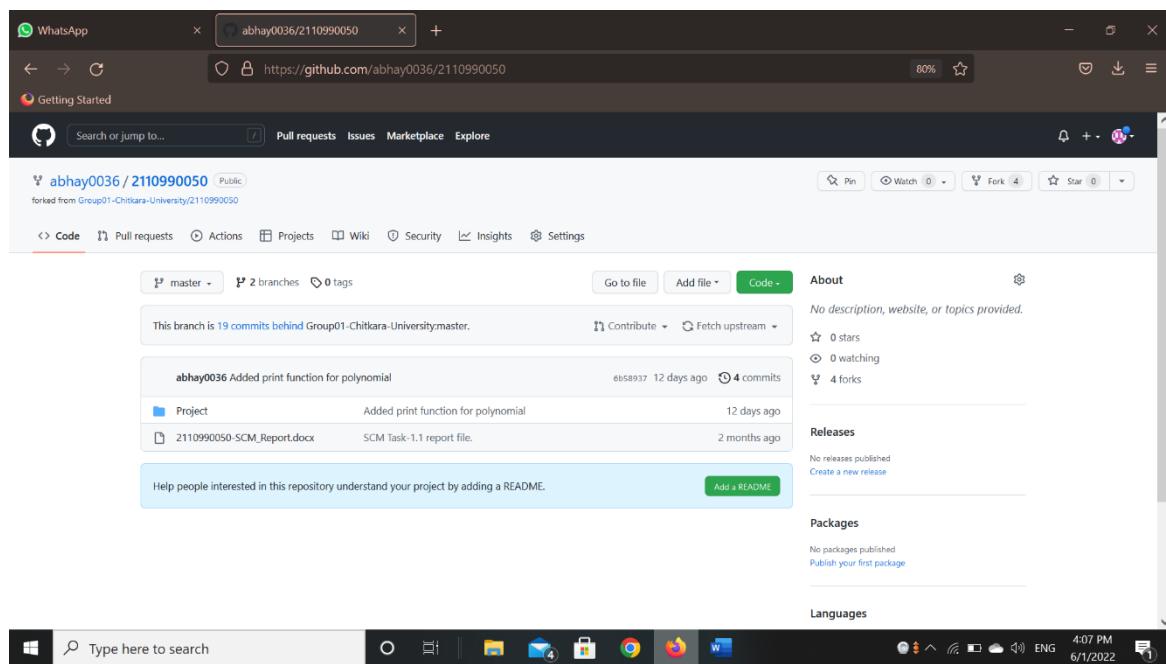
1. Once you have forked the repository, you can clone it into your computer using directly the option given on github or through running git clone command in git bash.

```
MINGW64 /d/forked-repo/2110990050
acer@ZI-471 MINGW64 /d
$ mkdir forked-repo
acer@ZI-471 MINGW64 /d
$ cd forked-repo
acer@ZI-471 MINGW64 /d/forked-repo
$ git clone https://github.com/abhay0036/2110990050.git
Cloning into '2110990050...'...
remote: Enumerating objects: 11, done.
remote: Counting objects: 100% (11/11), done.
remote: Compressing objects: 100% (8/8), done.
remote: Total 11 (delta 2), reused 10 (delta 1), pack-reused 0
Receiving objects: 100% (11/11), 1001.27 KiB | 419.00 KiB/s, done.
Resolving deltas: 100% (2/2), done.
acer@ZI-471 MINGW64 /d/forked-repo
$ ls
2110990050/
acer@ZI-471 MINGW64 /d/forked-repo
$ cd 2110990050
acer@ZI-471 MINGW64 /d/forked-repo/2110990050 (master)
$ git branch
* master
acer@ZI-471 MINGW64 /d/forked-repo/2110990050 (master)
$ git branch abhay0036
acer@ZI-471 MINGW64 /d/forked-repo/2110990050 (master)
$ git checkout abhay0036
Switched to branch 'abhay0036'
acer@ZI-471 MINGW64 /d/forked-repo/2110990050 (abhay0036)
$ git status
On branch abhay0036
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:   Project/Polynomials.cpp

no changes added to commit (use "git add" and/or "git commit -a")
acer@ZI-471 MINGW64 /d/forked-repo/2110990050 (abhay0036)
$ git add .
acer@ZI-471 MINGW64 /d/forked-repo/2110990050 (abhay0036)
```

2. Copy the URL of the forked repository

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



COMMITTING 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.

```
MINGW64/d/forked-repo/2110990060
$ ls
2110990050/  2110990066/

acer@Z1-471 MINGW64 /d/forked-repo
$ git clone https://github.com/abhay0036/2110990060.git
Cloning into '2110990060'...
remote: Enumerating objects: 15, done.
remote: Counting objects: 100% (15/15), done.
remote: Compressing objects: 100% (9/9), done.
remote: Total 15 (delta 2), reused 15 (delta 2), pack-reused 0
Receiving objects: 100% (15/15), 1.59 MiB | 203.00 KiB/s, done.
Resolving deltas: 100% (2/2), done.

acer@Z1-471 MINGW64 /d/forked-repo
$ ls
2110990050/  2110990060/  2110990066/

acer@Z1-471 MINGW64 /d/forked-repo
$ cd 2110990060

acer@Z1-471 MINGW64 /d/forked-repo/2110990060 (master)
$ git branch abhay0036

acer@Z1-471 MINGW64 /d/forked-repo/2110990060 (master)
$ git checkout abhay0036
Switched to branch 'abhay0036'

Type here to search  O  ⊞  ⊞  ⊞  ⊞  ⊞  ⊞  ⊞  ⊞  ⊞  ⊞  ENG  2:09 PM
5/25/2022 4
```

```
MINGW64/d/forked-repo/2110990060
$ git status
On branch abhay0036
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:   project/calculator.cpp

no changes added to commit (use "git add" and/or "git commit -a")

acer@Z1-471 MINGW64 /d/forked-repo/2110990060 (abhay0036)
$ git add .

acer@Z1-471 MINGW64 /d/forked-repo/2110990060 (abhay0036)
$ git commit -m "Added third case for multiplication."
[abhay0036 8986c2d] Added third case for multiplication.
 1 file changed, 2 insertions(+)

acer@Z1-471 MINGW64 /d/forked-repo/2110990060 (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), 405 bytes | 45.00 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/abhay0036/2110990060/pull/new/abhay0036
remote:
To https://github.com/abhay0036/2110990060.git

Type here to search  O  ⊞  ⊞  ⊞  ⊞  ⊞  ⊞  ⊞  ⊞  ⊞  ⊞  ENG  2:10 PM
5/25/2022 4
```



