Subject Name: **Source Code Management**

Subject Code: **CS181**

Cluster: **Beta**

Department: **CSE**

Submitted By-
Aarchi Sharma
2110990012
G01

Submitted To-
Dr. Monit Kapoor

INDEX

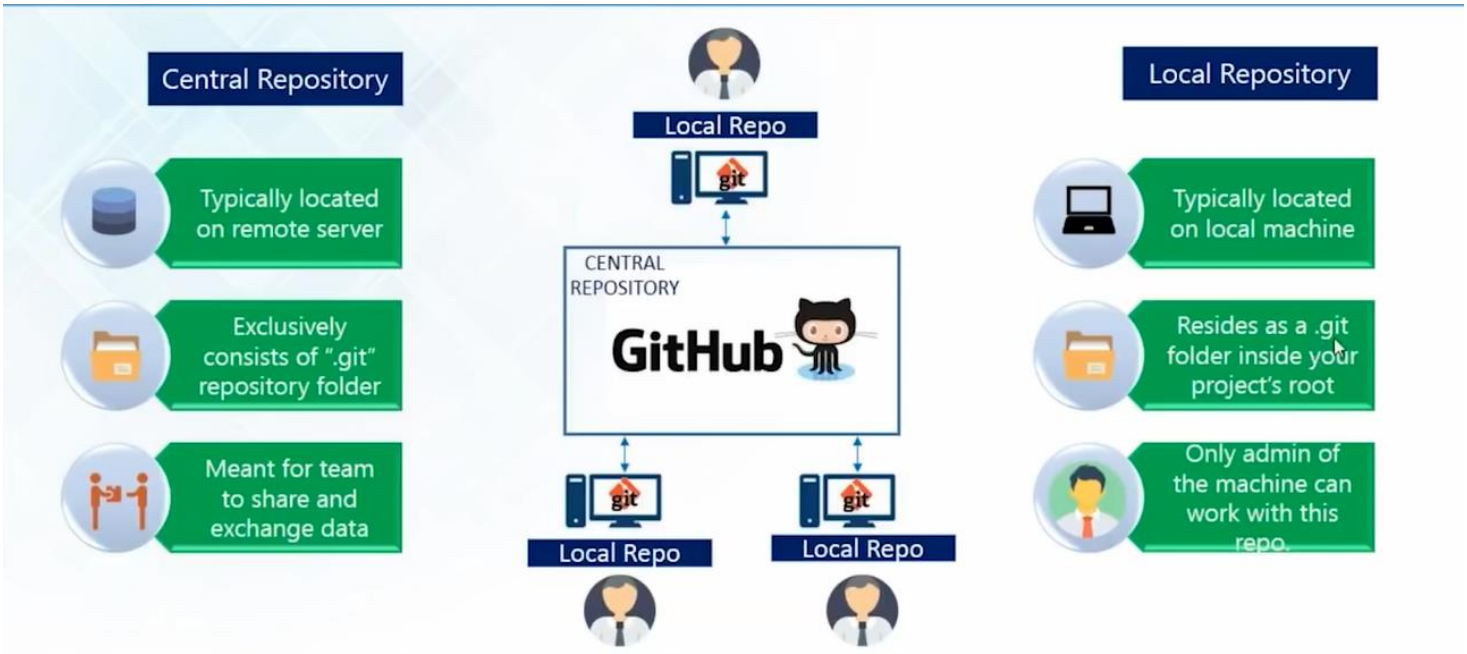
WHAT IS GIT

Git is a free and open-source distributed version control system designed to handle everything from small to very large projects with speed and efficiency. Git is easy to learn and has a tiny footprint with lightning-fast performance. It outclasses SCM tools like Subversion, CVS, Perforce, and ClearCase with features like cheap local branching, convenient staging areas, and multiple workflows.

WHAT IS GITHUB

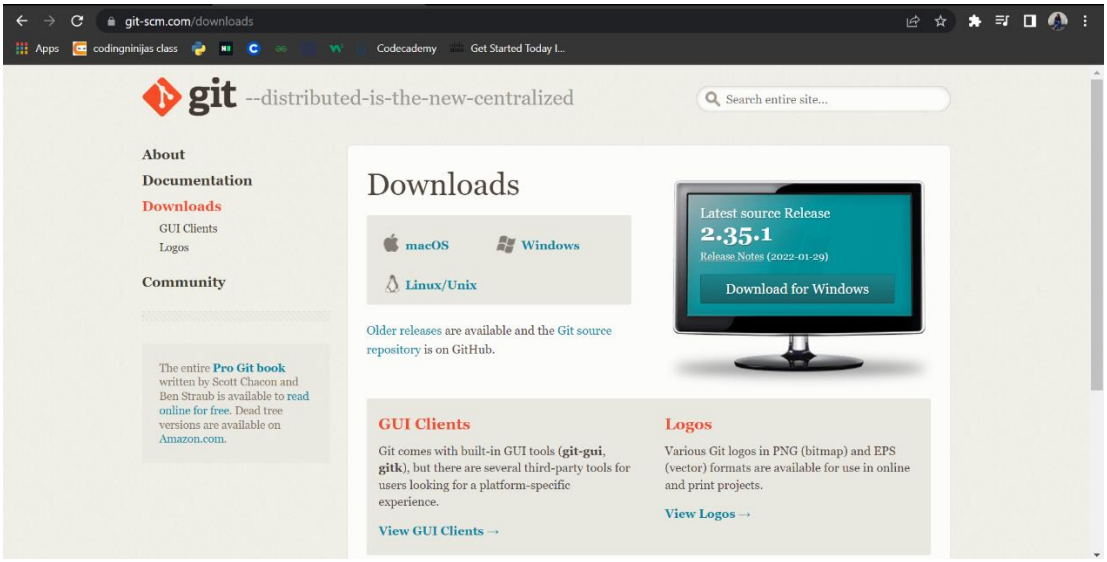
GitHub is a website and cloud-based service that helps developers store and manage their code, as well as track and control changes to their code.

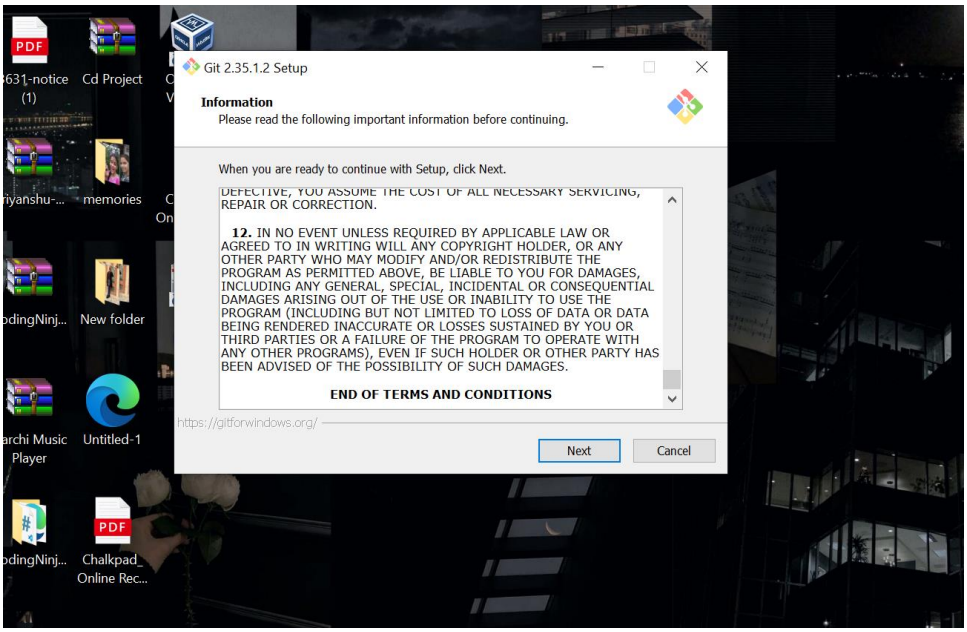
GIT VS GITHUB

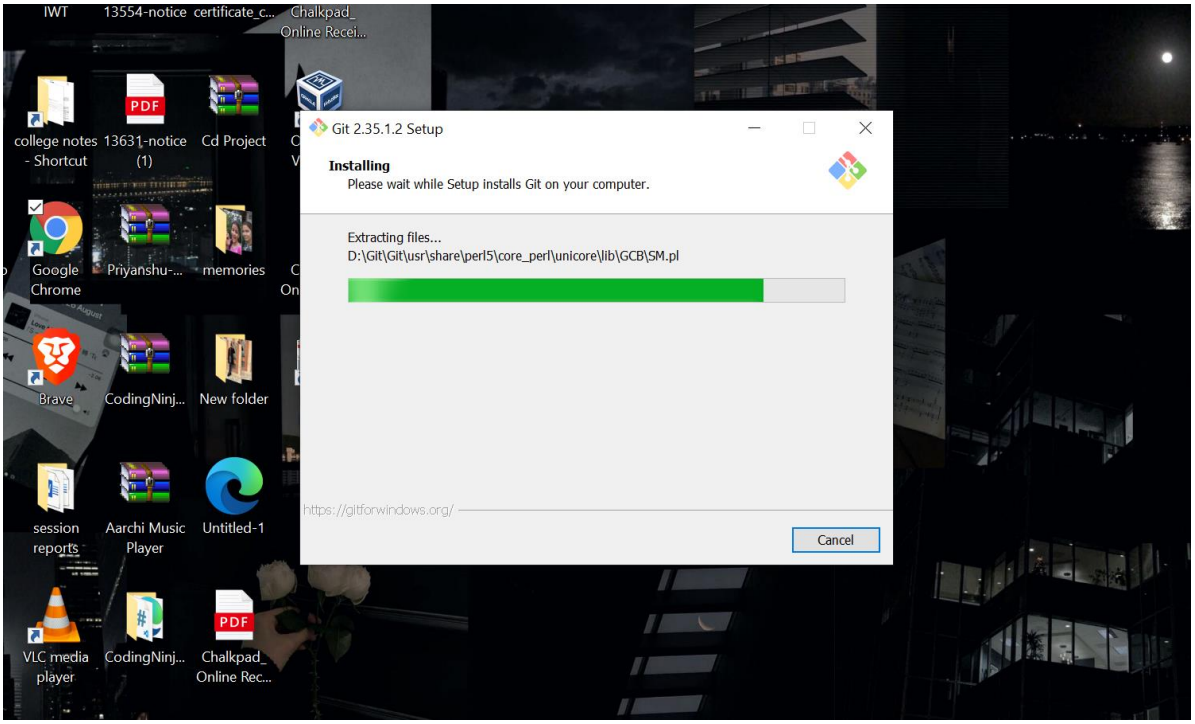


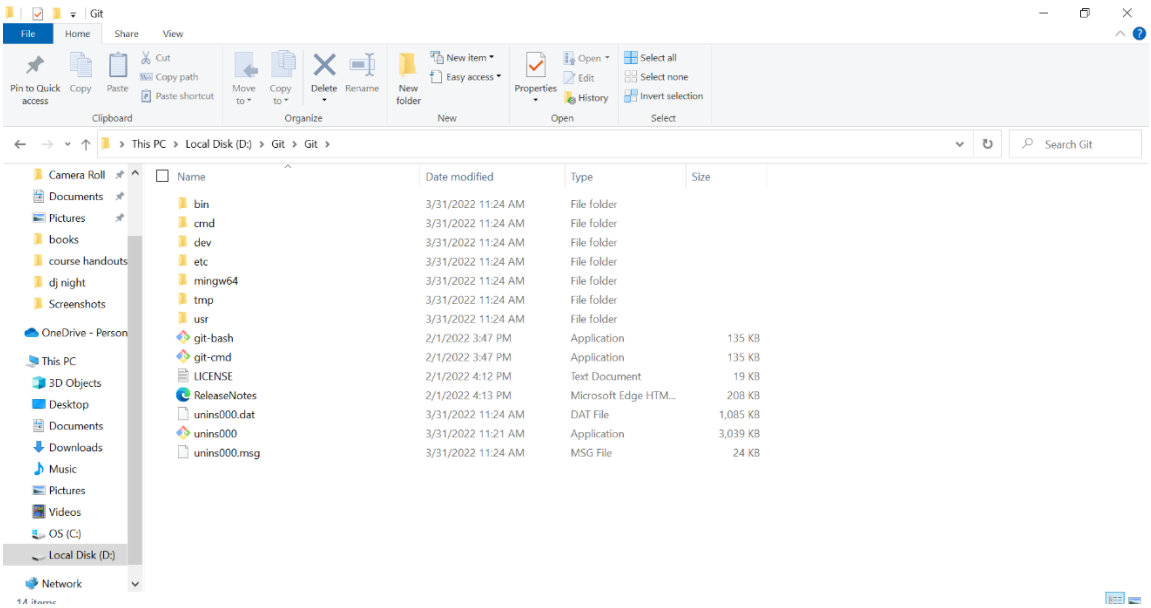
INSTALLING GIT

The most official build is available for download on the Git website. Just go to https://git-scm.com/download/win and the download will start automatically. Follow the steps listed further to proceed with the installation.



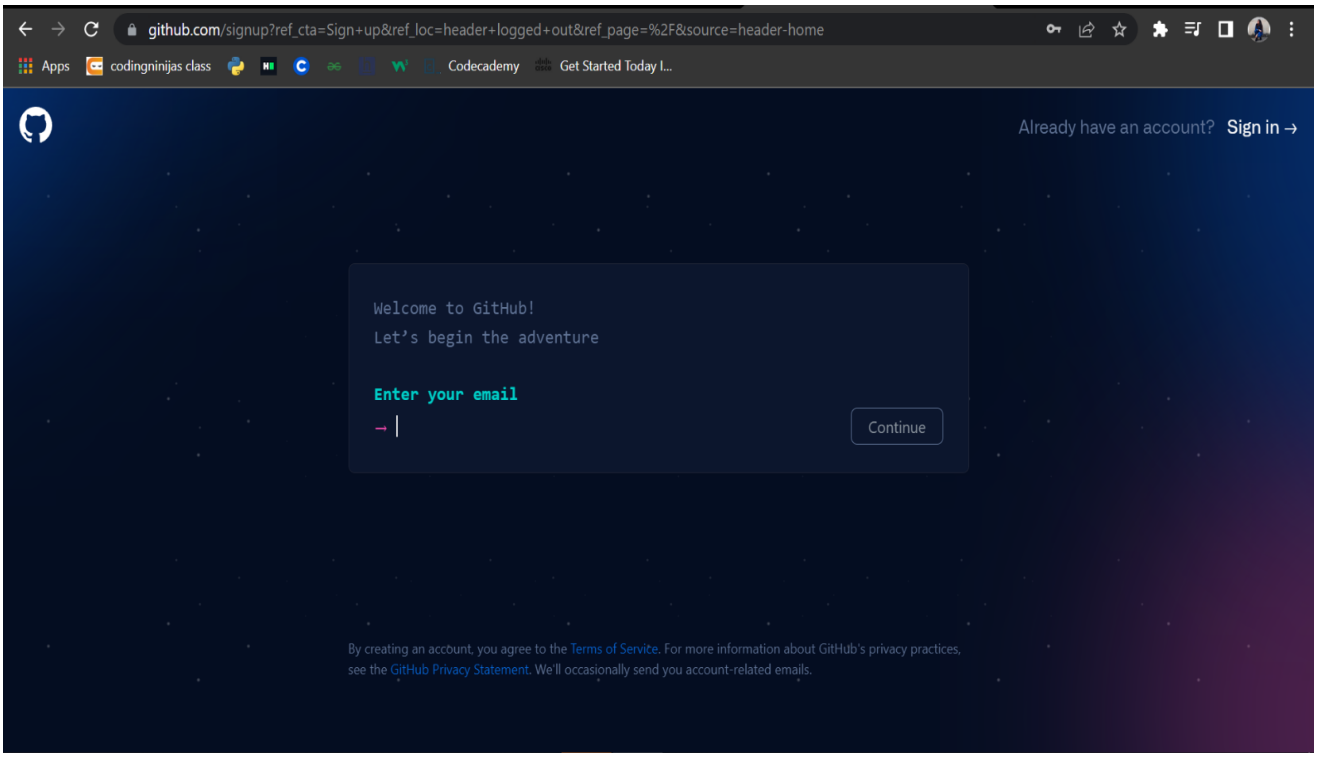



IWT
13554-notice
certificate_c...
Chalkpad_ Online Recei...
college notes - Shortcut
13631-notice (1)
Cd Project
Google Chrome
Priyanshu-...
memories
Brave
CodingNinj...
New folder
session reports
Aarchi Music Player
Untitled-1
VLC media player
CodingNinj...
Chalkpad_ Online Rec...
Git 2.35.1.2 Setup
Installing
Please wait while Setup installs Git on your computer.
Extracting files...
D:\Git\Git\usr\share\perl5\core_perl\unicore\lib\GCB\SM.pl
https://gitforwindows.org/
Cancel