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.

```
MINGW64:/c/Users/acer/Desktop/2110990036/SCM Project
Switched to a new branch 'Abhi1223Gupta-master'
acer@Z1-471 MINGW64 ~/Desktop/2110990036/SCM Project (Abhi1223Gupta-master)
$ git pull https://github.com/Abhi1223Gupta/2110990036.git master
remote: Enumerating objects: 7, done.
remote: Counting objects: 100% (7/7), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 4 (delta 0), reused 4 (delta 0), pack-reused 0
Unpacking objects: 100% (4/4), 1.12 KiB | 2.00 KiB/s, done.
From https://github.com/Abhi1223Gupta/2110990036
 * branch      master    -> FETCH_HEAD
Auto-merging SCM Project/Messmenu.html
CONFLICT (content): Merge conflict in SCM Project/Messmenu.html
Automatic merge failed; fix conflicts and then commit the result.

acer@Z1-471 MINGW64 ~/Desktop/2110990036/SCM Project (Abhi1223Gupta-master|MERGING)
$ git mergetool

This message is displayed because 'merge.tool' is not configured.
See 'git mergetool --tool-help' or 'git help config' for more details.
'git mergetool' will now attempt to use one of the following tools:
opendiff kdiff3 tkdiff xxdiff meld tortoisemerge gvimdiff diffuse diffmerge ecmmerge p4merge araxis bc cod
ecompare smerge emerge vimdiff nvimdiff
Merging:
SCM Project/Messmenu.html

Normal merge conflict for 'SCM Project/Messmenu.html':
{local}: modified file
{remote}: modified file
Hit return to start merge resolution tool (vimdiff): |
```



```
    return 0;
}
//break and continue statement
<<<<< HEAD
//sum of two numbers
//a+b
//sum=a+b
=====
#include<iostream>
using namespace std;
int main(){
int a,b;
cin>>a>>b;
int sum;
sum=a+b;
cout<<"sum"<<sum;
cout<<"hello world";
return 0;
>>>> feature
day2.cpp [dos] (20:47 22/05/2022)
-- INSERT --
```

```
    cout<<endl;
    cin>>n;
}while(n>0);
return 0;
}
//break and continue statement
<<<<< HEAD
//sum of two numbers
//a+b
//sum=a+b
=====
#include<iostream>
using namespace std;
int main(){
int a,b;
cin>>a>>b;
int sum;
sum=a+b;
cout<<"sum"<<sum;
cout<<"hello world";
return 0;
>>>> feature
day2.cpp [dos] (20:48 22/05/2022)
"day2.cpp" [dos] 59L, 91LB
```

59

```
    <<<<< HEAD
//break and continue statement
<<<<< HEAD
//sum of two numbers
//a+b
//sum=a+b
=====
#include<iostream>
using namespace std;
int main(){
int a,b;
cin>>a>>b;
int sum;
sum=a+b;
cout<<"sum"<<sum;
cout<<"hello world";
return 0;
>>>> feature
day2.cpp [dos] (20:47 22/05/2022)
-- INSERT --
```

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.



```
MINGW64:/d/forked-repo/2110990050
$ git add .
acer@ZI-471 MINGW64 /d/forked-repo/2110990050 (abhay0036)
$ git commit -m "Added print function for polynomial"
[abhay0036 6b58937] Added print function for polynomial
1 file changed, 9 insertions(+)
acer@ZI-471 MINGW64 /d/forked-repo/2110990050 (abhay0036)
$ git status
On branch abhay0036
nothing to commit, working tree clean
acer@ZI-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: 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@ZI-471 MINGW64 /d/forked-repo/2110990050 (abhay0036)
$ git checkout master
Switched to branch 'master'
Your branch is up to date with 'origin/master'.
acer@ZI-471 MINGW64 /d/forked-repo/2110990050 (master)
$ git merge abhay0036
Updating 264a54a..6b58937
Fast-Forward
 Project/Polynomials.cpp | 9 ++++++-
 1 file changed, 9 insertions(+)
acer@ZI-471 MINGW64 /d/forked-repo/2110990050 (master)
$ git push origin master
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/abhay0036/2110990050.git
 264a54a..6b58937 master -> master
acer@ZI-471 MINGW64 /d/forked-repo/2110990050 (master)
```

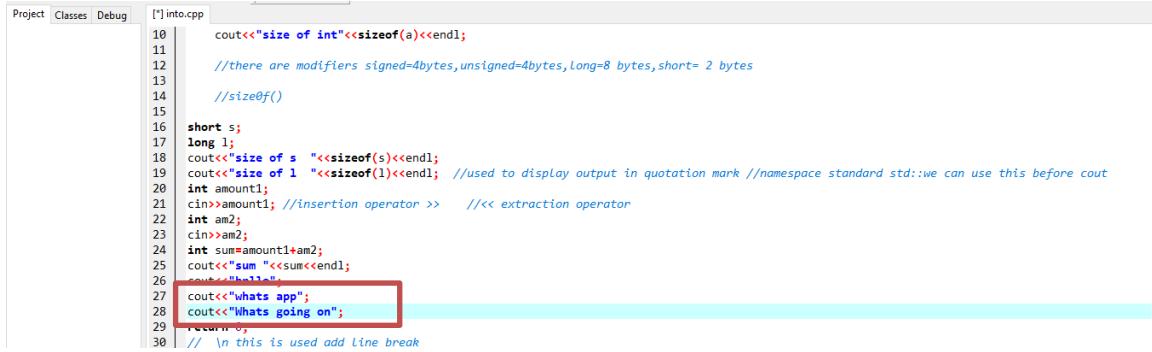
RESET AND REVERT

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.