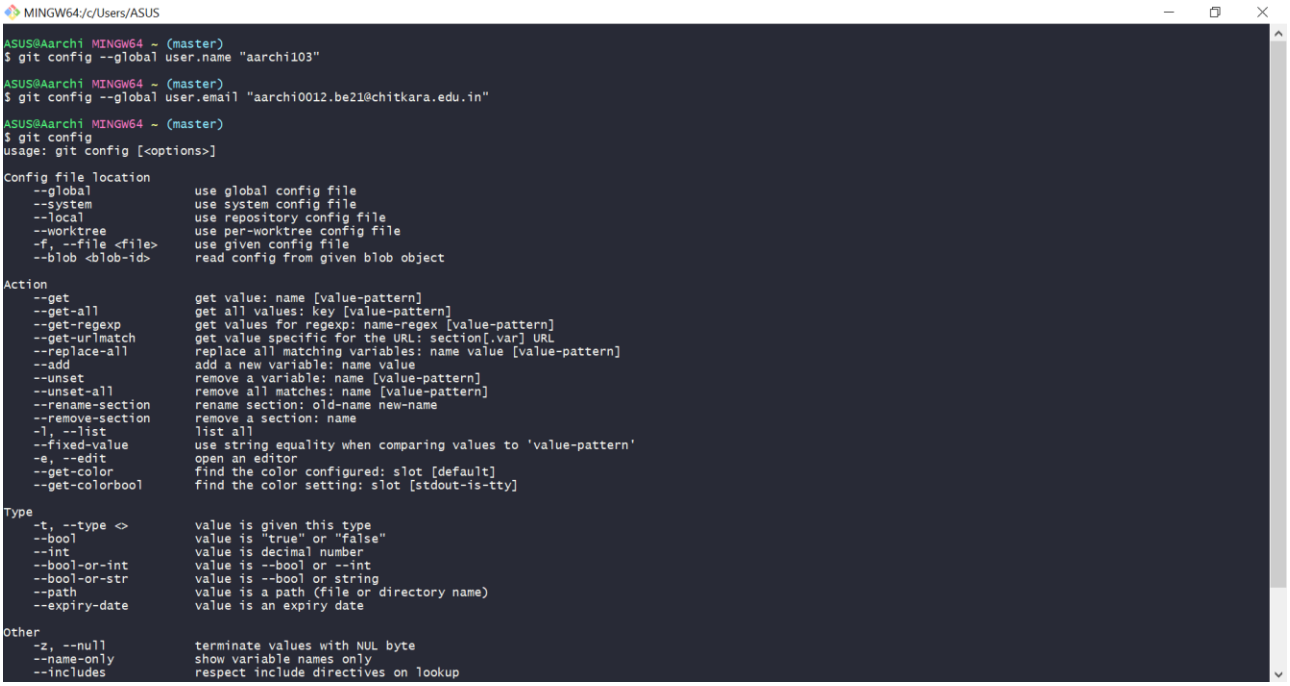

Git
File
Home
Share
View
Pin to Quick access
Copy
Paste
Cut
Copy path
Paste shortcut
Clipboard
Move to
Copy to
Delete
Rename
Organize
New item
Easy access
New folder
New
Properties
Open
Edit
History
Open
Select all
Select none
Invert selection
Select
This PC
Local Disk (D:)
Git
Search Git
Camera Roll
Documents
Pictures
books
course handouts
dj night
Screenshots
OneDrive - Person
This PC
3D Objects
Desktop
Documents
Downloads
Music
Pictures
Videos
OS (C:)
Local Disk (D:)
Network
Name
Date modified
Type
Size
bin 3/31/2022 11:24 AM File folder
cmd 3/31/2022 11:24 AM File folder
dev 3/31/2022 11:24 AM File folder
etc 3/31/2022 11:24 AM File folder
mingw64 3/31/2022 11:24 AM File folder
tmp 3/31/2022 11:24 AM File folder
usr 3/31/2022 11:24 AM File folder
git-bash 2/1/2022 3:47 PM Application 135 KB
git-cmd 2/1/2022 3:47 PM Application 135 KB
LICENSE 2/1/2022 4:12 PM Text Document 19 KB
ReleaseNotes 2/1/2022 4:13 PM Microsoft Edge HTM... 208 KB
unins000.dat 3/31/2022 11:24 AM DAT File 1,085 KB
unins000 3/31/2022 11:21 AM Application 3,039 KB
unins000.msg 3/31/2022 11:24 AM MSG File 24 KB

SETTING UP OF GITHUB ACCOUNT

You can set up a GitHub account by going onto the website https://github.com/ and choosing the 'Sign Up" option. Enter the required details and you are done with creating account on GitHub.



SETTING USER-NAME AND EMAIL



MAKING A DIRECTORY AND VARIOUS COMMANDS

mkdir=> It is used for creating a directory using Git Bash.

cd command=>Both Bash and Windows console host have a cd command. cd is an acronym for 'Change Directory'. cd is invoked with an appended directory name. Executing cd will change the terminal sessions current working directory to the passed directory argument.

pwd=>The Bash command pwd is used to print the 'present working directory'. This is the folder or path that the current Bash session resides in.

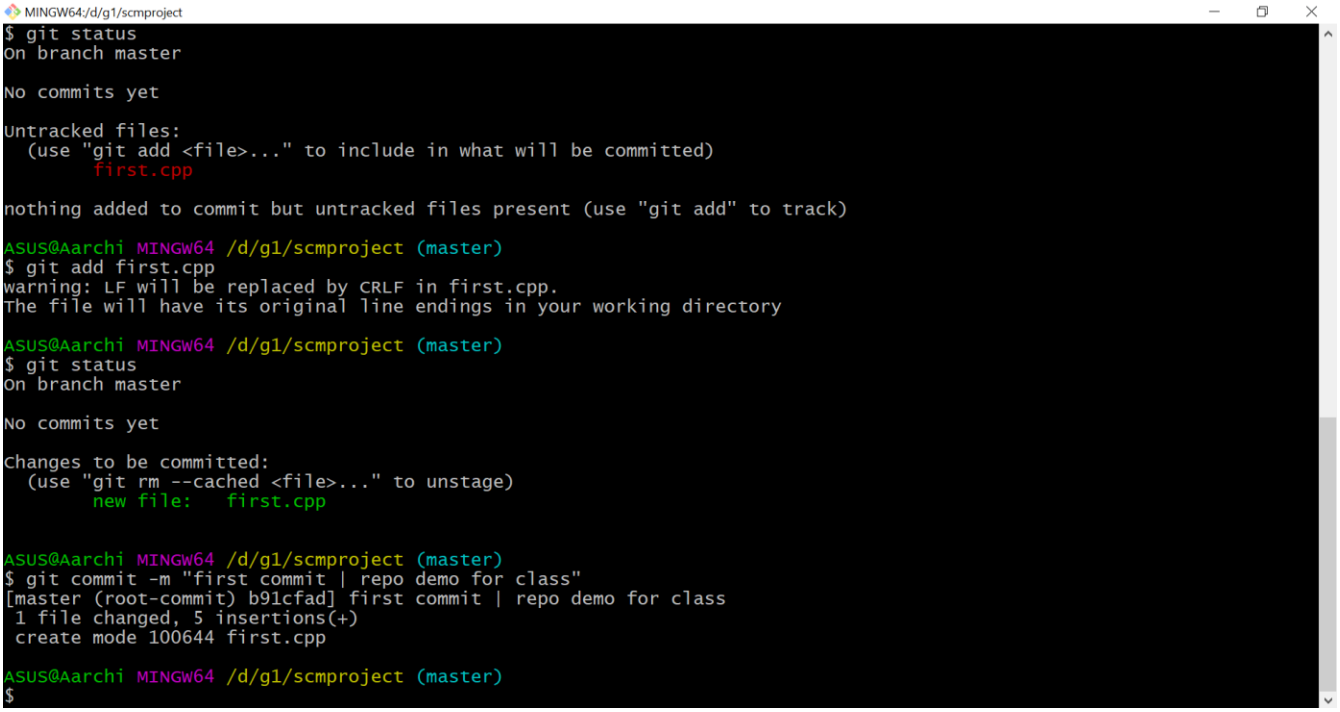
ls=>The Bash command ls is used to 'list' contents of the current working directory.

git status=>The main tool you use to determine which files are in which state is the git status command. the command tells you which branch you're on and informs you that it has not diverged from the same branch on the server.

git add =>In order to begin tracking a new file, you use the command git add. git add is a multipurpose command — you use it to begin tracking new files, to stage files, and to do other things like marking merge-conflicted files as resolved. It may be helpful to think of it more as "add precisely this content to the next commit" rather than "add this file to the project".

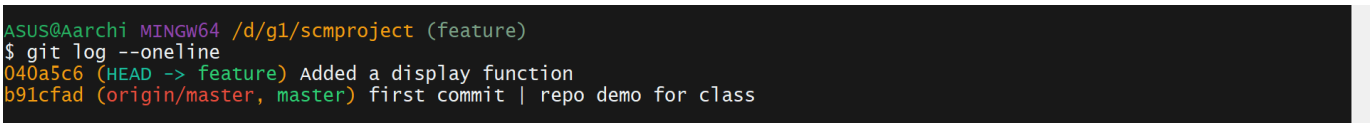
git commit=> The git commit command captures a snapshot of the project's currently staged changes. Committed snapshots can be thought of as "safe" versions of a project—Git will never change them unless you explicitly ask it to. Prior to the execution of git commit, The git add command is used to promote or 'stage' changes to the project that will be stored in a commit. These two commands git commit and git add are two of the most frequently used.

git push=> When you have your project at a point that you want to share, you have to push it upstream. The command for this is simple: git push . If you want to push your master branch to your origin server (again, cloning generally sets up both of those names for you automatically), then you can run this to push any commits you've done back up to the server: $
git push origin master



GENERATING LOGS

Logs are nothing but the history which we can see in Git by using the code Git log. It contains all the past commits, insertions and deletions which can be seen anytime.



GIT 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.

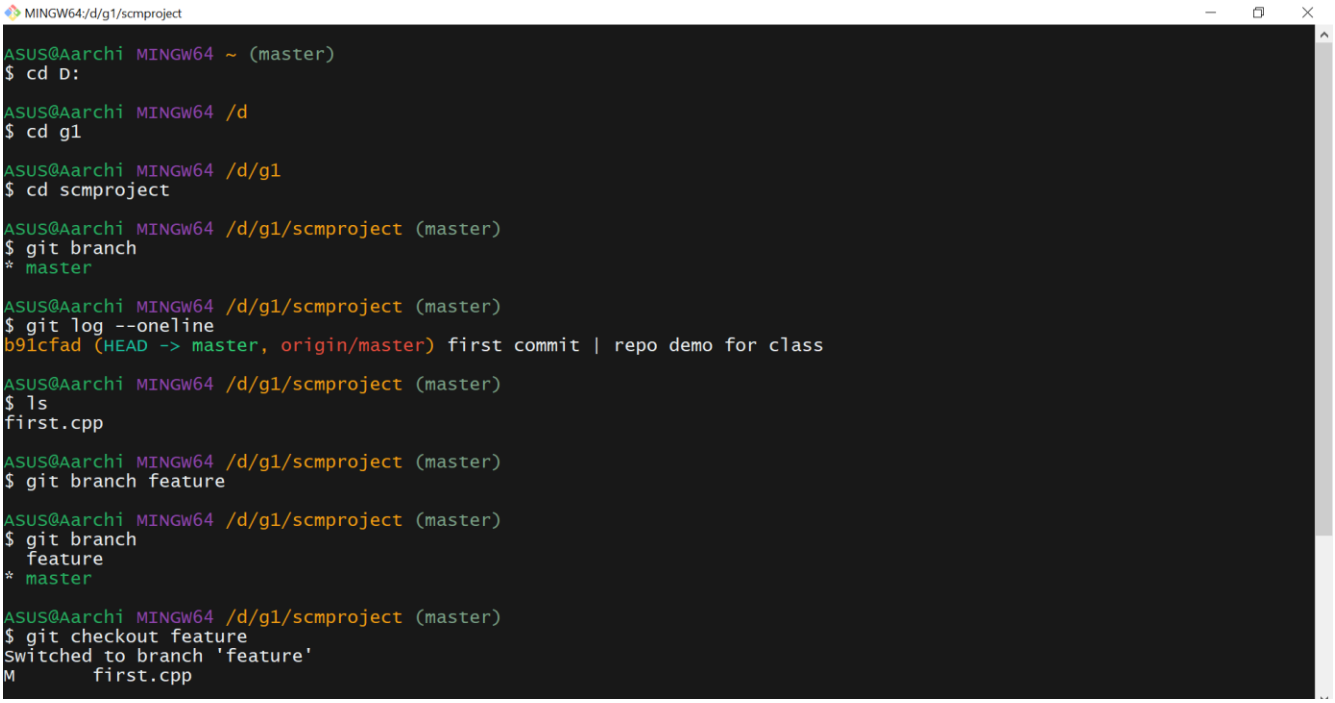
Firstly, add a new branch, let us suppose the branch name is activity1.

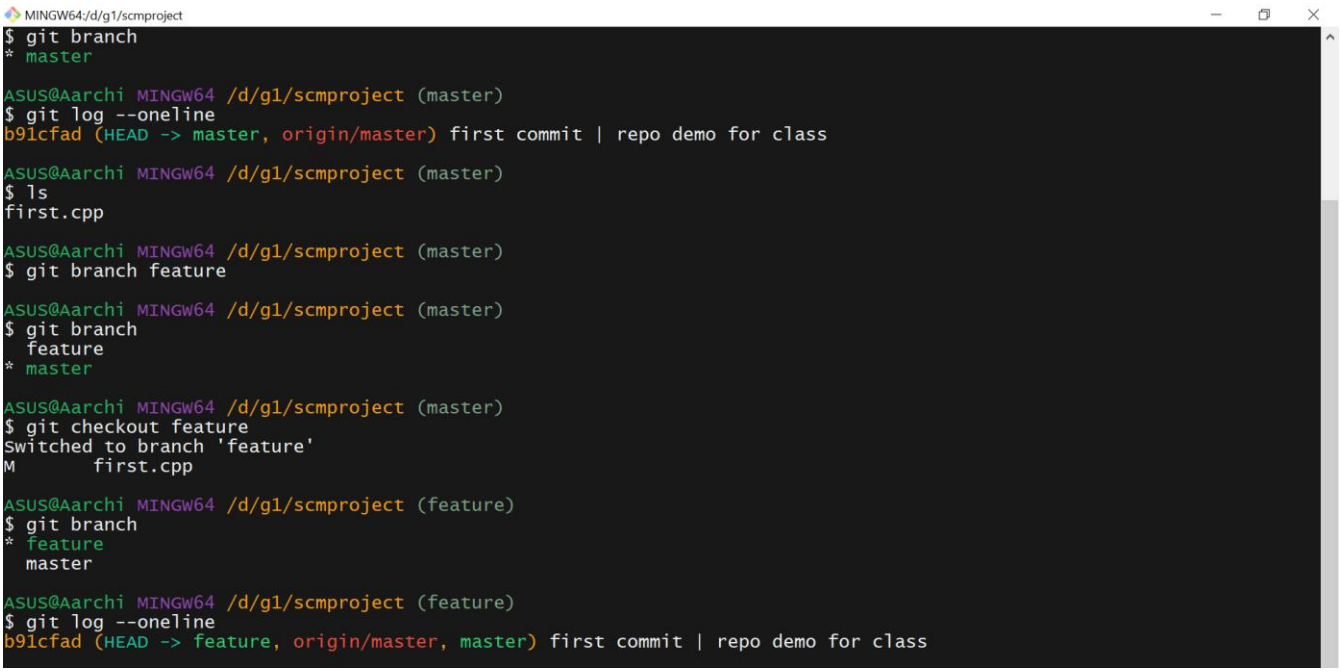
For this use command →

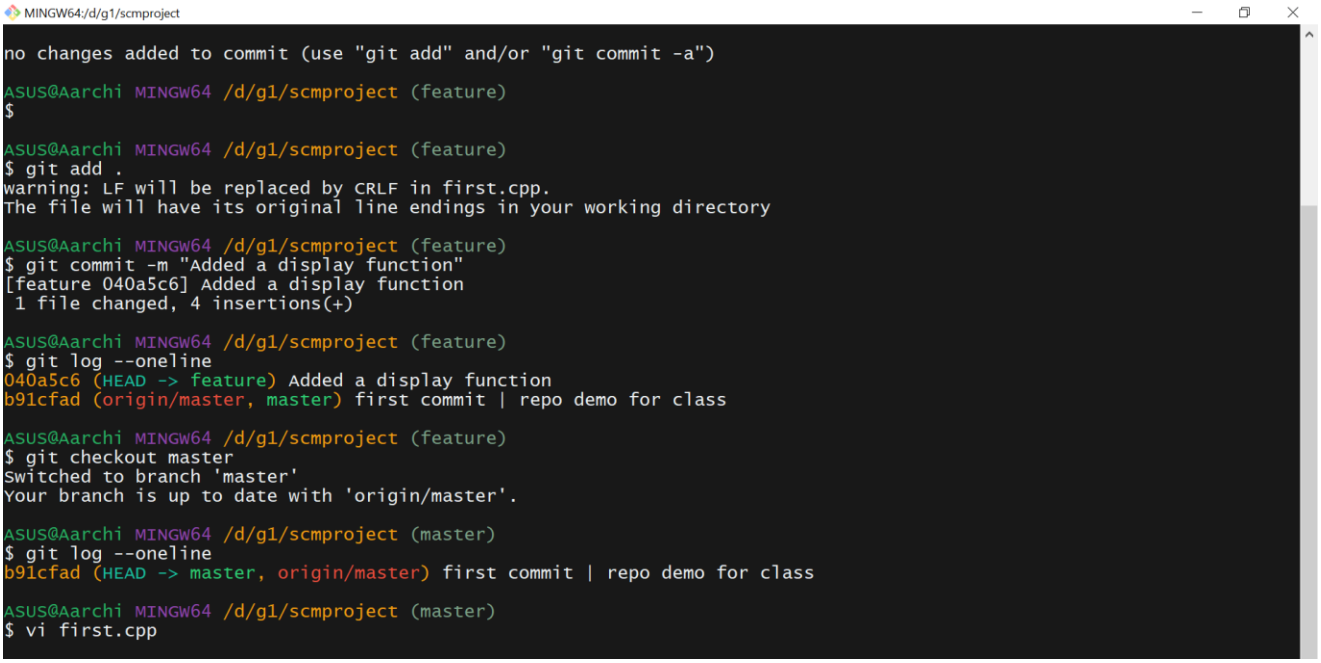
git branch name [adding new branch**]**

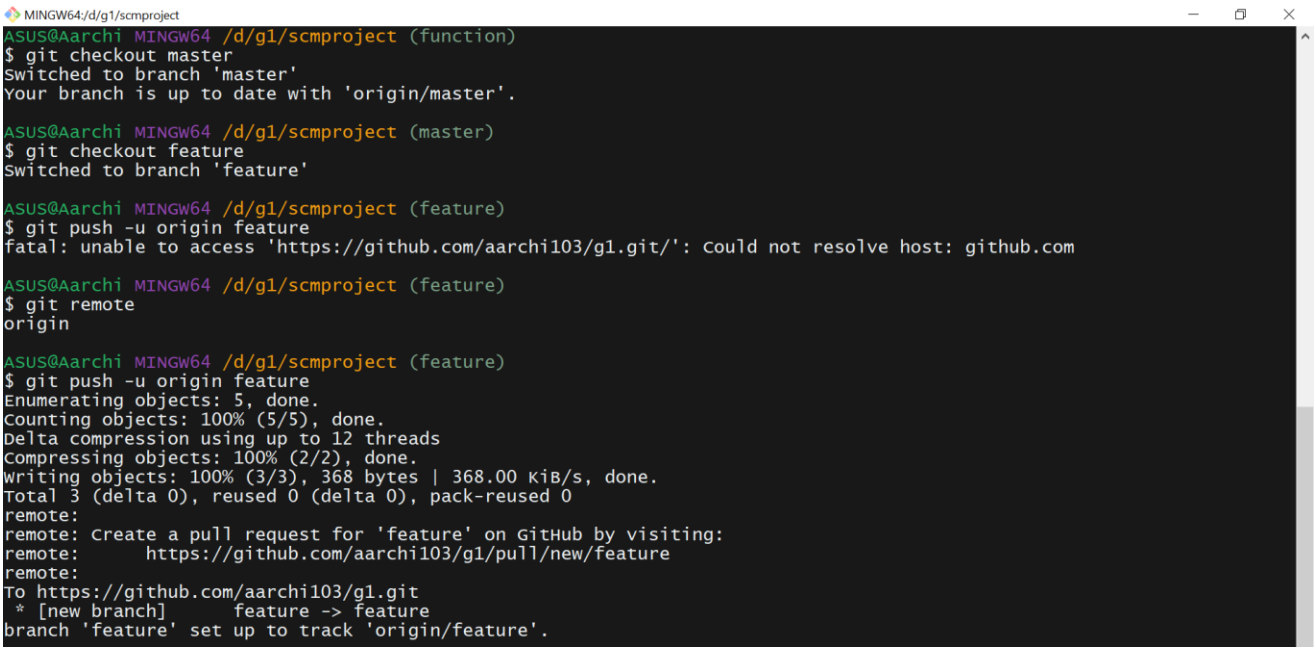
git branch [use to see the branch's names**]**

git checkout *branch name* [use to switch to the given branch]


MINGW64:/d/g1/scmproject
ASUS@Aarchi MINGW64 ~ (master)
$ cd D:
ASUS@Aarchi MINGW64 /d
$ cd g1
ASUS@Aarchi MINGW64 /d/g1
$ cd scmproject
ASUS@Aarchi MINGW64 /d/g1/scmproject (master)
$ git branch
* master
ASUS@Aarchi MINGW64 /d/g1/scmproject (master)
$ git log --oneline
b91cfad (HEAD -> master, origin/master) first commit | repo demo for class
ASUS@Aarchi MINGW64 /d/g1/scmproject (master)
$ ls
first.cpp
ASUS@Aarchi MINGW64 /d/g1/scmproject (master)
$ git branch feature
ASUS@Aarchi MINGW64 /d/g1/scmproject (master)
$ git branch
feature
* master
ASUS@Aarchi MINGW64 /d/g1/scmproject (master)
$ git checkout feature
Switched to branch 'feature'
M first.cpp


MINGW64:/d/g1/scmproject
$ git branch
* master
ASUS@Aarchi MINGW64 /d/g1/scmproject (master)
$ git log --oneline
b91cfad (HEAD -> master, origin/master) first commit | repo demo for class
ASUS@Aarchi MINGW64 /d/g1/scmproject (master)
$ ls
first.cpp
ASUS@Aarchi MINGW64 /d/g1/scmproject (master)
$ git branch feature
ASUS@Aarchi MINGW64 /d/g1/scmproject (master)
$ git branch
feature
* master
ASUS@Aarchi MINGW64 /d/g1/scmproject (master)
$ git checkout feature
Switched to branch 'feature'
M first.cpp
ASUS@Aarchi MINGW64 /d/g1/scmproject (feature)
$ git branch
* feature
master
ASUS@Aarchi MINGW64 /d/g1/scmproject (feature)
$ git log --oneline
b91cfad (HEAD -> feature, origin/master, master) first commit | repo demo for class


MINGW64:/d/g1/scmproject
no changes added to commit (use "git add" and/or "git commit -a")
ASUS@Aarchi MINGW64 /d/g1/scmproject (feature)
$
ASUS@Aarchi MINGW64 /d/g1/scmproject (feature)
$ git add .
warning: LF will be replaced by CRLF in first.cpp.
The file will have its original line endings in your working directory
ASUS@Aarchi MINGW64 /d/g1/scmproject (feature)
$ git commit -m "Added a display function"
[feature 040a5c6] Added a display function
1 file changed, 4 insertions(+)
ASUS@Aarchi MINGW64 /d/g1/scmproject (feature)
$ git log --oneline
040a5c6 (HEAD -> feature) Added a display function
b91cfad (origin/master, master) first commit | repo demo for class
ASUS@Aarchi MINGW64 /d/g1/scmproject (feature)
$ git checkout master
Switched to branch 'master'
Your branch is up to date with 'origin/master'.
ASUS@Aarchi MINGW64 /d/g1/scmproject (master)
$ git log --oneline
b91cfad (HEAD -> master, origin/master) first commit | repo demo for class
ASUS@Aarchi MINGW64 /d/g1/scmproject (master)
$ vi first.cpp


MINGW64:/d/g1/scmproject
ASUS@Aarchi MINGW64 /d/g1/scmproject (function)
$ git checkout master
Switched to branch 'master'
Your branch is up to date with 'origin/master'.
ASUS@Aarchi MINGW64 /d/g1/scmproject (master)
$ git checkout feature
Switched to branch 'feature'
ASUS@Aarchi MINGW64 /d/g1/scmproject (feature)
$ git push -u origin feature
fatal: unable to access 'https://github.com/aarchi103/g1.git/': Could not resolve host: github.com
ASUS@Aarchi MINGW64 /d/g1/scmproject (feature)
$ git remote
origin
ASUS@Aarchi MINGW64 /d/g1/scmproject (feature)
$ git push -u origin feature
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 12 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 368 bytes | 368.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'feature' on GitHub by visiting:
remote: https://github.com/aarchi103/g1/pull/new/feature
remote:
To https://github.com/aarchi103/g1.git
* [new branch] feature -> feature
branch 'feature' set up to track 'origin/feature'.

GIT LIFECYCLE DESCRIPTION

It is important for us to have an abstract idea of the different stages of Git before going into more detailed understanding of Git.
Files in a Git project have various stages like Creation, Modification, Refactoring, and Deletion and so on.
The three Git states:

- Working directory
- Staging area
- Git directory

Working Directory:

Consider a project residing in your local system. This project may or may not be tracked by Git. In either case, this project directory is called your Working directory.

Staging Area:

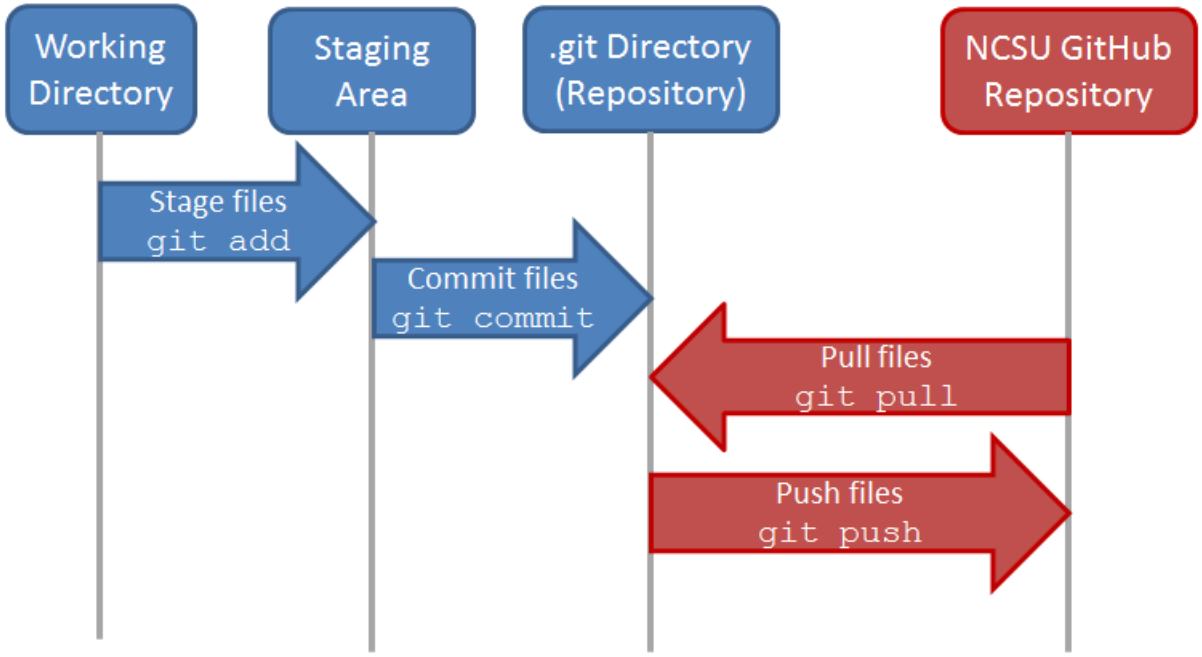
Staging area is the playground where you group, add and organize the files to be committed to Git for tracking their versions.

Git Directory:

Now that the files to be committed are grouped and ready in the staging area, we can commit these files. So, we commit this group of files along with a commit message explaining what is the commit about. Apart from commit message, this step also records the author and time of the commit. Now, a snapshot of the files in the commit is recorded by Git. The information related to this commit is stored in the Git directory.

Remote Repository:

Means mirror or clone of the local Git repository in GitHub. And pushing means uploading the commits from local Git repository to remote repository hosted in GitHub.



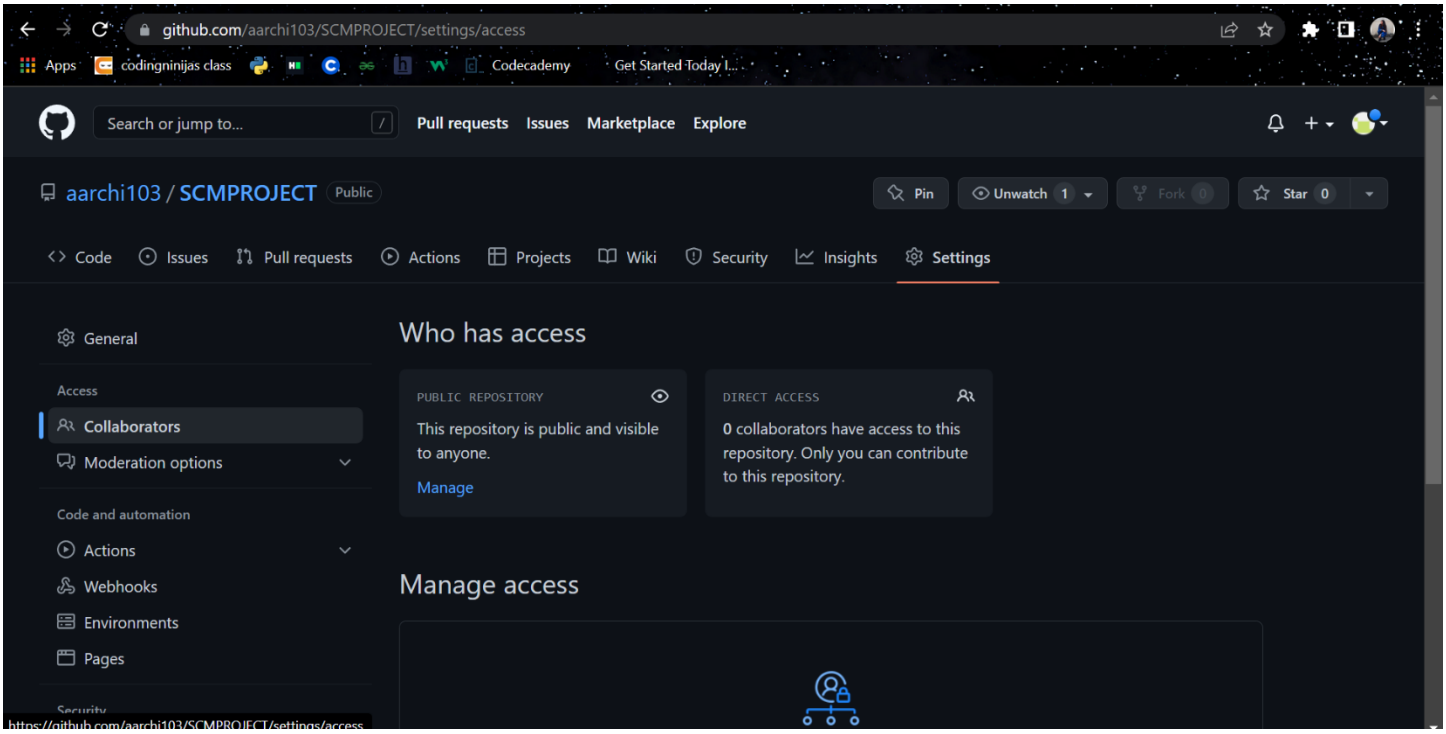
ADD COLLABORATORS ON GITHUB REPOSITORY.

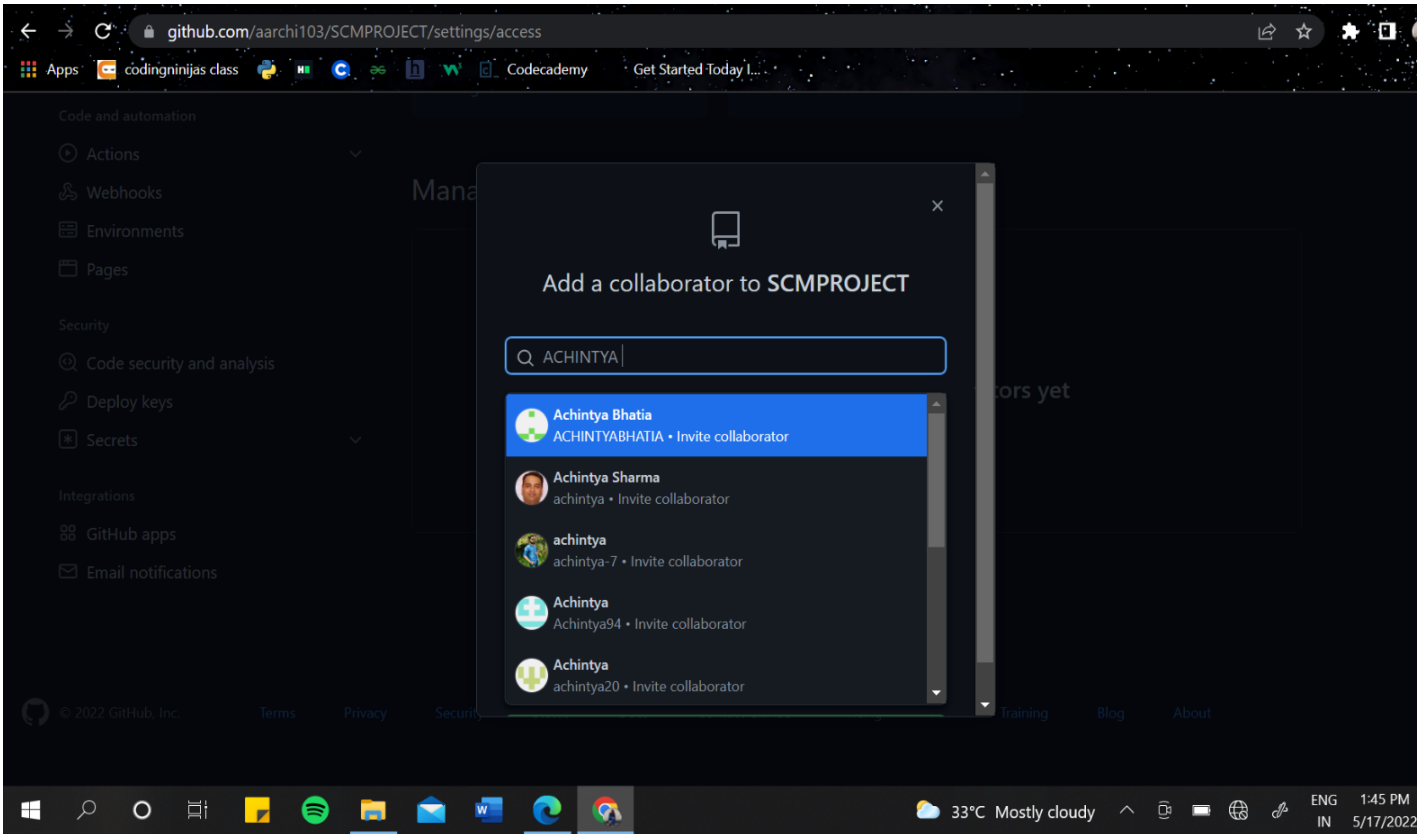
In GitHub, we can invite other GitHub users to become collaborators to our private repositories (which expires after 7 days if not accepted, restoring any unclaimed licenses). Being a collaborator, of a personal repository you can pull (read) the contents of the repository and push (write) changes to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


github.com/aarchi103/SCMPROJECT/settings/access
Search or jump to...
Pull requests
Issues
Marketplace
Explore
aarchi103 / SCMPROJECT
Public
Pin
Unwatch 1
Fork 0
Star 0
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
General
Access
Collaborators
Moderation options
Code and automation
Actions
Webhooks
Environments
Pages
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