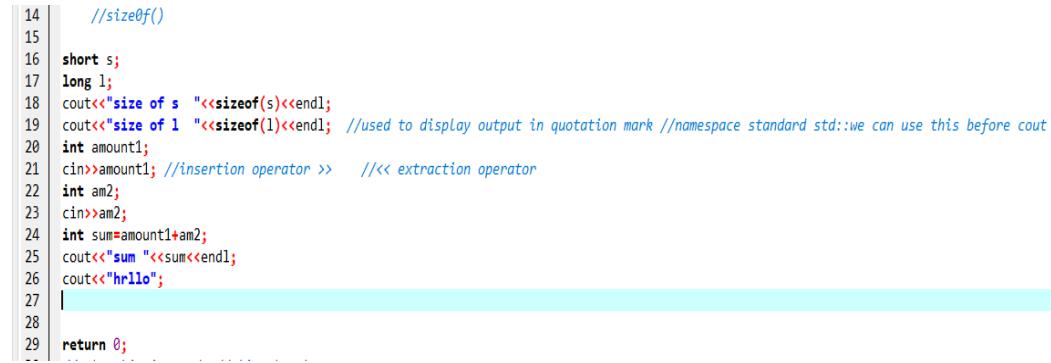
Suppose we make edits to a file, stage it and commit it



```
[*] into.cpp
10     cout<<"size of int"<<sizeof(a)<<endl;
11
12 //there are modifiers signed=4bytes,unsigned=4bytes,Long=8 bytes,short= 2 bytes
13
14 //sizeOf()
15
16 short s;
17 long l;
18 cout<<"size of s "<<sizeof(s)<<endl;
19 cout<<"size of l "<<sizeof(l)<<endl; //used to display output in quotation mark //namespace standard std::we can use this before cout
20 int amount1;
21 cin>>amount1; //insertion operator >> //<< extraction operator
22 int am2;
23 cin>>am2;
24 int sum=amount1+am2;
25 cout<<"sum "<<sum<<endl;
26 cout<<"_sum13";
27 cout<<"\"whats app\"";
28 cout<<"\"whats going on\"";
29
30 return 0;
// \n this is used add line break
```



```
MINGW64:/c/Users/acer/Desktop/SCM
acer@ZI-471 MINGW64 ~/Desktop/SCM (main)
$ git log
commit 17eb4bf6e75061b0911a38f12dbf9e60920dfb9 (HEAD -> main)
author: abhay0036 <abhay0036.be21@chitkara.edu.in>
date: Sun May 22 14:39:54 2022 +0530
    My Project
acer@ZI-471 MINGW64 ~/Desktop/SCM (main)
$ |
```



```
14 //sizeOf()
15
16 short s;
17 long l;
18 cout<<"size of s "<<sizeof(s)<<endl;
19 cout<<"size of l "<<sizeof(l)<<endl; //used to display output in quotation mark //namespace standard std::we can use this before cout
20 int amount1;
21 cin>>amount1; //insertion operator >> //<< extraction operator
22 int am2;
23 cin>>am2;
24 int sum=amount1+am2;
25 cout<<"sum "<<sum<<endl;
26 cout<<"\"hrllo";
27
28
29 return 0;
```

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.



```
MINGW64:/c/Users/acer/Desktop/SCM/2110990036
$ git reset
acer@ZI-471 MINGW64 ~/Desktop/SCM/2110990036 (master)
$ git log
commit 5178af4f4222a21ab3beb4fc89326c9659ddb5 (HEAD -> master, origin/master,
origin/HEAD)
Author: abhay <abhay0036.be21@chitkara.edu.in>
Date: Tue Apr 12 20:05:31 2022 +0530

Task 1.1 File SCM
acer@ZI-471 MINGW64 ~/Desktop/SCM/2110990036 (master)
$ |
```

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.

```
MINGW64:/c/Users/HP/Desktop/c++/apna c++
    }
    int space=2*n-2*i;
    for(int j=i;j<=space;j++){
        cout<<" ";
    }
    for(int j=i;j<=i;j++){
        cout<<"*";
    }
    cout<<endl;
}
for(int i=n;i>1;i--){
    for(int j=1;j<=i;j++){
        cout<<"*";
    }
    int space=2*n-2*i;
    for(int j=1;j<=space;j++){
        cout<<" ";
    }
    for(int j=1;j<=i;j++){
        cout<<"*";
    }
    cout<<endl;
}
}
}
}

#include <iostream>
using namespace std;
int main(){
int n;
cin>>n;
for( int i=1;i<=n;i++){
    for( int j=1;j<=i;j<=n-i;j++){
        cout<<j<<" ";
    }
    cout<<endl;
}
cout<<"Hello world";
return 0;
}
```

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



```
MINGW64:/c/Users/acer/Desktop/SCM/2110990036
acer@ZI-471 MINGW64 ~/Desktop/SCM/2110990036 (master)
$ git add .
acer@ZI-471 MINGW64 ~/Desktop/SCM/2110990036 (master)
$ git commit -m "second"
[master ceb5708] second
1 file changed, 32 insertions(+)
create mode 100644 SCM Project/Messmenu.html
acer@ZI-471 MINGW64 ~/Desktop/SCM/2110990036 (master)
$ git log
commit ceb57089705c3136ecbc1a62e70d9caiceab46 (HEAD -> master)
Author: abhay0036 <abhay0036.be21@chitkara.edu.in>
Date:   Thu Jun 2 13:33:18 2022 +0530

    second

commit 5178af4f4222a21ab3beb4f4c89326c9659dedb5 (origin/master, origin/HEAD)
Author: abhay <abhay0036.be21@chitkara.edu.in>
Date:   Tue Apr 12 20:05:31 2022 +0530

    Task 1.1 File SCM

acer@ZI-471 MINGW64 ~/Desktop/SCM/2110990036 (master)
$ |
```

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

```
MINGW64:/c/Users/acer/Desktop/SCM/2110990036
Task 1.1 File SCM
acer@ZI-471 MINGW64 ~/Desktop/SCM/2110990036 (master)
$ git status
On branch master
Your branch is ahead of 'origin/master' by 1 commit.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean
acer@ZI-471 MINGW64 ~/Desktop/SCM/2110990036 (master)
$ git log
commit ceb57089705c3136ecbc1a62e70d9caiceab46 (HEAD -> master)
Author: abhay0036 <abhay0036.be21@chitkara.edu.in>
Date:   Thu Jun 2 13:33:18 2022 +0530

    second

commit 5178af4f4222a21ab3beb4f4c89326c9659dedb5 (origin/master, origin/HEAD)
Author: abhay <abhay0036.be21@chitkara.edu.in>
Date:   Tue Apr 12 20:05:31 2022 +0530

    Task 1.1 File SCM

acer@ZI-471 MINGW64 ~/Desktop/SCM/2110990036 (master)
$ git add .
Nothing specified, nothing added.
hint: Maybe you wanted to say 'git add .'?
hint: Turn this message off by running
hint: "git config advice.addEmptyPathspec false"
acer@ZI-471 MINGW64 ~/Desktop/SCM/2110990036 (master)
$ git ls
git: 'ls' is not a git command. See 'git --help'.

The most similar command is
  lfs

acer@ZI-471 MINGW64 ~/Desktop/SCM/2110990036 (master)
$ ls
'SCM Project'/  SCM-2110990036.pdf
acer@ZI-471 MINGW64 ~/Desktop/SCM/2110990036 (master)
$ status
bash: status: command not found
acer@ZI-471 MINGW64 ~/Desktop/SCM/2110990036 (master)
$ |
```

```
acer@ZI-471 MINGW64 ~/Desktop/SCM/2110990036 (master)
$ git status
On branch master
Your branch is ahead of 'origin/master' by 1 commit.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean

acer@ZI-471 MINGW64 ~/Desktop/SCM/2110990036 (master)
$ git log
commit 5178af4f4222a21ab3beb4f4c89326c9659dedb5 (HEAD -> master)
Author: abhay0036 <abhay0036_be21@chitkara.edu.in>
Date:  Tue Apr 12 20:05:31 2022 +0530

    second

commit e6b57089705c53136c8c1a162e70d9c1licesb46 (origin/master, origin/HEAD)
Author: abhay0036 <abhay0036_be21@chitkara.edu.in>
Date: Thu Jun 2 13:35:18 2022 +0530

Task 1.2 File SCM

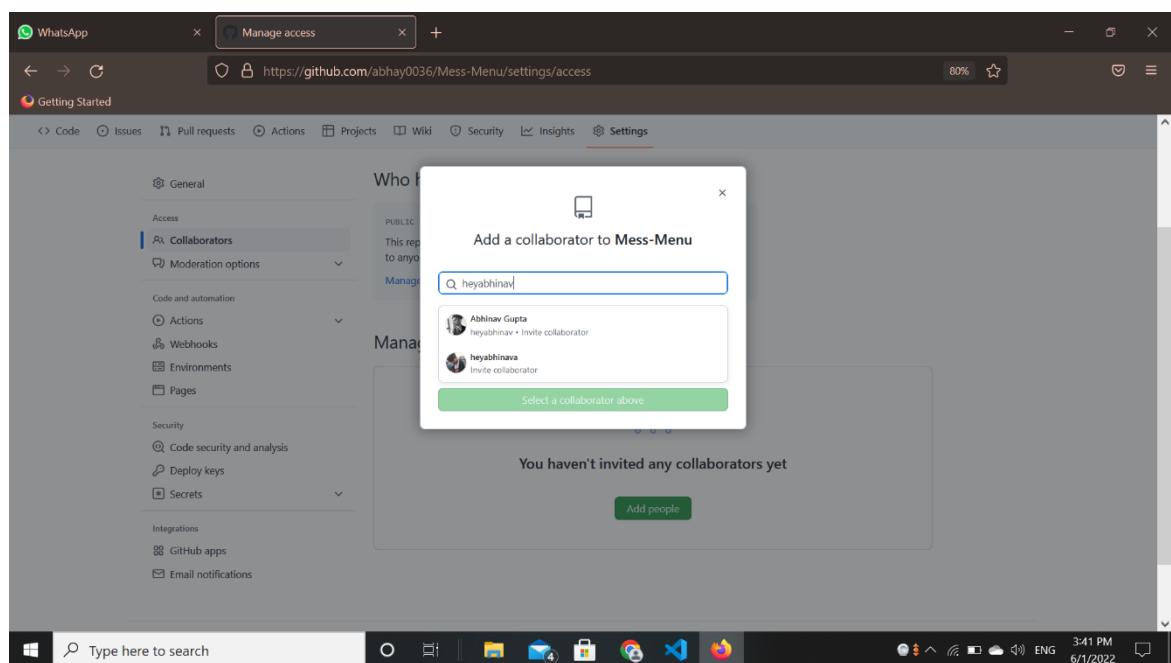
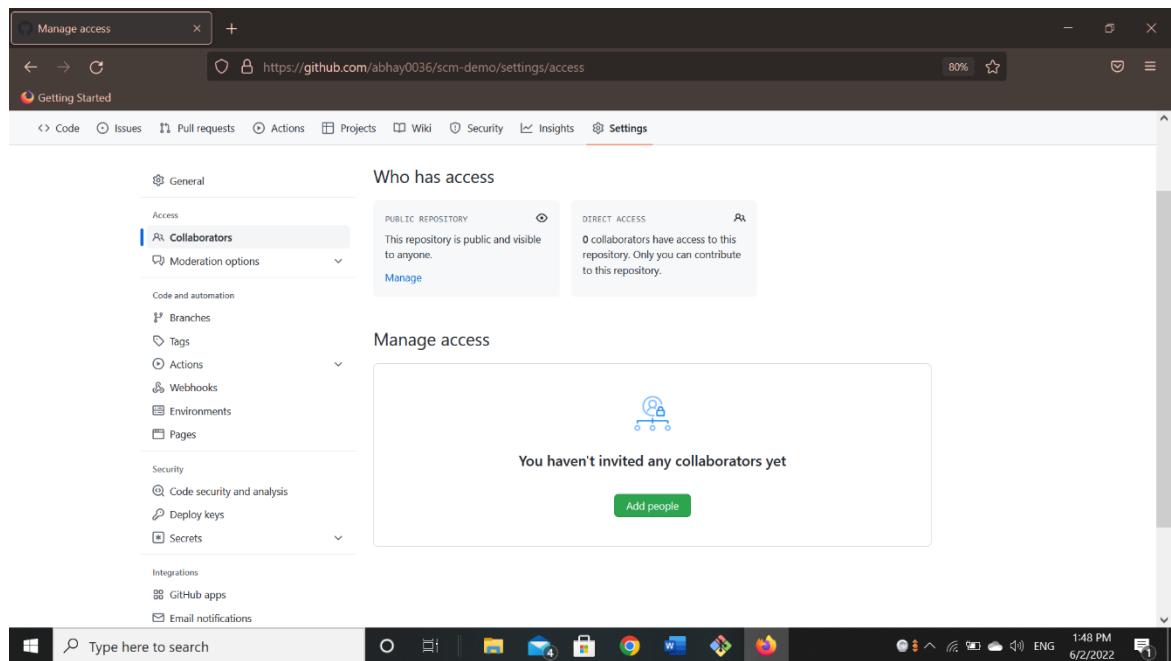
acer@ZI-471 MINGW64 ~/Desktop/SCM/2110990036 (master)
$ git add .
Nothing specified, nothing added.
hint: Maybe you wanted to say git add .?
hint: Turn off this message off by running
hint: "git config advice.addEmptyPathspec false"
```