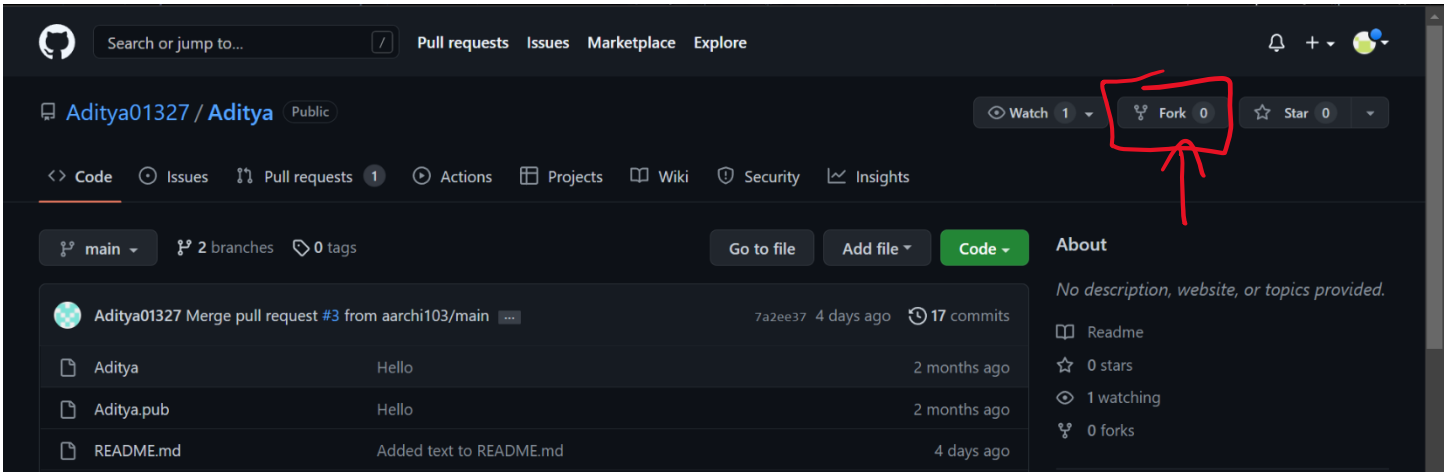

github.com/aarchi103/SCMPROJECT/settings/access
Add a collaborator to SCMPROJECT
ACHINTYA
Achintya Bhatia
ACHINTYABHATIA • Invite collaborator
Achintya Sharma
achintya • Invite collaborator
achintya
achintya-7 • Invite collaborator
Achintya
Achintya94 • Invite collaborator
Achintya
achintya20 • Invite collaborator
33°C Mostly cloudy
ENG
IN
1:45 PM
5/17/2022

FORK AND COMMIT

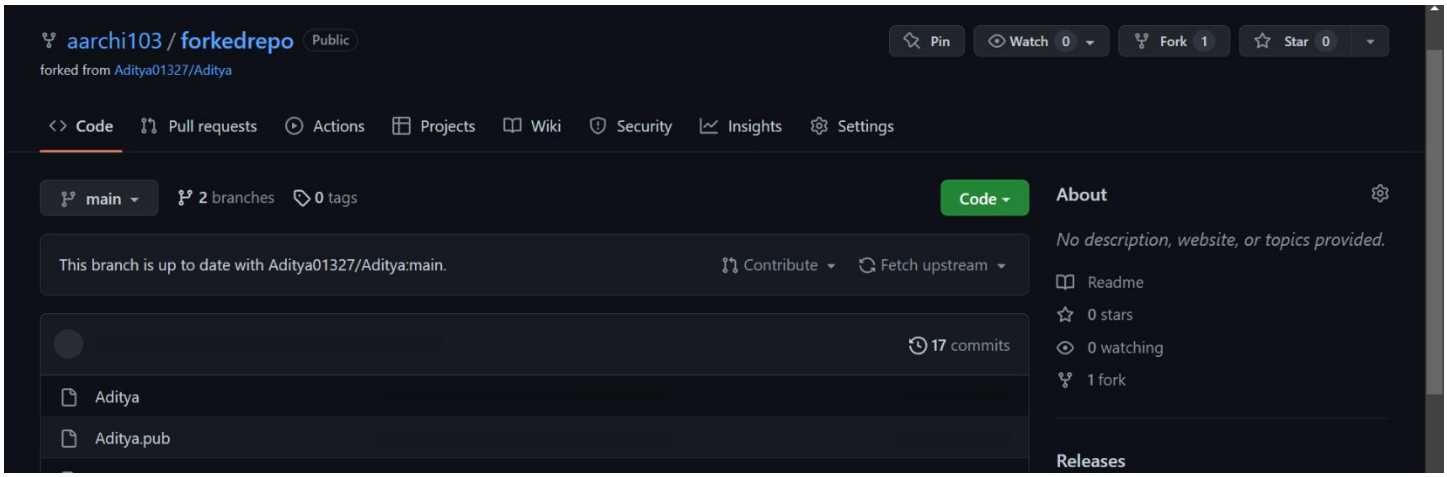
A fork is a copy of a repository. Forking a repository allows you to freely experiment with changes without affecting the original project. Most commonly, forks are used to either propose changes to someone else's project to which you do not have write access, or to use someone else's project as a starting 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. Name your forked repository and click on 'Create New Fork'.
4. 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.

```
ASUS@Aarchi MINGW64 /d/forkedrepo (main)
$ git checkout -b newb
Switched to a new branch 'newb'

ASUS@Aarchi MINGW64 /d/forkedrepo (newb)
$ ls
Aditya  Aditya.pub  README.md  Script.js  aditya.htm  pull.txt

ASUS@Aarchi MINGW64 /d/forkedrepo (newb)
$ vi script.js

ASUS@Aarchi MINGW64 /d/forkedrepo (newb)
$ git status
On branch newb
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:   Script.js

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

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

```
ASUS@Aarchi MINGW64 /d/forkedrepo (newb)
$ git push -u origin newb
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 12 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 356 bytes | 356.00 KiB/s, done.
Total 3 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
remote:
remote: Create a pull request for 'newb' on GitHub by visiting:
remote:      https://github.com/aarchi103/forkedrepo/pull/new/newb
remote:
To https://github.com/aarchi103/forkedrepo.git
 * [new branch]      newb -> newb
branch 'newb' set up to track 'origin/newb'.
```

```
ASUS@Aarchi MINGW64 /d/forkedrepo (newb)
$ git add Script.js

ASUS@Aarchi MINGW64 /d/forkedrepo (newb)
$ git commit -m "added a display function"
[newb ea2a984] added a display function
 1 file changed, 6 insertions(+), 1 deletion(-)
```

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 ‘first.cpp’. Make changes to it, add and commit them.

```
ASUS@Aarchi MINGW64 /d/g1/scmproject (master)
$ git checkout feature
Switched to branch 'feature'
Your branch is up to date with 'origin/feature'.

ASUS@Aarchi MINGW64 /d/g1/scmproject (feature)
$ git log --oneline
040a5c6 (HEAD -> feature, tag: t1, origin/feature) Added a display function
b91cfad (tag: t2, tag: b91cfad, origin/master, master) first commit | repo demo for class
```

```
ASUS@Aarchi MINGW64 /d/g1/scmproject (feature)
$ git checkout master
Switched to branch 'master'
Your branch is ahead of 'origin/master' by 1 commit.
  (use "git push" to publish your local commits)

ASUS@Aarchi MINGW64 /d/g1/scmproject (master)
$ vi first.cpp

ASUS@Aarchi MINGW64 /d/g1/scmproject (master)
$ git add .

ASUS@Aarchi MINGW64 /d/g1/scmproject (master)
$ git commit -m "added in display function"
[master ea6373f] added in display function
 1 file changed, 1 insertion(+)
```

3. Similarly, change the same lines of first.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.

```
ASUS@Aarchi MINGW64 /d/g1/scmproject (master)
$ git merge feature
Auto-merging first.cpp
CONFLICT (content): Merge conflict in first.cpp
Automatic merge failed; fix conflicts and then commit the result.

ASUS@Aarchi MINGW64 /d/g1/scmproject (master|MERGING)
$
```

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/g1/scmproject

```
#include <iostream>
using namespace std;
int main(){
cout<<"hello world"<<endl;
int a=10, b=20,c;
c=a+b;
}
void display(){
        cout<<c;
        cout<<"changing in master bra
```

<76.cpp [dos] (14:07 11/05/2022)1,1 All

```
#include <iostream>
using namespace std;
int main(){
cout<<"hello world"<<endl;
int a=10, b=20,c;
c=a+b;
}
void display(){
        cout<<c;
```

<76.cpp [dos] (14:07 11/05/2022)1,1 All

```
#include <iostream>
using namespace std;
int main(){
cout<<"hello world"<<endl;
int a=10, b=20,c;
c=a+b;
}
void display(){
        cout<<c;
        cout<<"HEllo this is soemthi
```

<cpp [dos] (14:07 11/05/2022)1,0-1 All

```
#include <iostream>
using namespace std;
int main(){
cout<<"hello world"<<endl;
int a=10, b=20,c;
c=a+b;
}
void display(){
        cout<<c;
<<<<<<< HEAD
        cout<<"changing in master branch"
=======
        cout<<"HEllo this is soemthing new added"
>>>>>>> feature
```

first.cpp [dos] (14:07 11/05/2022) 10,9-16 All

"first.cpp" [dos] 15L, 264B

MINGW64:/d/g1/scmproject

```
On branch master
Your branch is ahead of 'origin/master' by 2 commits.
  (use "git push" to publish your local commits)

All conflicts fixed but you are still merging.
  (use "git commit" to conclude merge)

Changes to be committed:
        modified:   first.cpp

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


ASUS@Aarchi MINGW64 /d/g1/scmproject (master|MERGING)
$ git add .

ASUS@Aarchi MINGW64 /d/g1/scmproject (master|MERGING)
$ git commit -m "merging branches"
[master eaaf07c] merging branches

ASUS@Aarchi MINGW64 /d/g1/scmproject (master)
$ git log -nleine

ASUS@Aarchi MINGW64 /d/g1/scmproject (master)
$ git log --oneline
eaaf07c (HEAD -> master) merging branches
ea6373f added in display function
66464fc (feature) changed display function
040a5c6 (tag: t1, origin/feature) Added a display function
b91cfad (tag: t2, tag: b91cfad, origin/master) first commit | repo demo for class
```

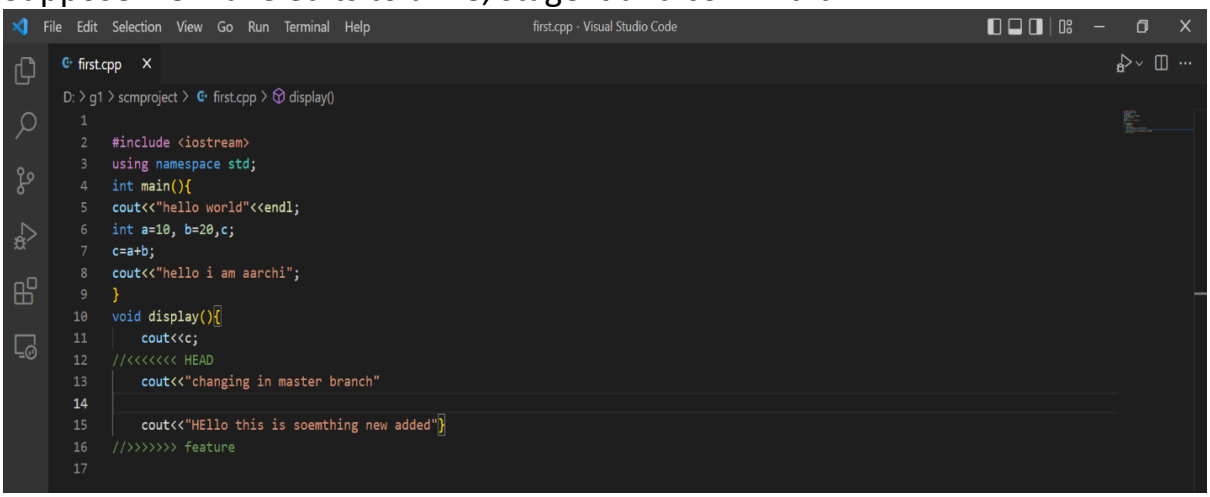
RESET AND REVERT

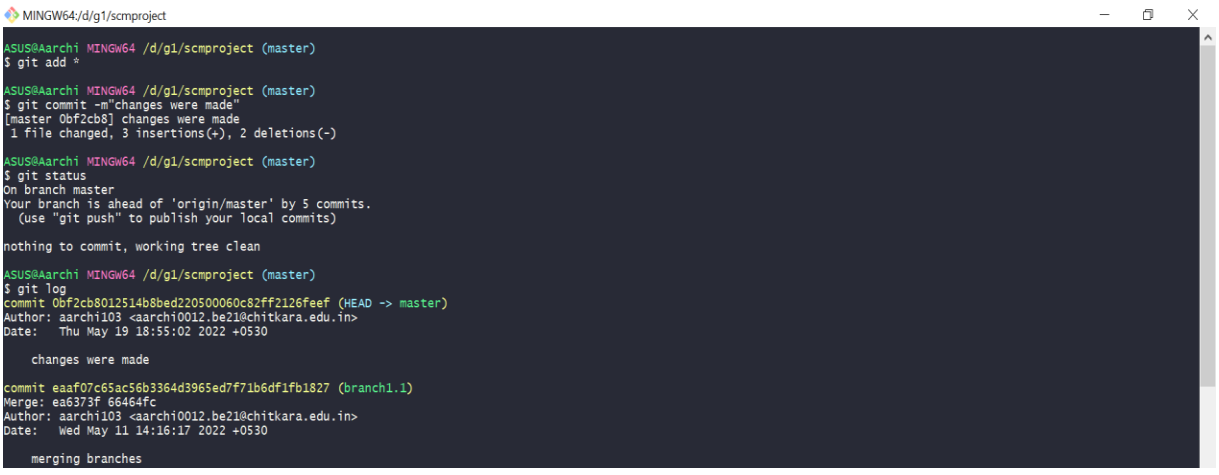
While Working with Git in certain situations we want to undo changes in the working area or index area, sometimes remove commits locally or remotely and we need to reverse those changes. We can do it by using the git reset, git revert, git checkout commands.

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.

```
ASUS@Aarchi MINGW64 /d/g1/scmproject (master)
$ git reset --hard HEAD^1
HEAD is now at eaaf07c merging branches

ASUS@Aarchi MINGW64 /d/g1/scmproject (master)
$ git log
commit eaaf07c65ac56b3364d3965ed7f71b6df1fb1827 (HEAD -> master, branch1.1)
Merge: ea6373f 66464fc
Author: aarchi103 <aarchi0012.be21@chitkara.edu.in>
Date:   Wed May 11 14:16:17 2022 +0530

    merging branches

commit ea6373ffd30a15adb1753fb34dfe06355475d51a
Author: aarchi103 <aarchi0012.be21@chitkara.edu.in>
Date:   Wed May 11 14:04:55 2022 +0530

    added in display function
```

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

first.cpp

D: > g1 > scmproject > first.cpp > ...

```
1
2   #include <iostream>
3   using namespace std;
4   int main(){
5   cout<<"hello world"<<endl;
6   int a=10, b=20,c;
7   c=a+b;
8   }
9   void display(){
10      cout<<c;
11  //<<<<<<< HEAD
12      cout<<"changing in master branch"
13  =======
14      cout<<"HEllo this is soemthing new added"
15  //>>>>>>> feature
16
```

REVERT-

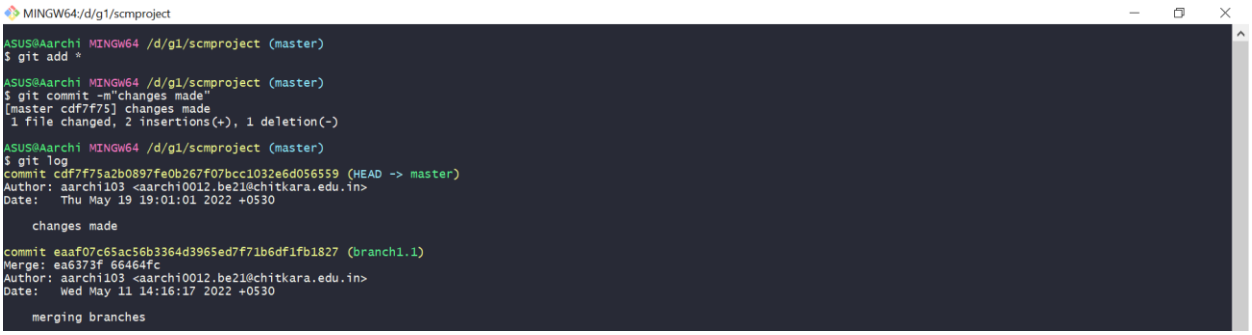
git revert is used to remove the commits from the remote repository. git revert removes the commit that we have done but adds one more commit which tells us that the revert has been done.

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

first.cpp M

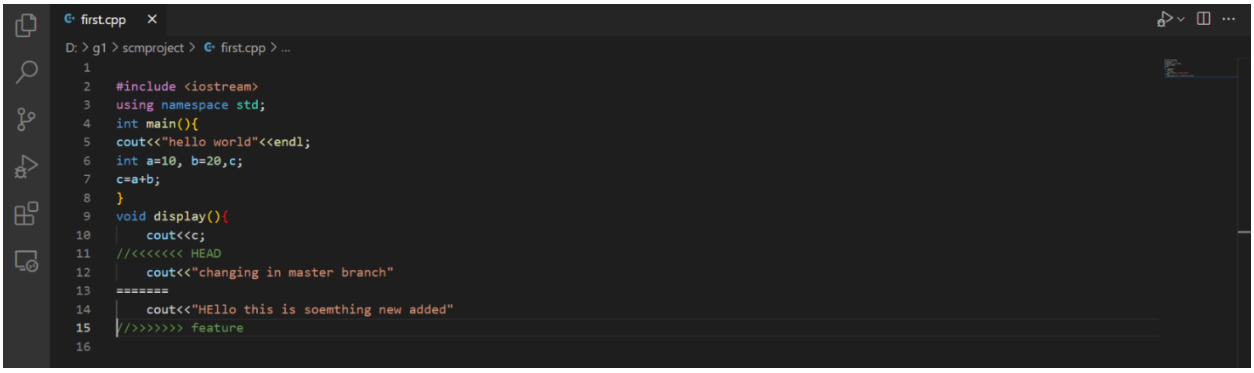
D: > g1 > scmproject > first.cpp > display()

```
1
2   #include <iostream>
3   using namespace std;
4   int main(){
5   cout<<"hello world"<<endl;
6   int a=10, b=20,c;
7   c=a+b;
8   cout<<"scm project";
9   }
10  void display(){
11      cout<<c;
12  //<<<<<<< HEAD
13      cout<<"changing in master branch"
14  =======
15      cout<<"HEllo this is soemthing new added"}
16  //>>>>>>> feature
17
```



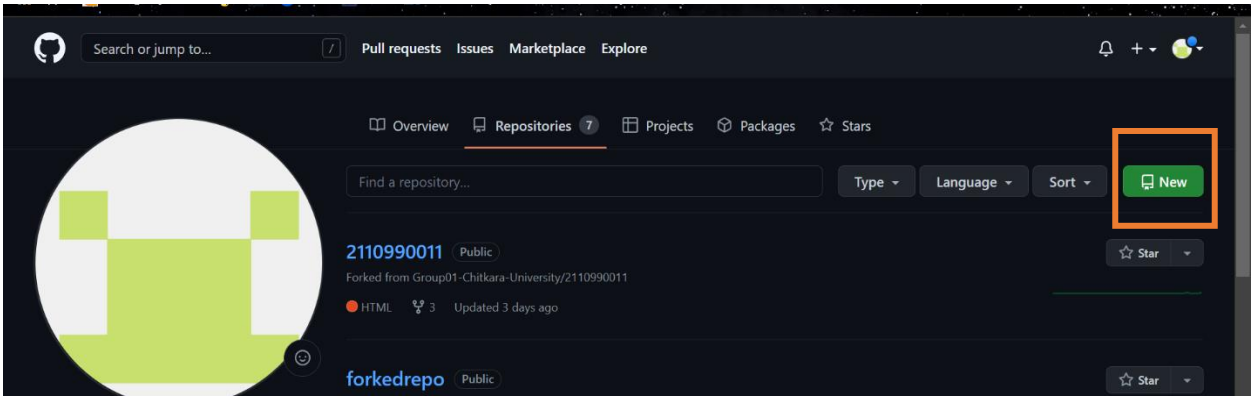
Now to revert the changes made in the commit run the “git revert <commit id>” command.

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

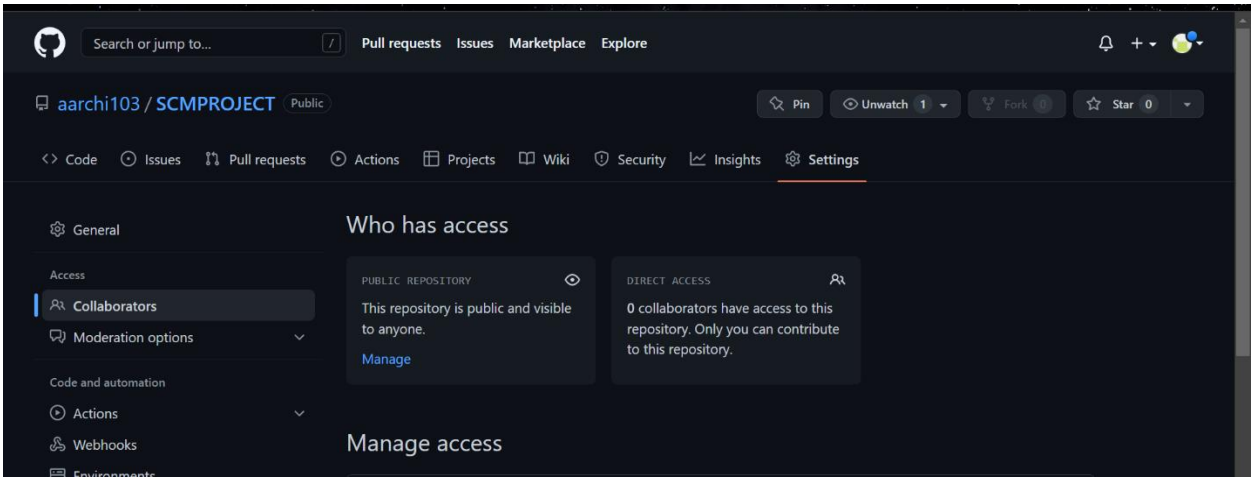


CREATE A DISTRIBUTED REPOSITORY AND ADD MEMBERS IN PROJECT TEAM

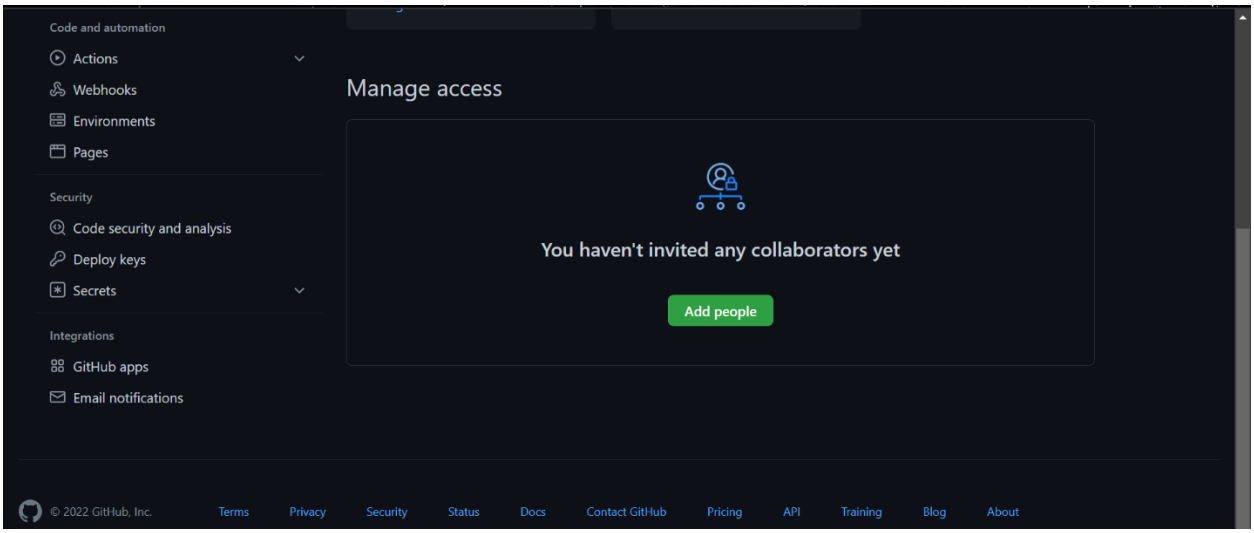
1. On the homepage of your GitHub account, click on Repositories option in the menu bar.
2. Click on the 'New' button in the top right corner.

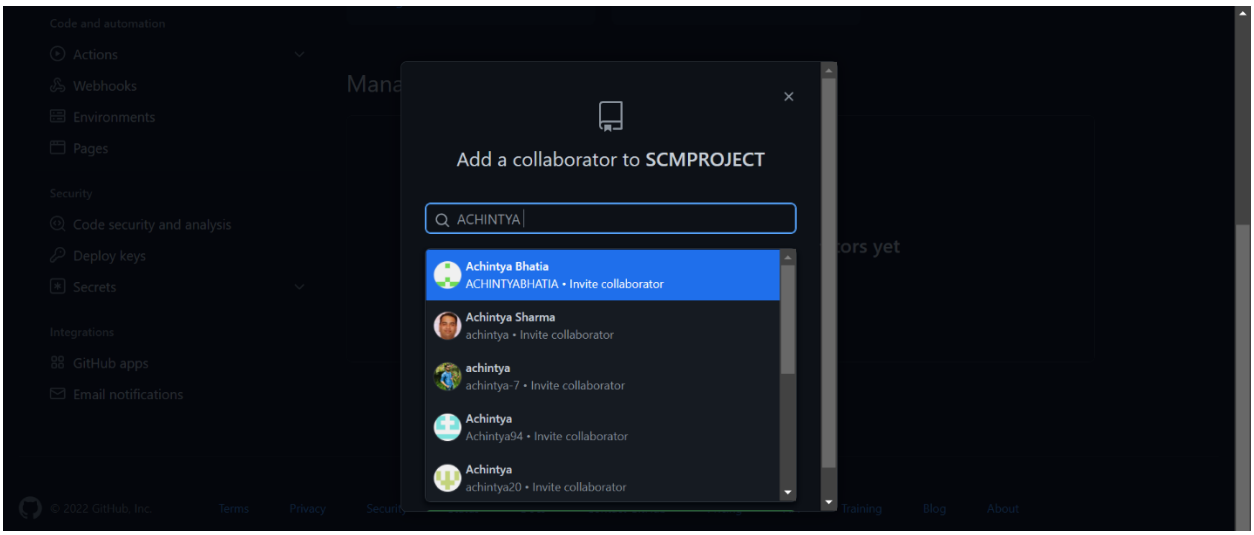


3. Enter the Repository name and add the description of the repository.
4. To add members to your repository, open your repository and select settings option in the navigation bar.
5. Click on Collaborators option under the access tab.

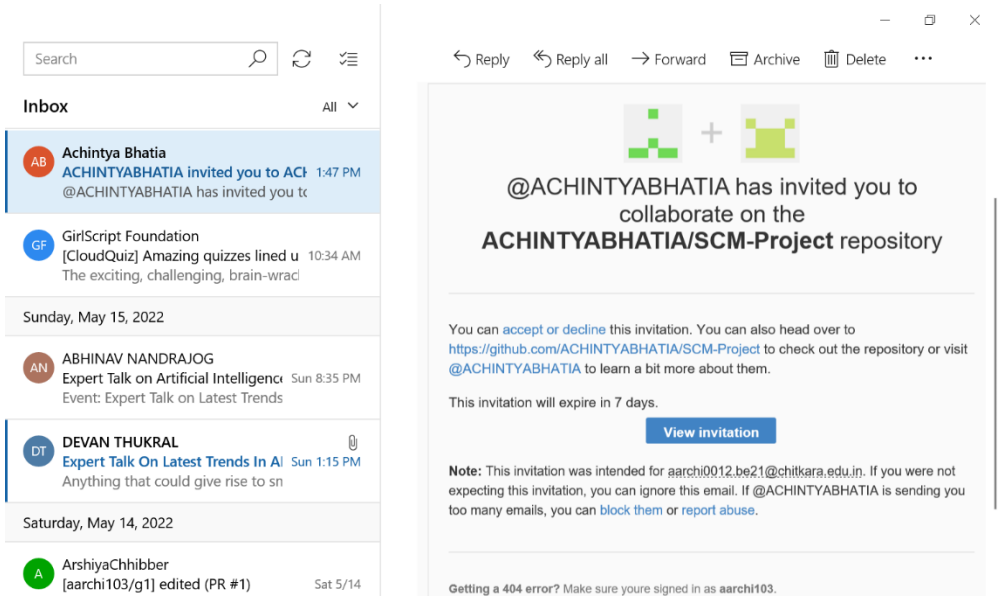


6. You can manage access and add/remove team members to your project.
7. To add members, click on the add people option and search the id of your respective team member.

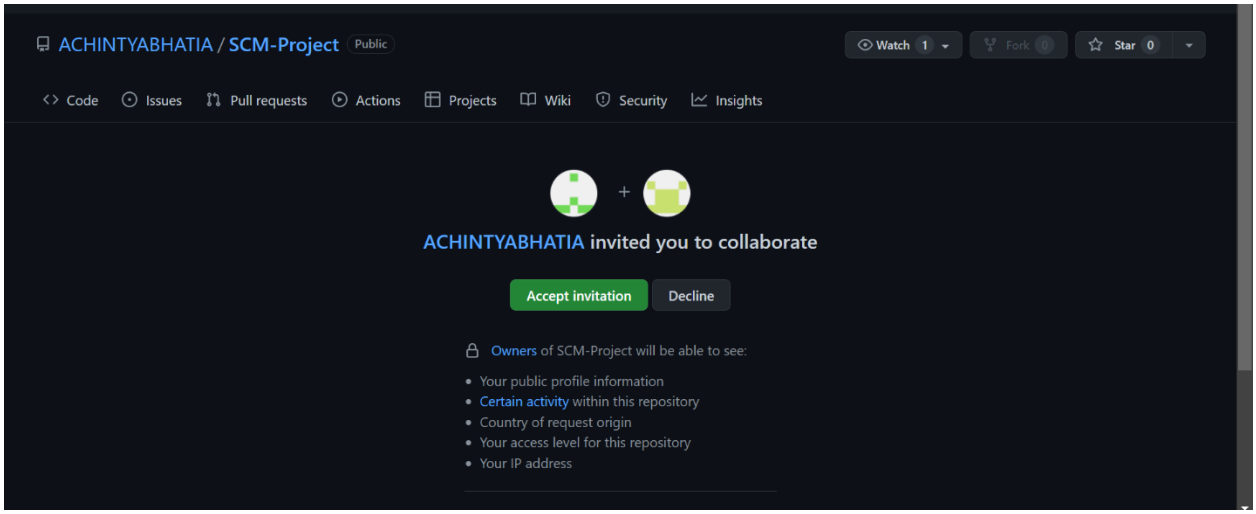




8. To accept the invitation from your team member, open your email registered with GitHub.
9. You will receive an invitation mail from the repository owner. Open the email and click on accept invitation.



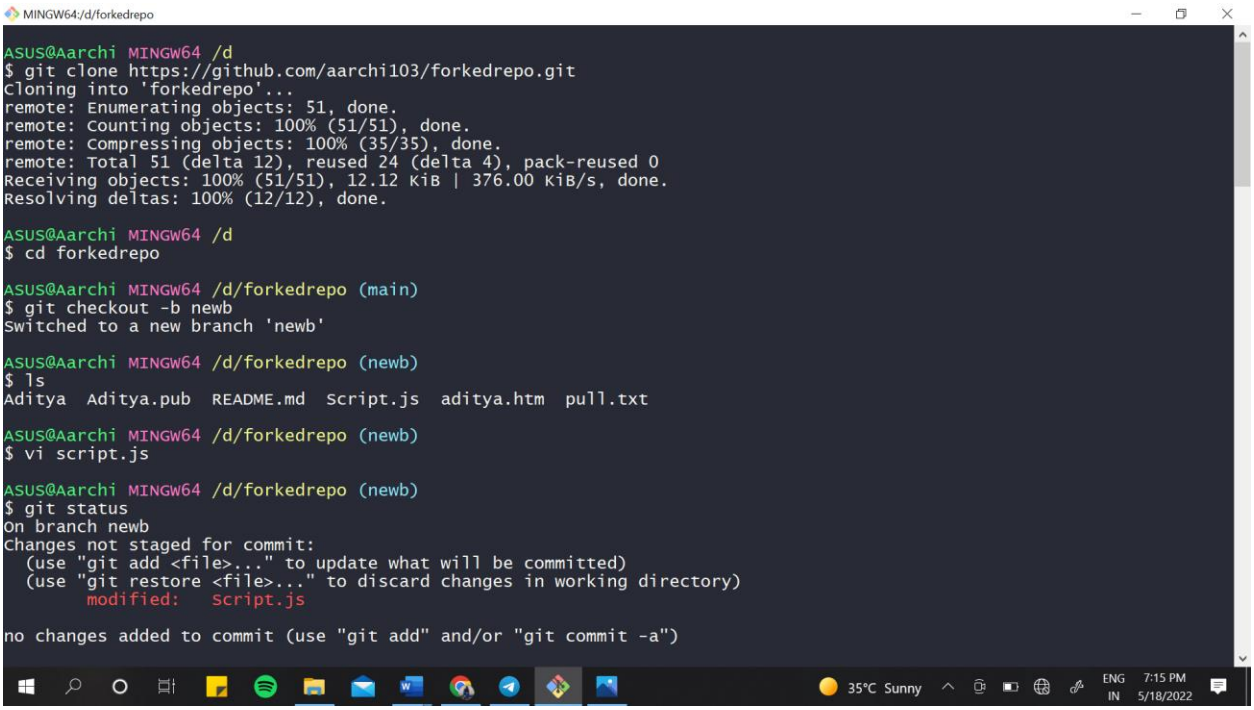
10. You will be redirected to GitHub where you can either select to accept or decline the invitation.



Similarly, you can add more collaborators to your project.

OPEN AND CLOSE A PULL REQUEST

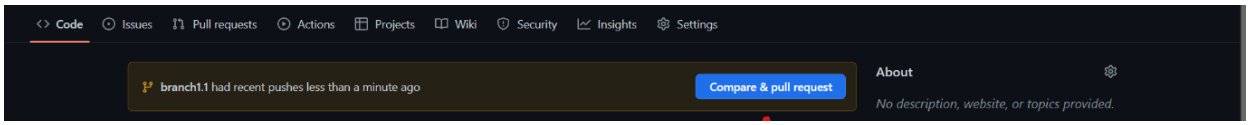
1. First, select a repository of the other person in which you want to make changes and create a pull request.
2. Clone it into your local storage.
3. To open a pull request we first have to make a new branch, by using git checkout -b *branch name* option.
4. After making new branch we add a file to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



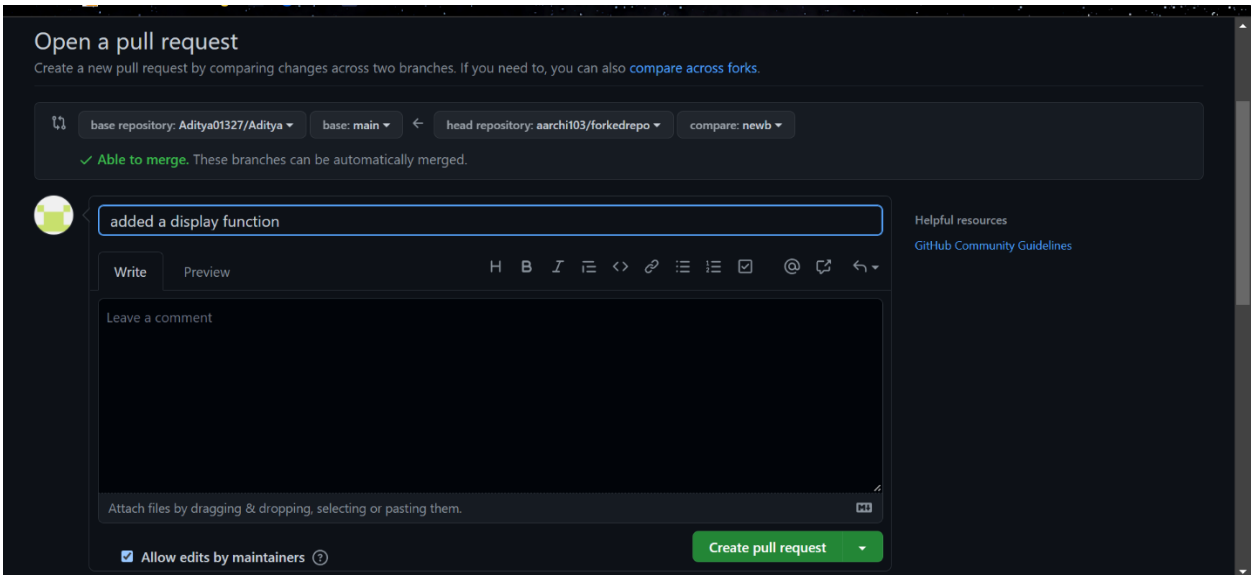
6. Use git push origin *branch name* option to push the new branch to the main repository.