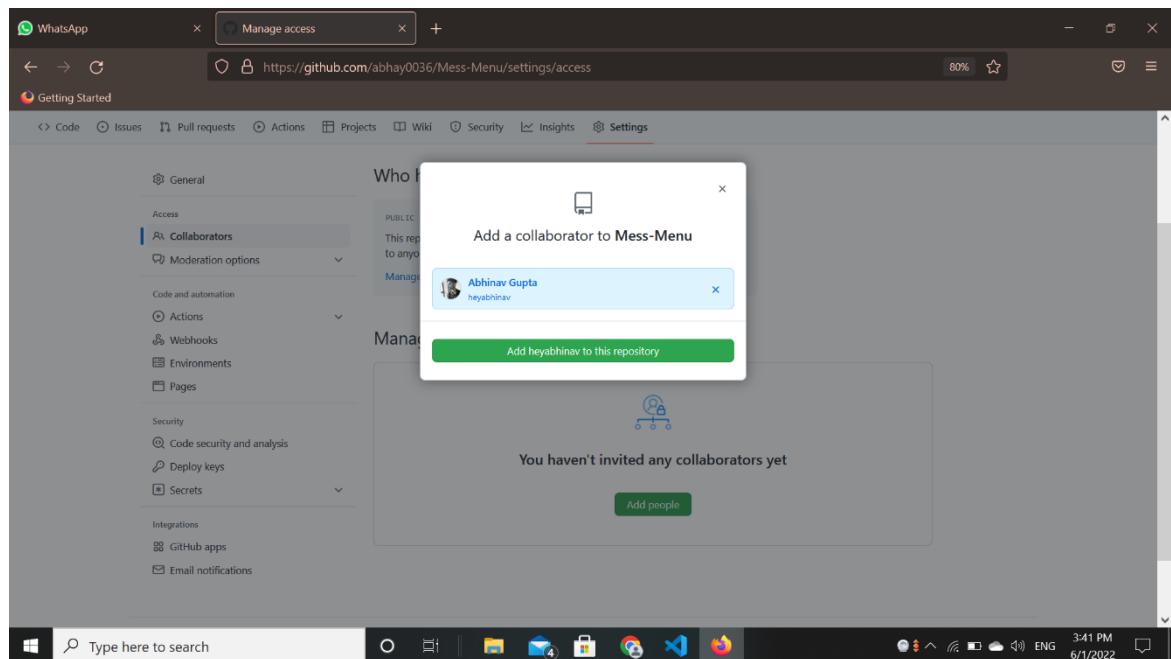
You can see that a new commit as 'revert "changes made"' is there and the file has returned to its previous state.

CREATE A DISTRIBUTED REPOSITORY AND ADD MEMBERS IN PROJECT TEAM

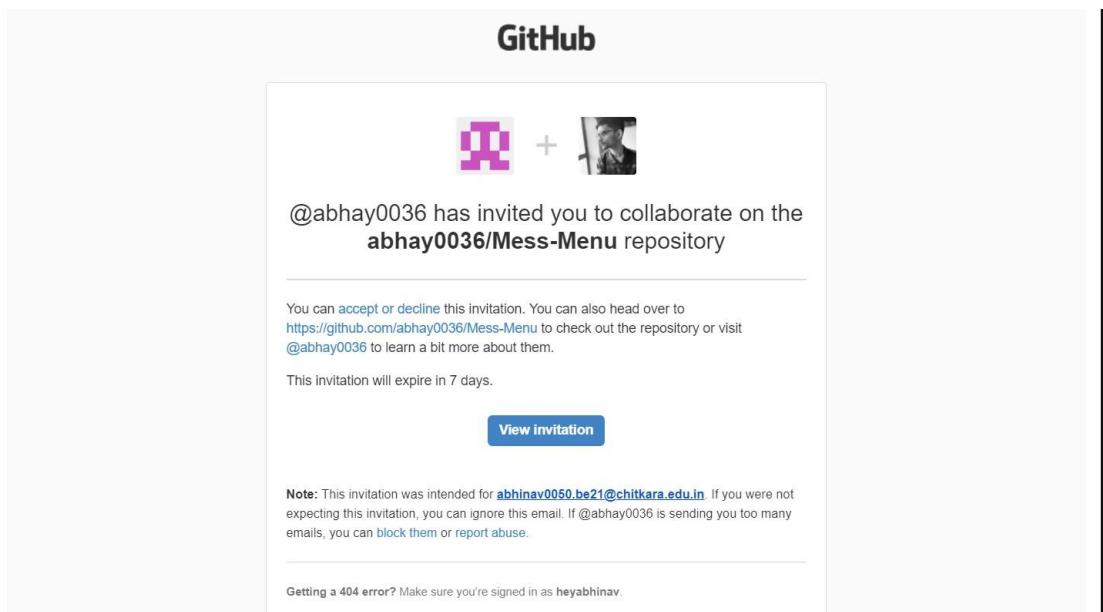
1. On the homepage of your GitHub account, click on Repositories option in the menu bar.
2. Click on the 'New' button in the top right corner.

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.





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.



```
user@ZI-471 MINGW64 /d
$ mkdir forked-repo
user@ZI-471 MINGW64 /d
$ cd forked-repo

user@ZI-471 MINGW64 /d/forked-repo
$ git clone https://github.com/abhay0036/2110990050.git
Cloning into '2110990050'...
remote: Enumerating objects: 11, done.
remote: Counting objects: 100% (11/11), done.
remote: Compressing objects: 100% (8/8), done.
remote: Total 11 (delta 2), reused 10 (delta 1), pack-reused 0
Receiving objects: 100% (11/11), 1001.27 KiB | 419.00 KiB/s, done.
Resolving deltas: 100% (2/2), done.

user@ZI-471 MINGW64 /d/forked-repo
$ ls
2110990050/
user@ZI-471 MINGW64 /d/forked-repo
$ cd 2110990050/
```

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



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

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

Comparing changes

Choose two branches to see what's changed or to start a new pull request. If you need to, you can also compare across forks.

base repository: aanchalsharma1024/cpp ▾ base: master ← head repository: Aarushi2021/cpp-1 ▾ compare: master ▾

✓ Able to merge. These branches can be automatically merged.

Discuss and review the changes in this comparison with others. [Learn about pull requests](#) [Create pull request](#)

→ 1 commit ↗ 1 file changed ↗ 1 contributor

Commits on May 22, 2022

Create README.md [Verified](#) [d731617](#) [🔗](#)

Aarushi2021 committed 1 minute ago

Showing 1 changed file with 2 additions and 0 deletions. [Split](#) [Unified](#)

README.md [🔗](#)

... ... @@ -0,0 +1,2 @@
 1 + # cpp-1
 2 + This is a forked repo

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

Pull requests · Group01-Chitkara

Getting Started

Search or jump to... Pull requests Issues Marketplace Explore

Group01-Chitkara-University / 2110990036 Public

Code Issues Pull requests 1 Actions Projects Wiki Security Insights Settings

Label issues and pull requests for new contributors

Now, GitHub will help potential first-time contributors [discover issues](#) labeled with [good first issue](#). [Dismiss](#)

Filters is:pr is:open

1 Open 0 Closed

Added snacks column, also listed snacks day wise in that column.
#1 opened 15 hours ago by Abhishek0060

ProTip! Notify someone on an issue with a mention, like: @abhay0036.

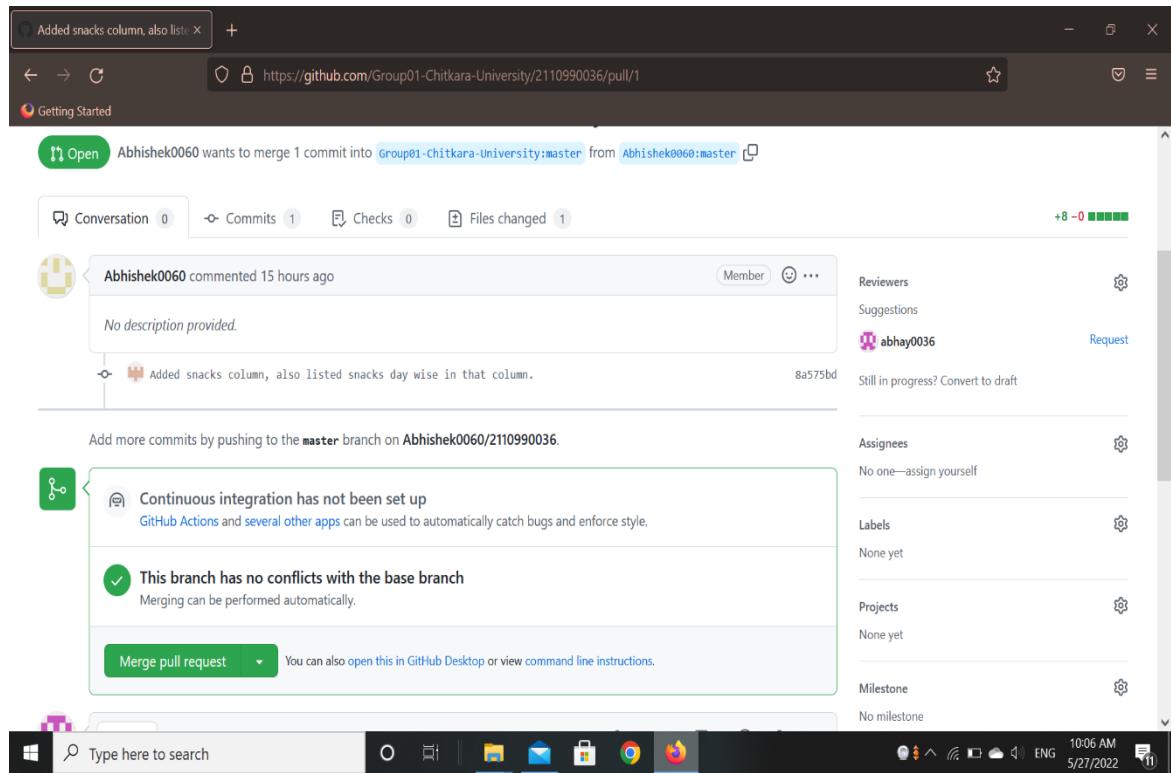


The screenshot shows a GitHub pull request page for a repository named 'Group01-Chitkara-University / 2110990036'. The pull request is titled 'Added snacks column, also listed snacks day wise in that column.' and has been opened by 'Abhishek0060' from the branch 'Abhishek0060:master' into the 'master' branch. The commit message is 'Added snacks column, also listed snacks day wise in that column.' and it was pushed 15 hours ago. The pull request has 8 approvals and 0 reviews. A note at the bottom of the pull request page says 'Continuous integration has not been set up. GitHub Actions and several other apps can be used to automatically catch bugs and enforce style.'

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.

The screenshot shows a GitHub pull request page for a repository named 'aanchalsharma1024/cpp'. The pull request is titled 'aanchalsharma1024 / cpp' and has 1 pull request. The status bar indicates the pull request is 'Closed'.