```
ASUS@Aarchi MINGW64 /d/forkedrepo (newb)
$ git push -u origin newb
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 12 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 356 bytes | 356.00 KiB/s, done.
Total 3 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
remote:
remote: Create a pull request for 'newb' on GitHub by visiting:
remote:      https://github.com/aarchi103/forkedrepo/pull/new/newb
remote:
To https://github.com/aarchi103/forkedrepo.git
 * [new branch]      newb -> newb
branch 'newb' set up to track 'origin/newb'.

ASUS@Aarchi MINGW64 /d/forkedrepo (newb)
$
```

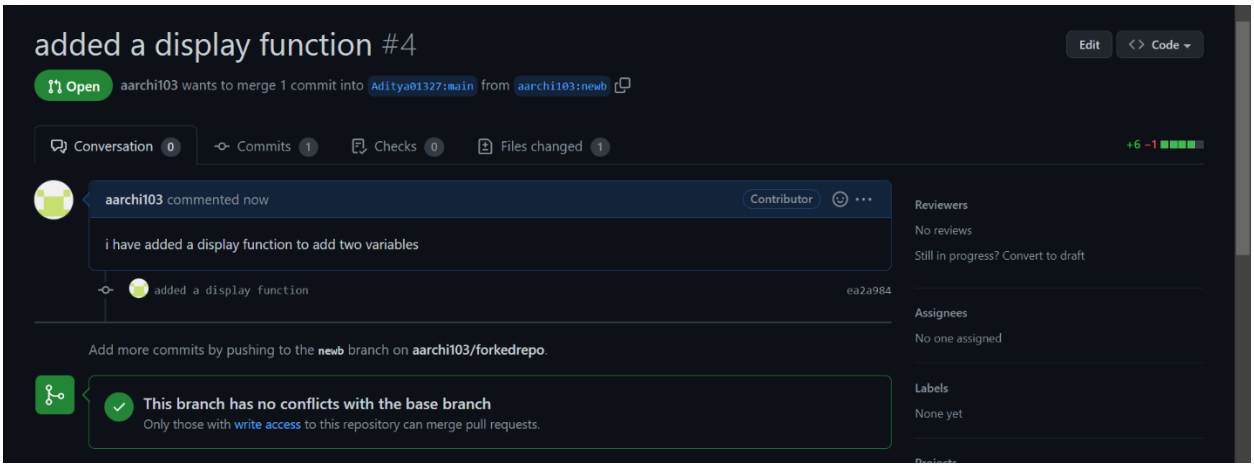
7. After pushing new branch GitHub will either automatically ask you to create a pull request or you can create your own pull request.



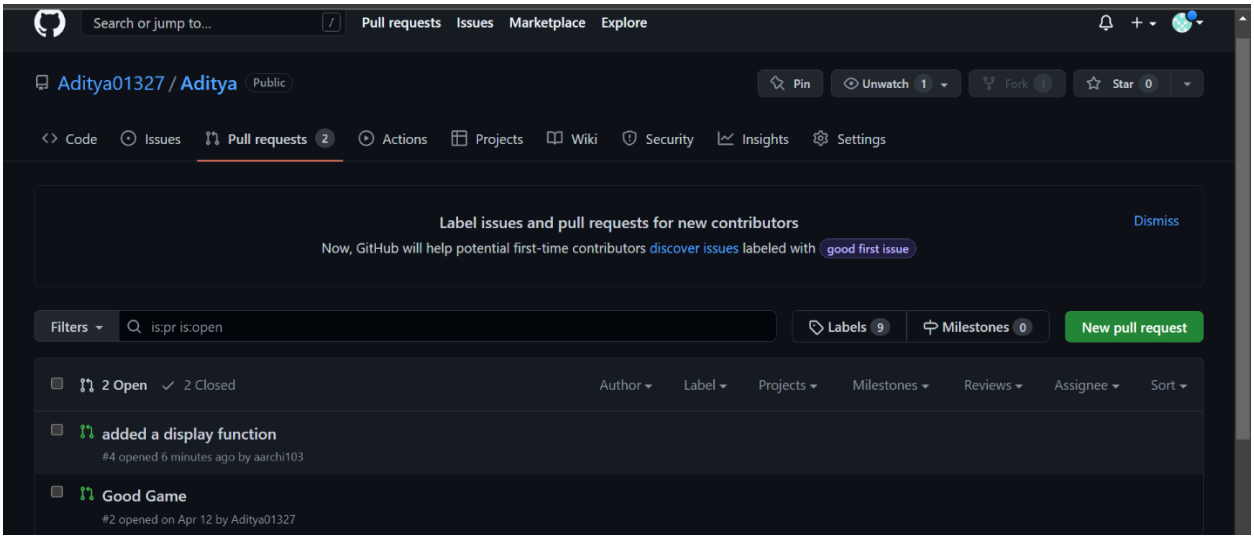
8. To create your own pull request, click on pull request option.



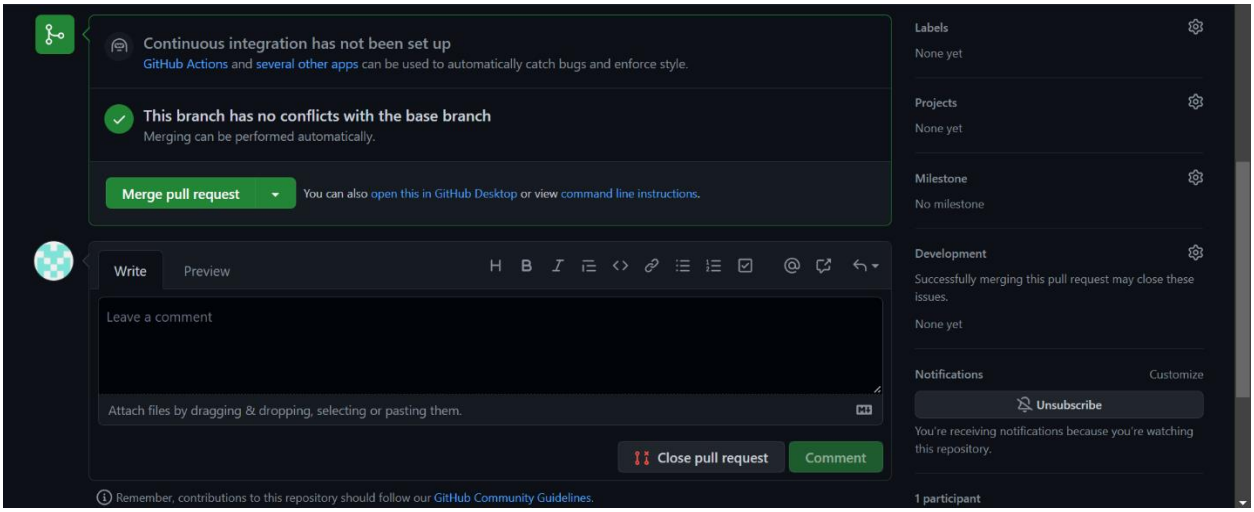
9. GitHub will detect any conflicts and ask you to enter a description of your pull request.

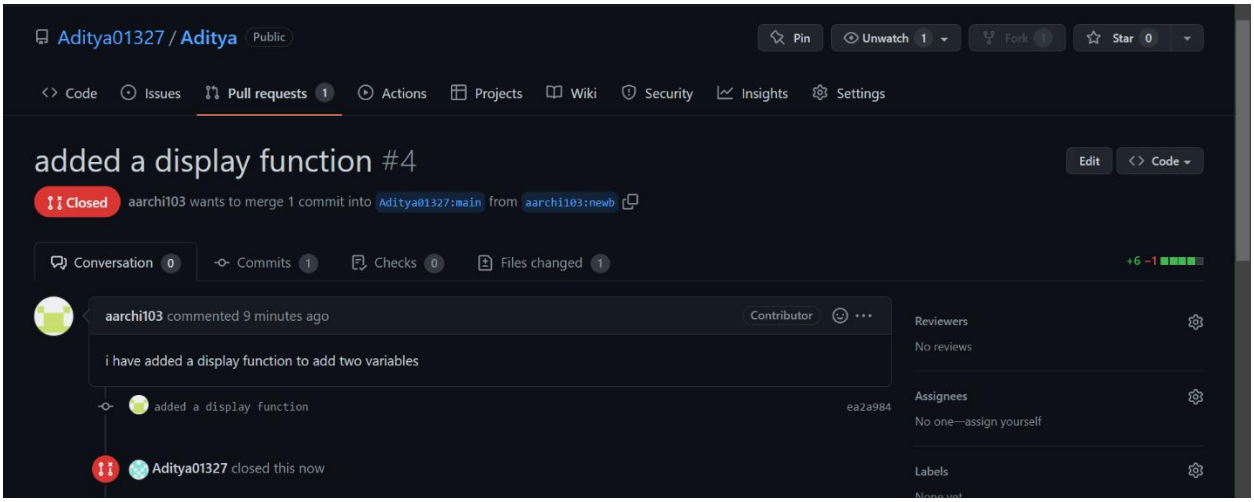


10. After opening a pull request the owner of the original repository will be sent the request if they want to merge or close the request.



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.


Continuous integration has not been set up
GitHub Actions and several other apps can be used to automatically catch bugs and enforce style.
This branch has no conflicts with the base branch
Merging can be performed automatically.
Merge pull request
You can also open this in GitHub Desktop or view command line instructions.
Write
Preview
Leave a comment
Attach files by dragging & dropping, selecting or pasting them.
Close pull request
Comment
Remember, contributions to this repository should follow our GitHub Community Guidelines.
Labels
None yet
Projects
None yet
Milestone
No milestone
Development
Successfully merging this pull request may close these issues.
None yet
Notifications
Customize
Unsubscribe
You're receiving notifications because you're watching this repository.
1 participant


Aditya01327 / Aditya
Public
Pin
Unwatch 1
Fork
Star 0
Code
Issues
Pull requests 1
Actions
Projects
Wiki
Security
Insights
Settings
added a display function #4
Edit
Code
Closed
aarchi103 wants to merge 1 commit into Aditya01327:main from aarchi103:newb
Conversation 0
Commits 1
Checks 0
Files changed 1
+6 −1
aarchi103 commented 9 minutes ago
Contributor
i have added a display function to add two variables
added a display function
ea2a984
Aditya01327 closed this now
Reviewers
No reviews
Assignees
No one—assign yourself
Labels

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

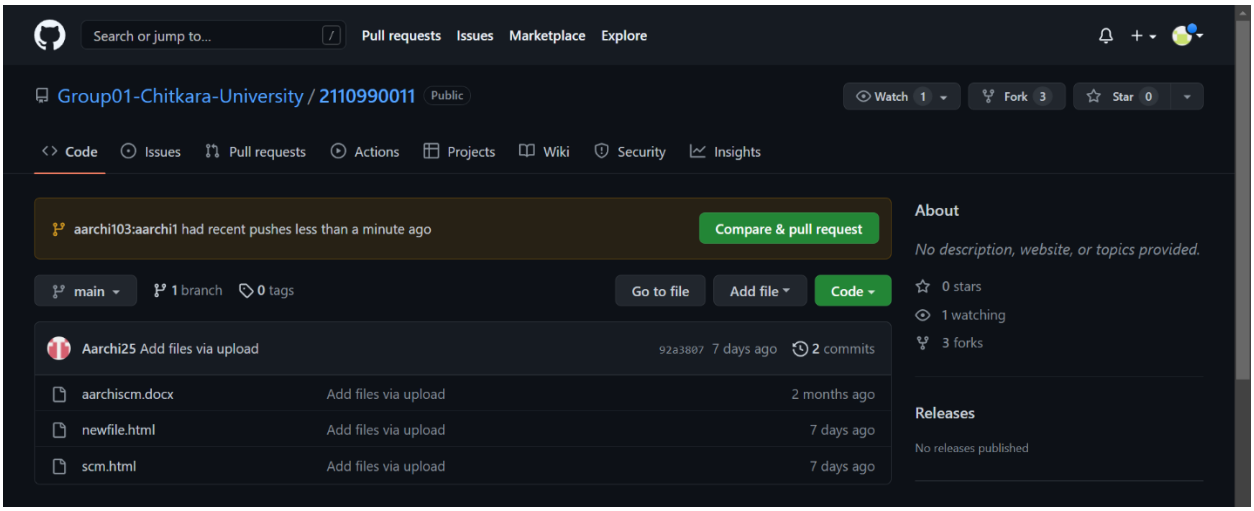
#OPENING PULL REQUESTS ON TEAM MEMBER'S REPOSITORY

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.

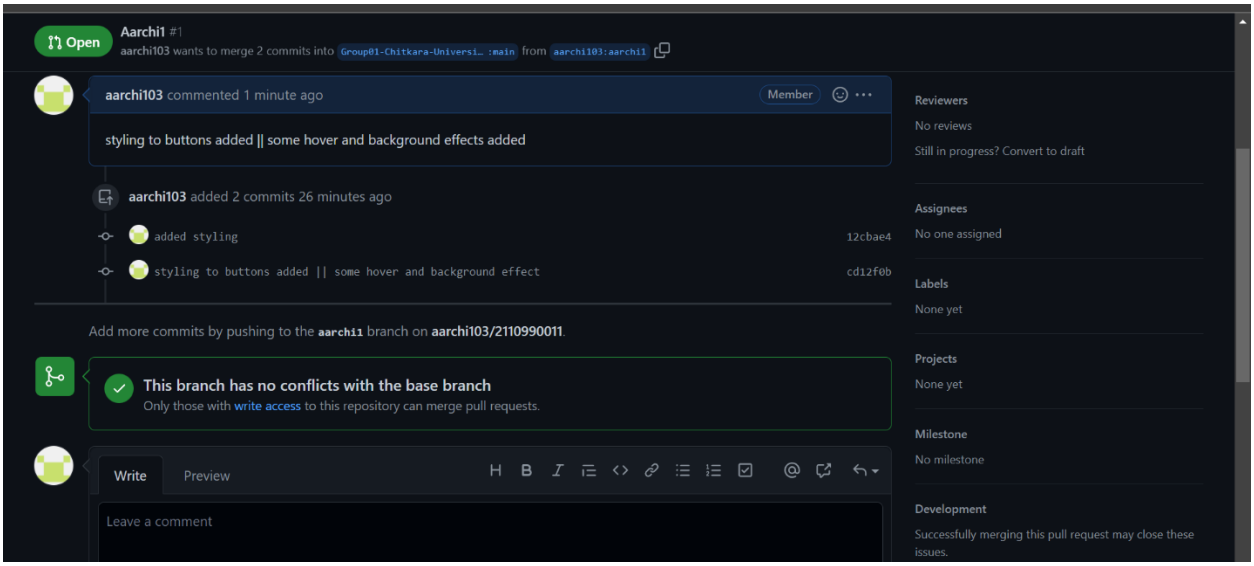
```
ASUS@Aarchi MINGW64 /d
$ git clone https://github.com/aarchi103/2110990011.git
Cloning into '2110990011'...
remote: Enumerating objects: 6, done.
remote: Counting objects: 100% (6/6), done.
remote: Compressing objects: 100% (5/5), done.
remote: Total 6 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (6/6), 2.08 MiB | 2.34 MiB/s, done.