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.

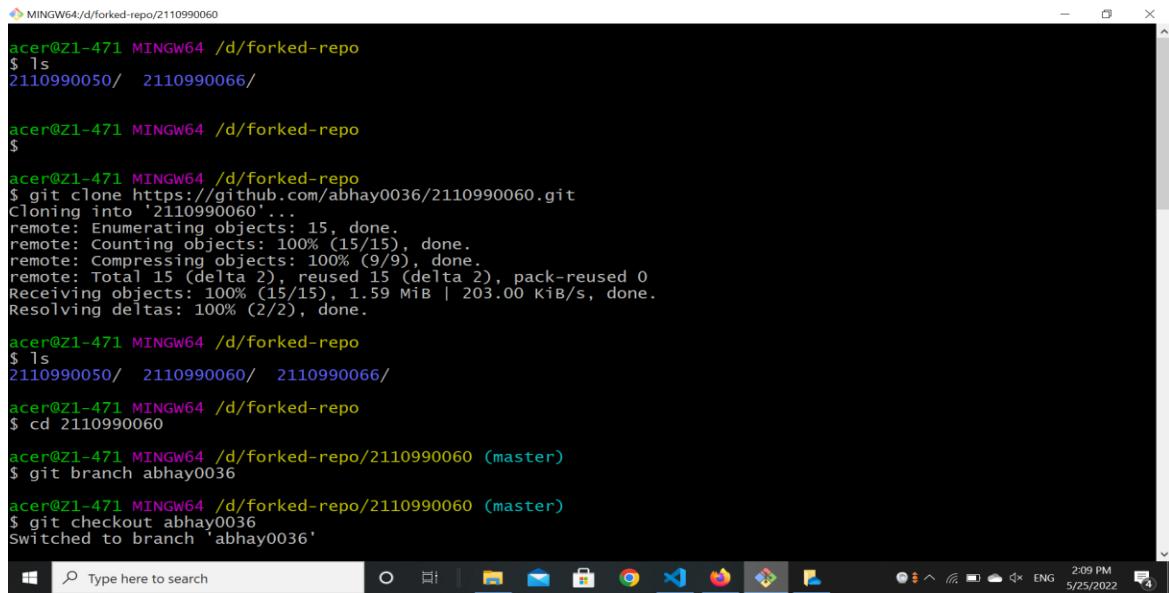


CREATE A PULL REQUEST ON A TEAM MEMBER'S REPO AND CLOSE PULL REQUESTS GENERATED BY TEAM MEMBERS ON OWN REPOSITORY AS A MAINTAINER

#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.



```

MINGW64:/d/forked-repo/2110990060
$ ls
2110990050/ 2110990066/

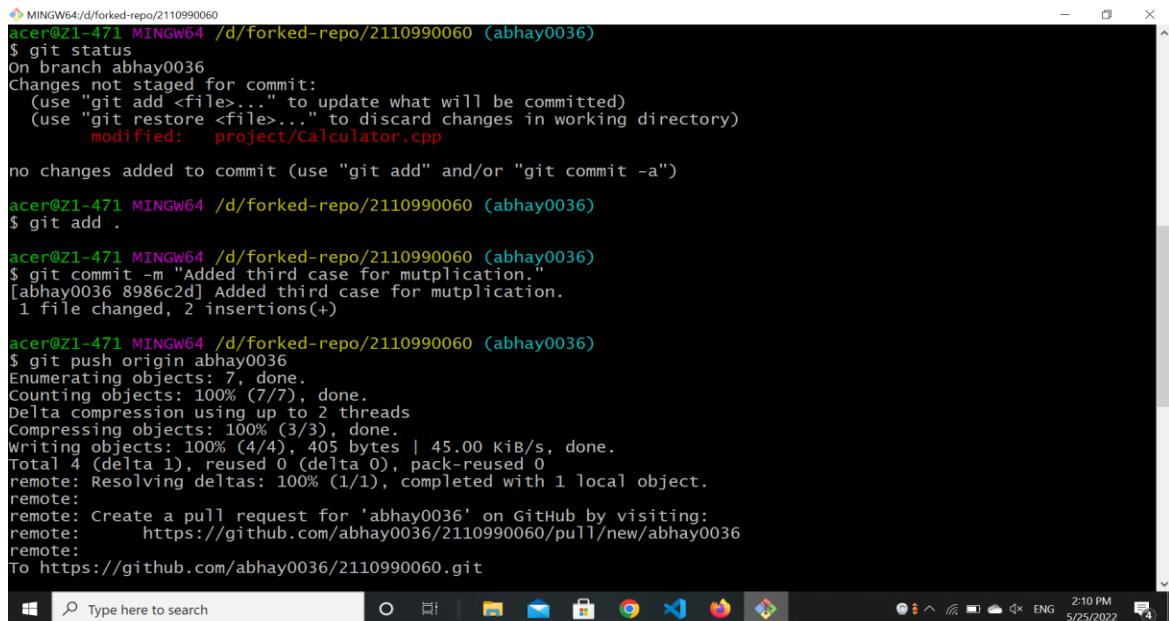
acer@Z1-471 MINGW64 /d/forked-repo
$ 

acer@Z1-471 MINGW64 /d/forked-repo
$ git clone https://github.com/abhay0036/2110990060.git
Cloning into '2110990060'...
remote: Enumerating objects: 15, done.
remote: Counting objects: 100% (15/15), done.
remote: Compressing objects: 100% (9/9), done.
remote: Total 15 (delta 2), reused 15 (delta 2), pack-reused 0
Receiving objects: 100% (15/15), 1.59 MiB | 203.00 KiB/s, done.
Resolving deltas: 100% (2/2), done.

acer@Z1-471 MINGW64 /d/forked-repo
$ ls
2110990050/ 2110990060/ 2110990066/

acer@Z1-471 MINGW64 /d/forked-repo
$ cd 2110990060
acer@Z1-471 MINGW64 /d/forked-repo/2110990060 (master)
$ git branch abhay0036
acer@Z1-471 MINGW64 /d/forked-repo/2110990060 (master)
$ git checkout abhay0036
Switched to branch 'abhay0036'

```



```

MINGW64:/d/forked-repo/2110990060
$ git status
On branch abhay0036
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:   project/calculator.cpp

no changes added to commit (use "git add" and/or "git commit -a")

acer@Z1-471 MINGW64 /d/forked-repo/2110990060 (abhay0036)
$ git add .

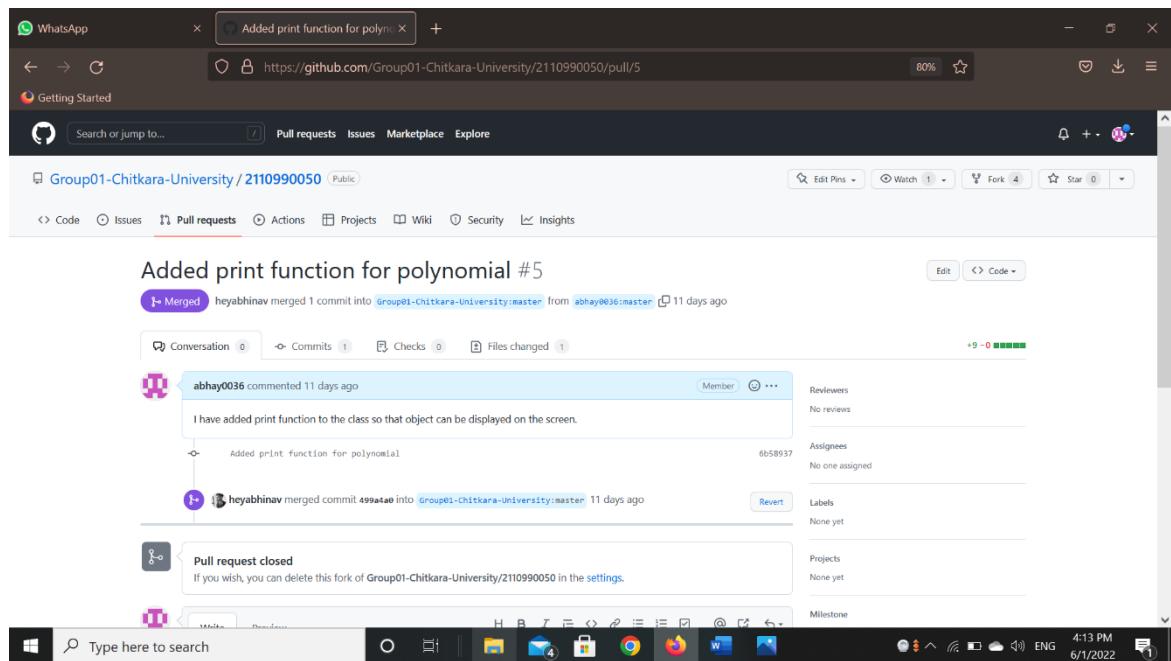
acer@Z1-471 MINGW64 /d/forked-repo/2110990060 (abhay0036)
$ git commit -m "Added third case for multiplication."
[abhay0036 8986c2d] Added third case for multiplication.
 1 file changed, 2 insertions(+)

acer@Z1-471 MINGW64 /d/forked-repo/2110990060 (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), 405 bytes | 45.00 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/abhay0036/2110990060/pull/new/abhay0036
remote:
To https://github.com/abhay0036/2110990060.git

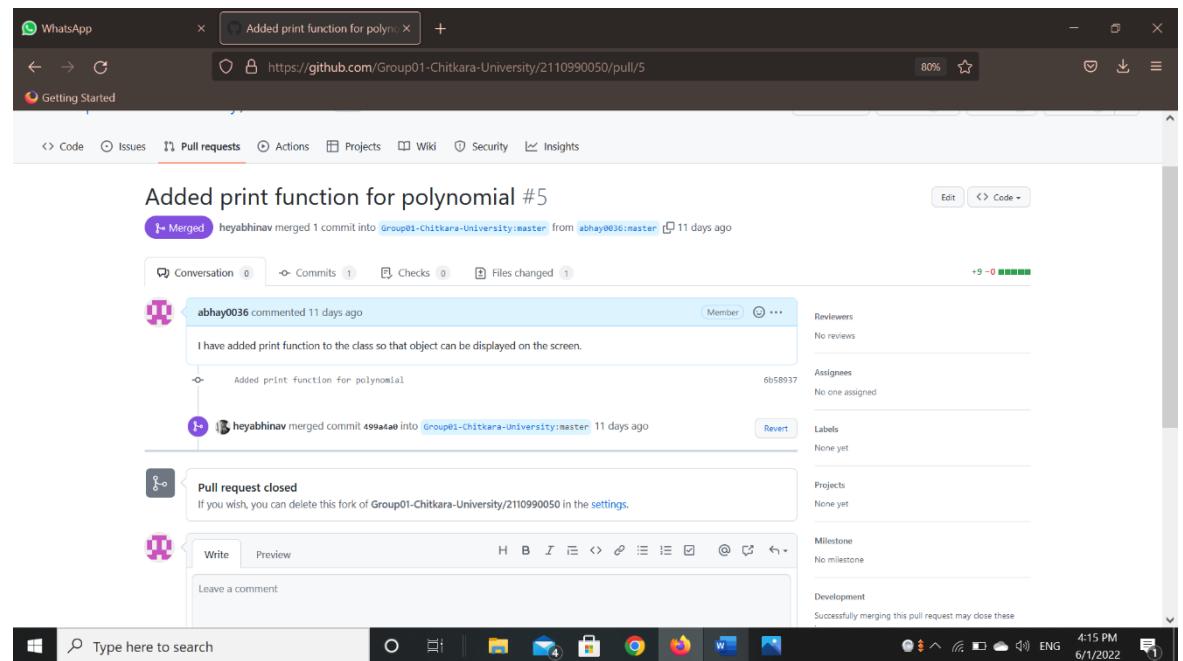
```

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.



The screenshot shows a GitHub pull request page for a repository named "Group01-Chitkara-University". The pull request is titled "Added print function for polynomial #5" and has been merged. The commit message from "abhay0036" states: "I have added print function to the class so that object can be displayed on the screen." The pull request has 1 commit, 0 checks, and 1 file changed. The status bar at the bottom indicates the taskbar, system tray, and system information like battery level and network status.