ASUS@Aarchi MINGW64 /d
$ cd 2110990011

ASUS@Aarchi MINGW64 /d/2110990011 (main)
$ git branch
* aarchi
  main

ASUS@Aarchi MINGW64 /d/2110990011 (aarchi)
$ git status
On branch aarchi
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:   newfile.html

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

ASUS@Aarchi MINGW64 /d/2110990011 (aarchi)
$ git commit -a -m "added styling"
[aarchi 12cbae4] added styling
 1 file changed, 87 insertions(+), 77 deletions(-)
 rewrite newfile.html (64%)

ASUS@Aarchi MINGW64 /d/2110990011 (aarchi)
$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

ASUS@Aarchi MINGW64 /d/2110990011 (main)
$ git merge aarchi
Updating 92a3807..12cbae4
Fast-forward
 newfile.html | 100 ++++++++++++++++++++++++++++++++++----------------------------
```

```
ASUS@Aarchi MINGW64 /d/2110990011 (main)
$ git push -u origin main
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 12 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 805 bytes | 402.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/aarchi103/2110990011.git
   92a3807..12cbae4  main -> main
branch 'main' set up to track 'origin/main'.
```

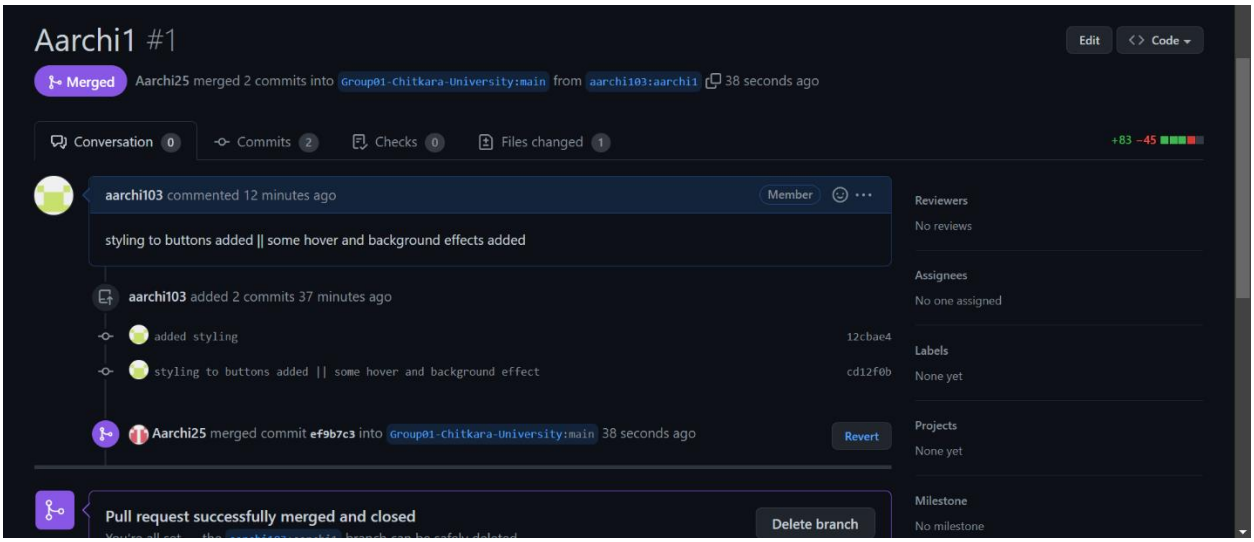
3. Open a pull request by following the procedure from the above experiment.
4. The pull request will be created and will be visible to all the team members.



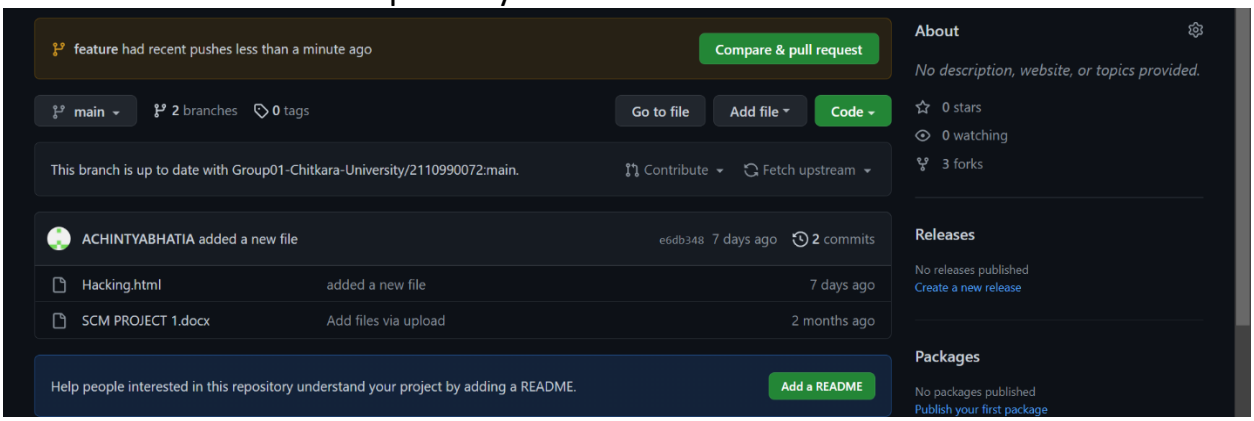
5. Click on compare & pull request option appearing at the top.
6. After you enter a description, click on the open request button.

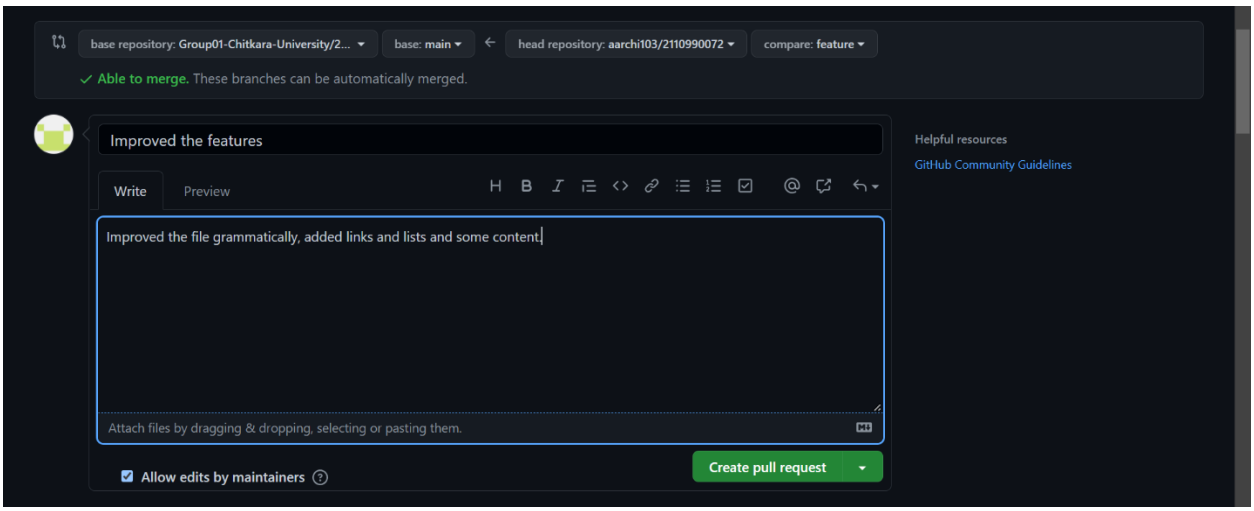


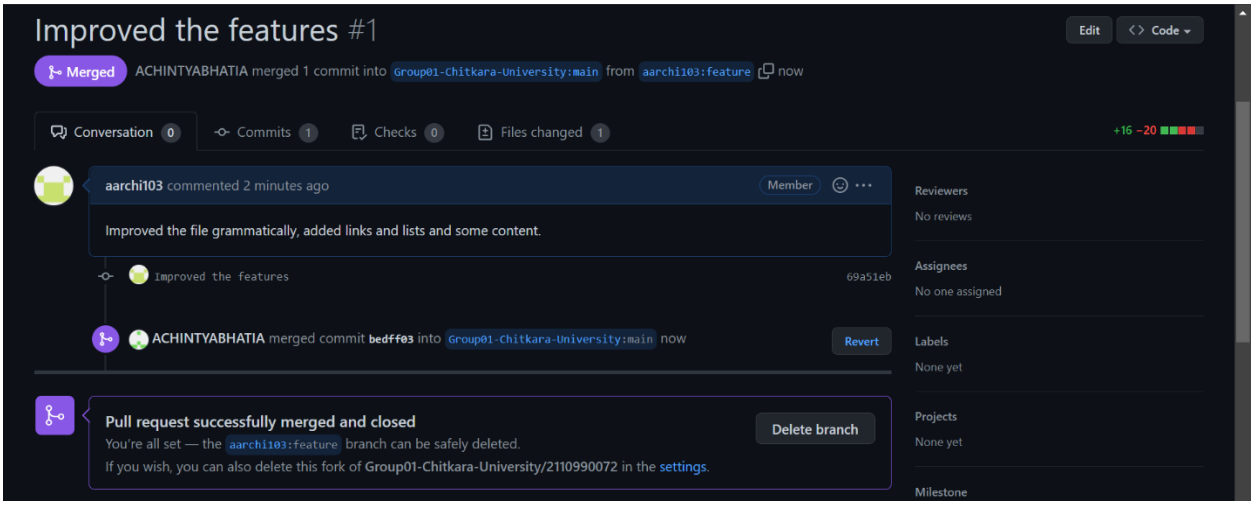
7. Pull request has been sent to your team member. They can choose to either close the pull request or merge it.
8. Suppose they merged it, then the changes you made to the forked repository will be introduced into the owner's original repository and you will be notified about merging.



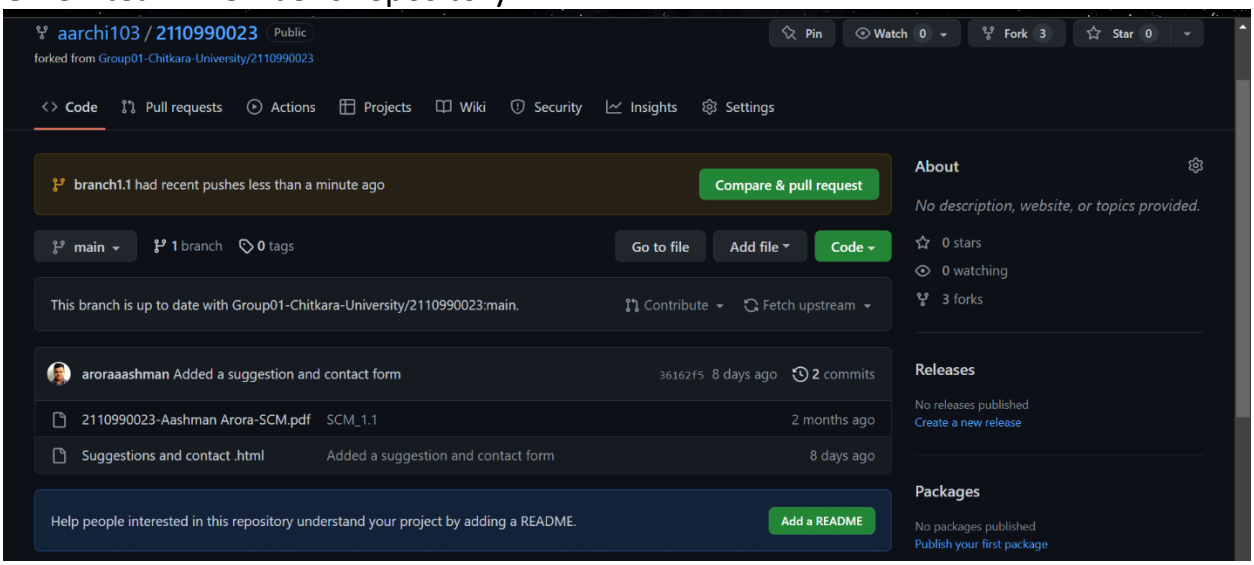
Similarly, create pull requests on other members repositories.
On 2nd team member's repository:

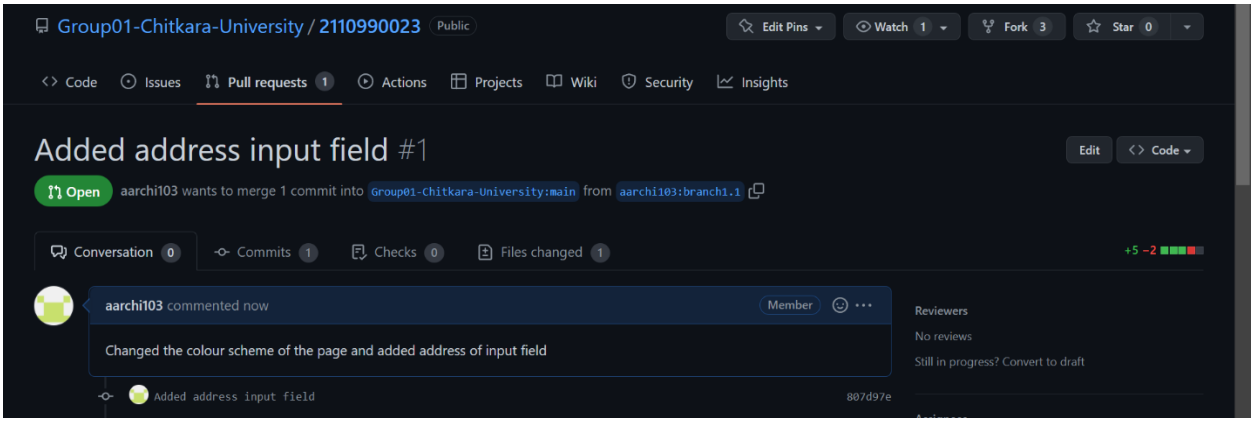


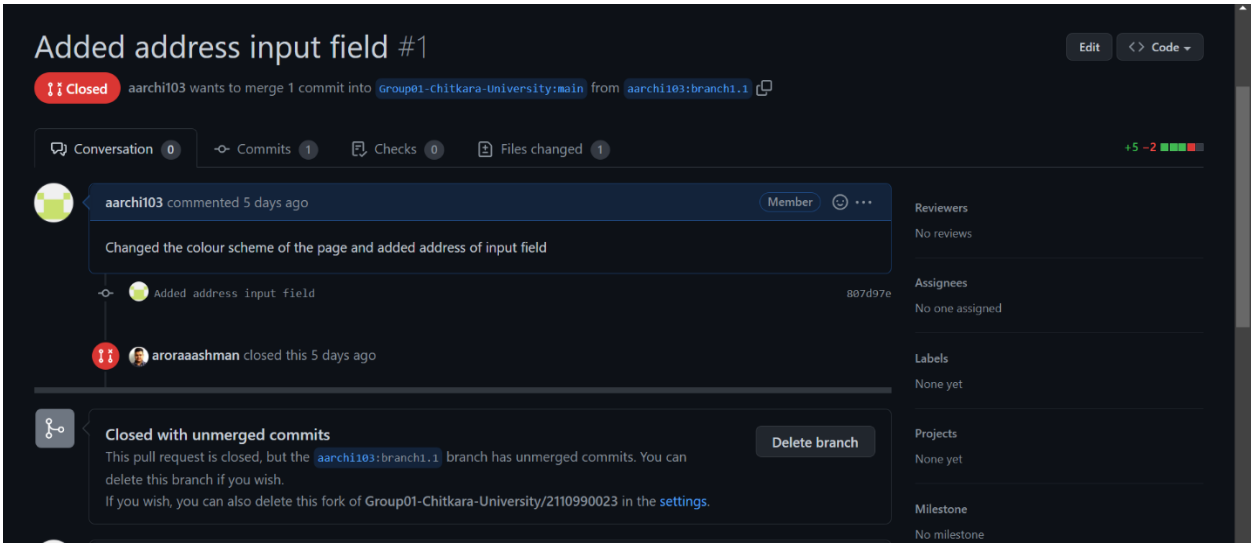




On 3rd team member's repository



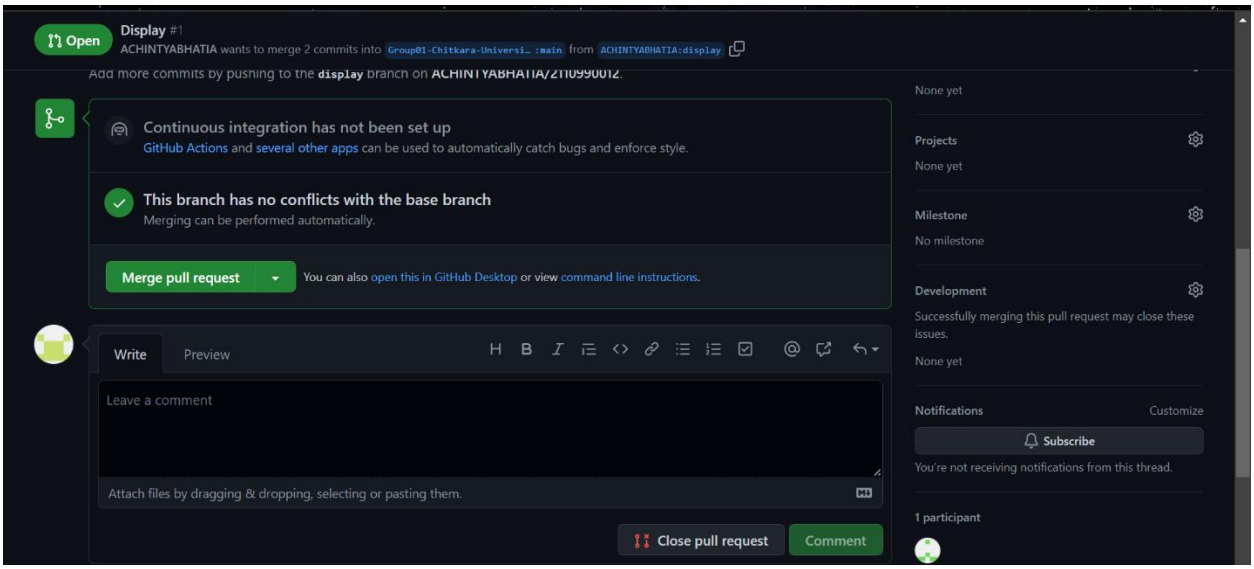




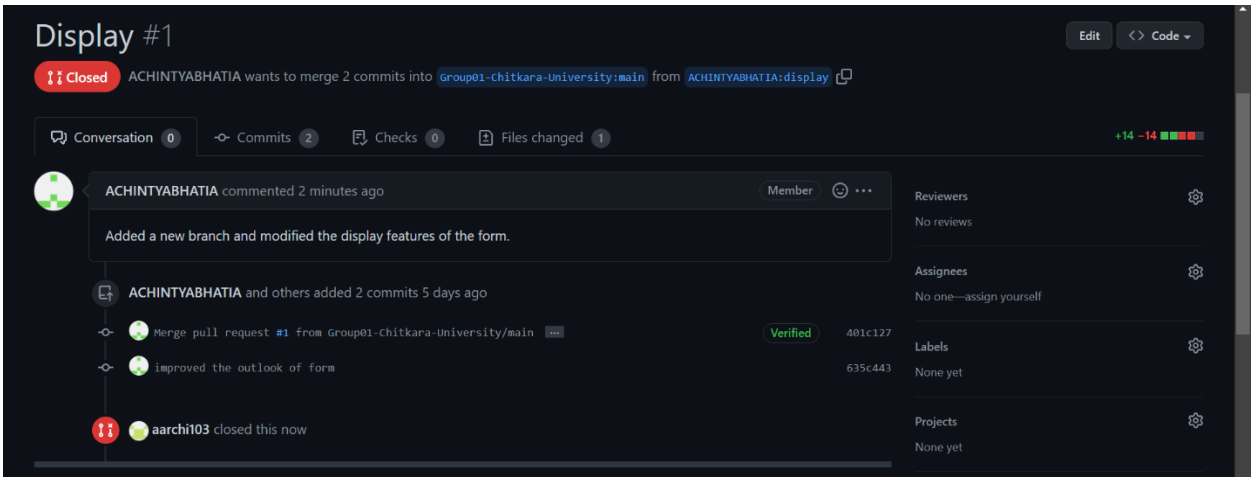
Here, the owner of the repository has closed the pull request without merging the commits that we made.

#CLOSING/MERGING PULL REQUESTS GENERATED BY TEAM MEMBERS

1. In the pull request menu of your repository their will be a notification. Open it.
2. You can either choose to close it without merging the commits or you can merge it.

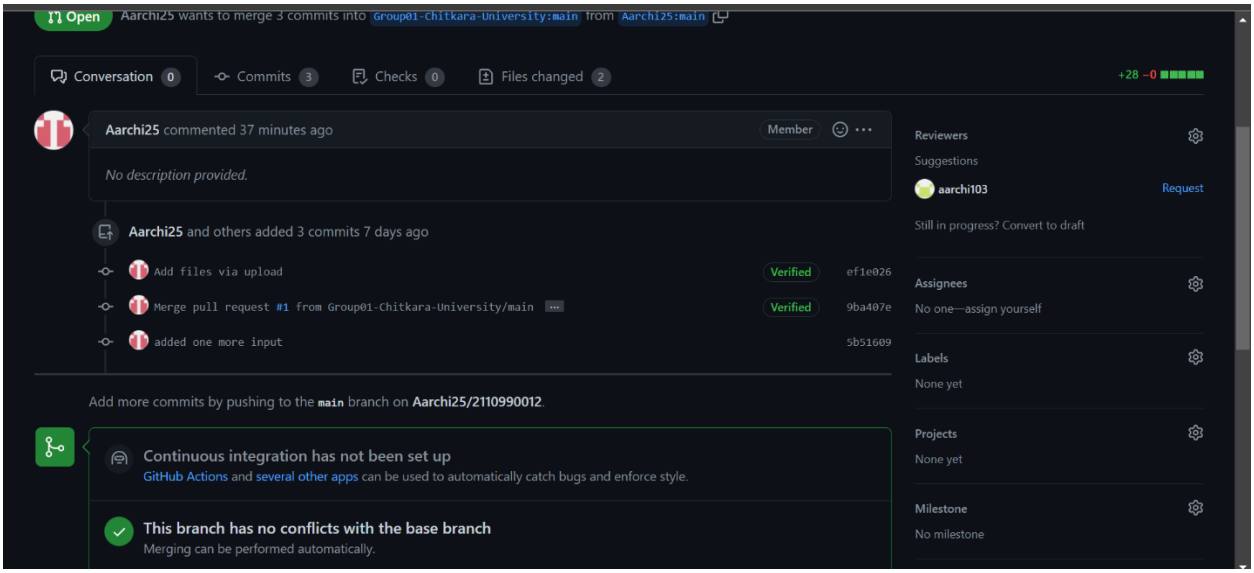


3. After closing it, this screen will appear.

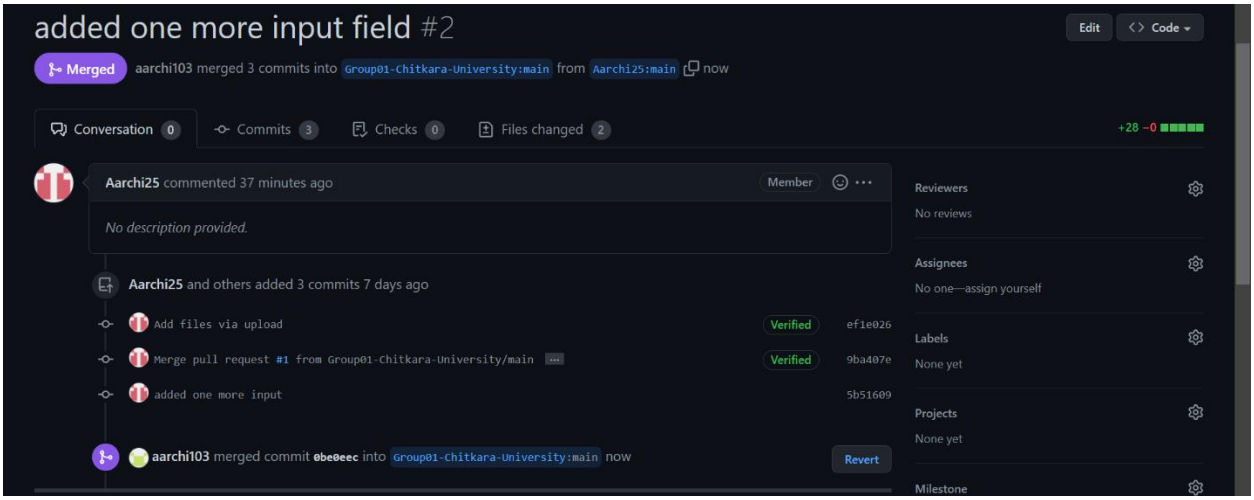


Similarly, close/merging the pull requests of other team members,

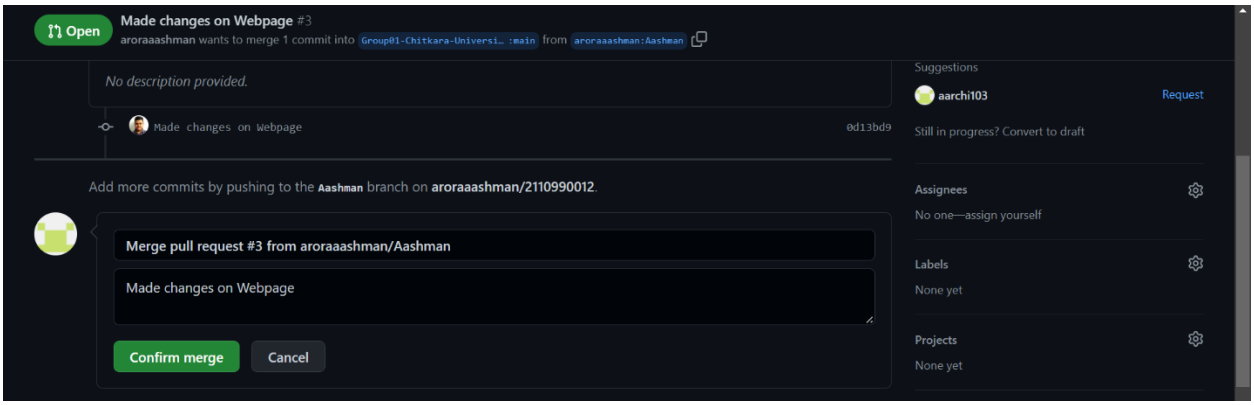
2nd team member:

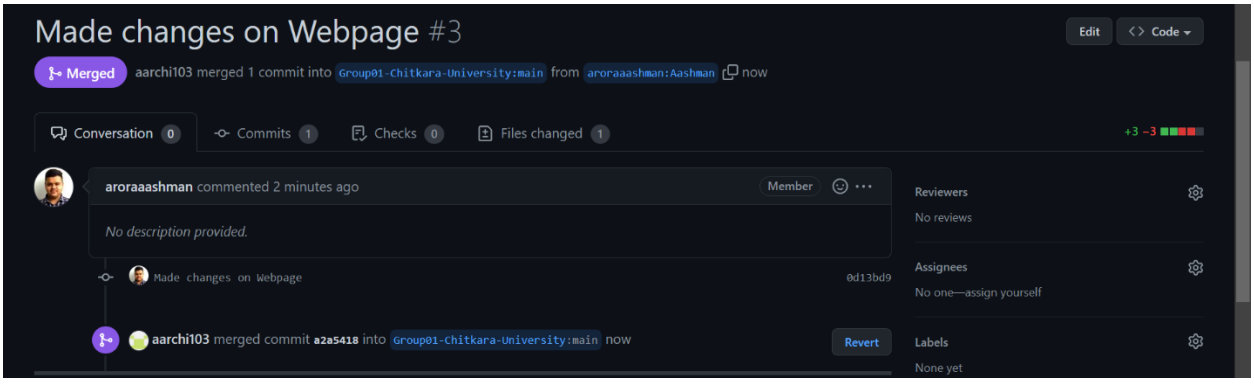


Here, I have chosen to merge the pull request which will introduce the changes made by them into my own repository.

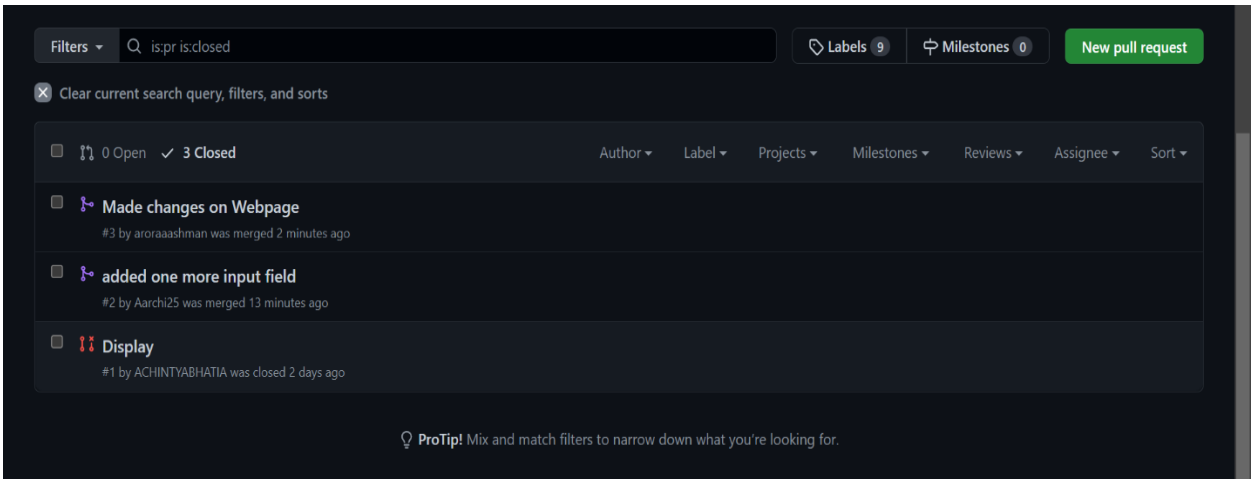


3rd team member:





1. To view the pull requests history of your repository, click on the pull request button in the menu.
2. You will be shown by default the open requests screen.
3. Click on Closed option to see the history.

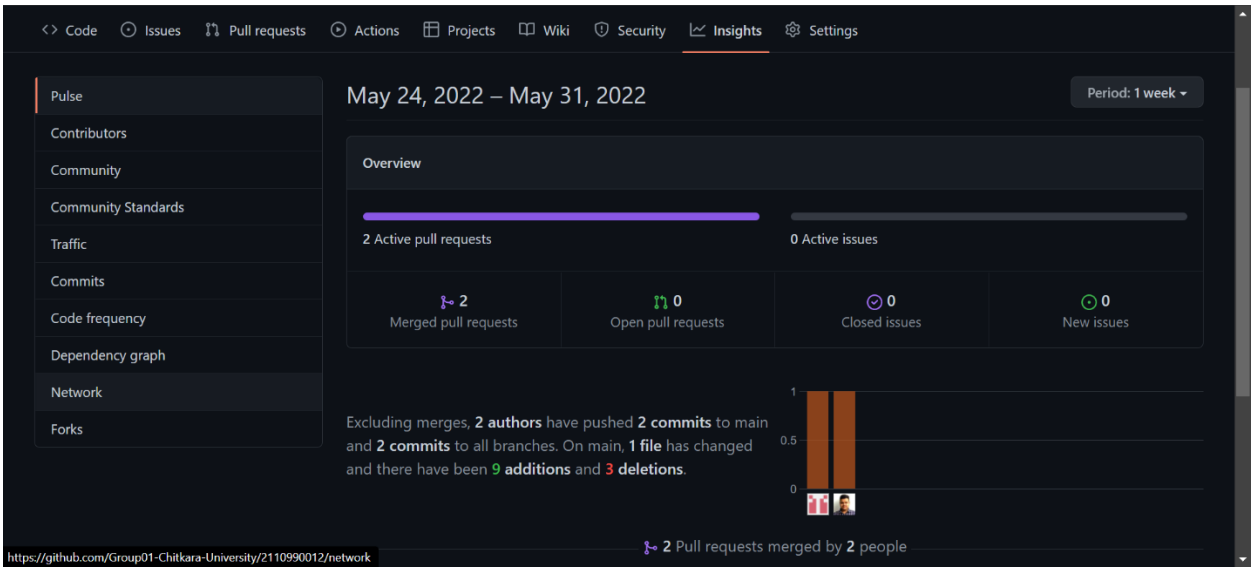


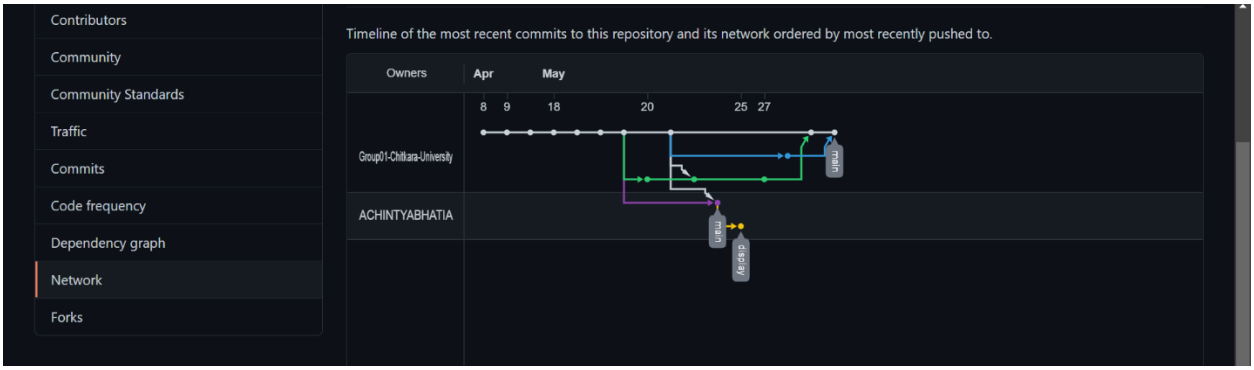
NETWORK GRAPHS

To view the network graphs of your repository, follow the steps:

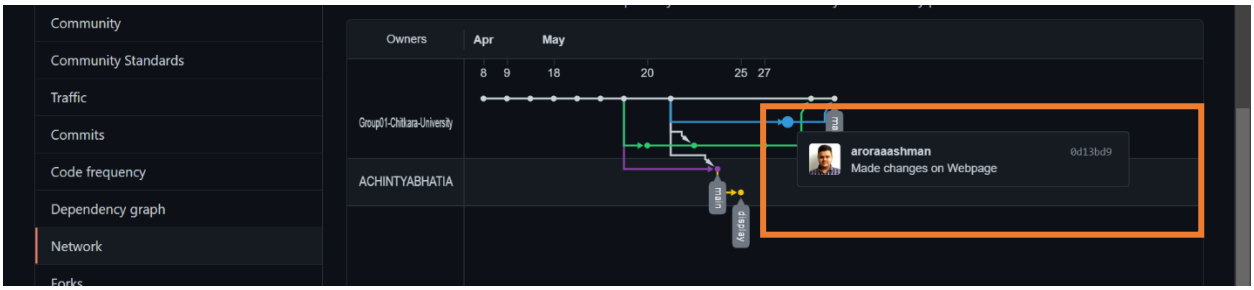
1. Go to the repository of which you want the graph/details.
2. Click on the 'Insights' option it the menu bar.
3. In the right menu list click on network.
4. You can see the network graph there.

It shows the timeline of the most recent commits to this repository and its network ordered by most recently pushed to.





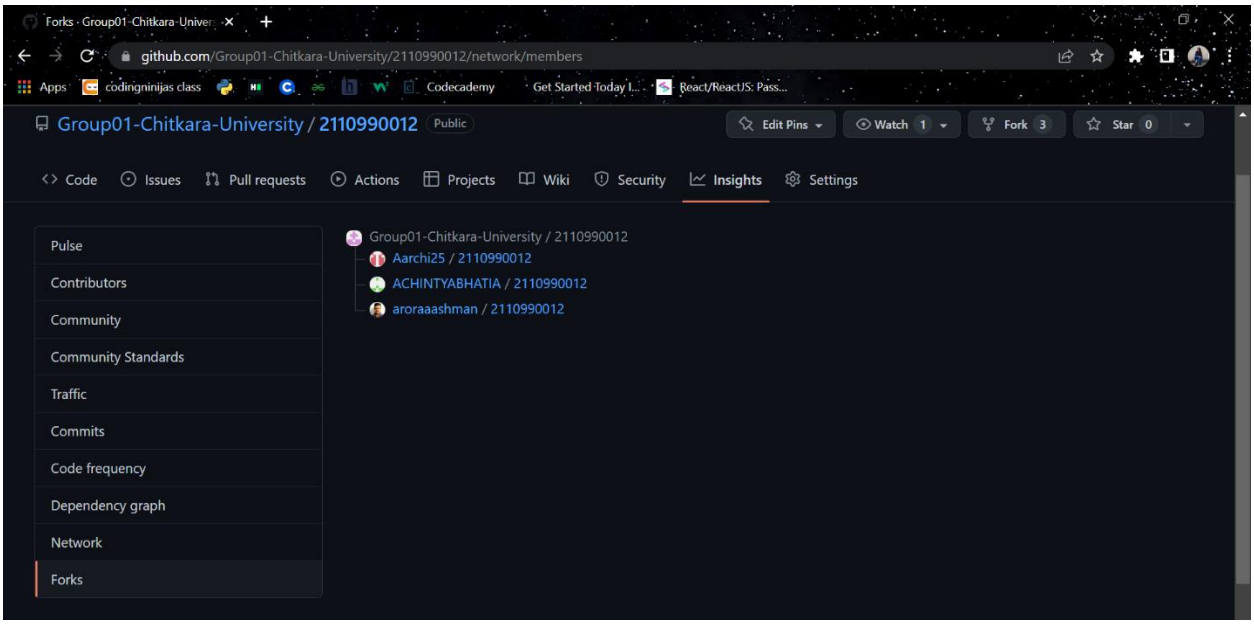
The points in the network graph represents the commits. By hovering over the points, you can see the information about the commit such as author, checksum, message of commit.



LISTING THE FORKS IN THE REPOSITORY

1. Go to the repository of which you want to list the forks.
2. Click on the insights tab.
3. In the left sidebar, click Forks.

4. Here you can see the forks made of your repository.



TEAM MEMBERS

1. **Aarchi Sharma (2110990012)**
2. **Achintya Bhatia (2110990072)**
3. **Aarchi (2110990011)**
4. **Aashman Arora (2110990023)**