

Subject Name: Source Code Management

Subject Code: CS181

Cluster: Beta

Department: CSE

**CHITKARA
UNIVERSITY**



Submitted By-

Aarchi Sharma
2110990012
G01

Submitted To-

Dr. Monit Kapoor

INDEX

S.NO.	TOPIC	Pg.no.
1	Setting up of Git Client	3
2	Setting up GitHub Account	6
3	Generate logs	9
4	Create and visualize branches	9
5	Git lifecycle description	12
6	Add collaborators on GitHub Repo	14
7	Fork and Commit	16
8	Merge and Resolve conflicts created due to own activity and collaborators activity.	18
9	Reset and revert	21
10	Create a distributed Repository and add members in project team	24
11	Open and close a pull request.	27
12	Each project member shall create a pull request on a team members repo and close pull requests generated by team members on own Repo as a maintainer.	31
13	Publish and print network graphs	38

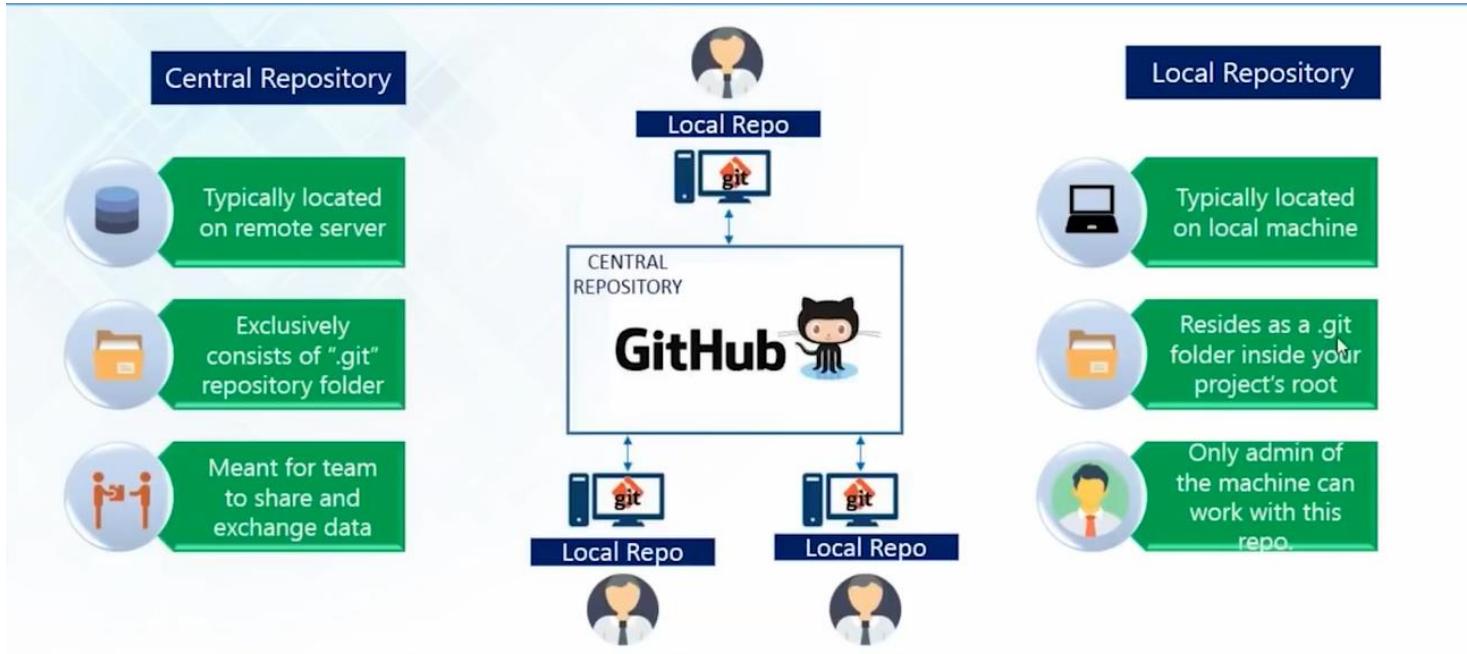
WHAT IS GIT

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

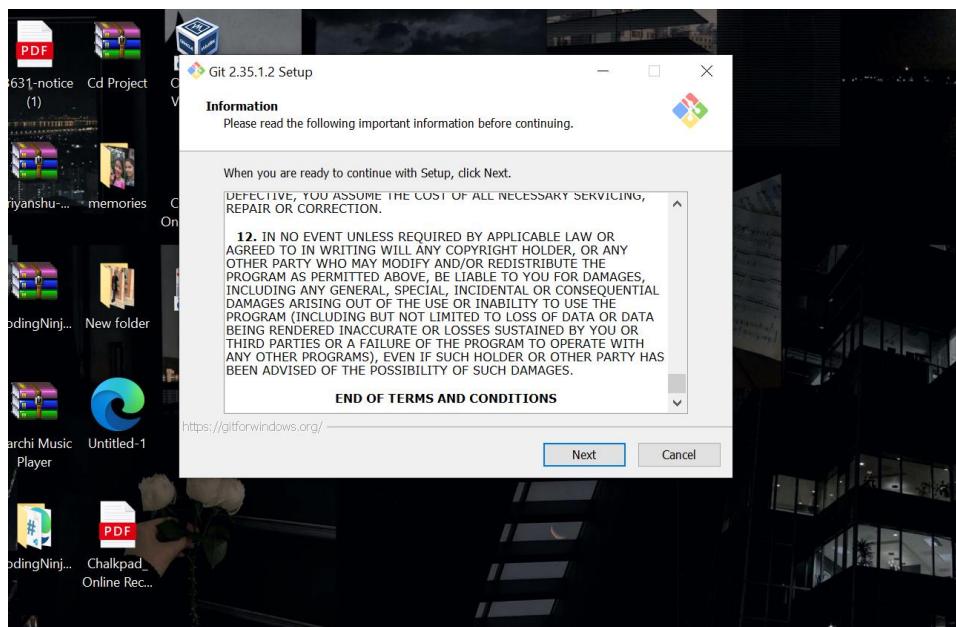
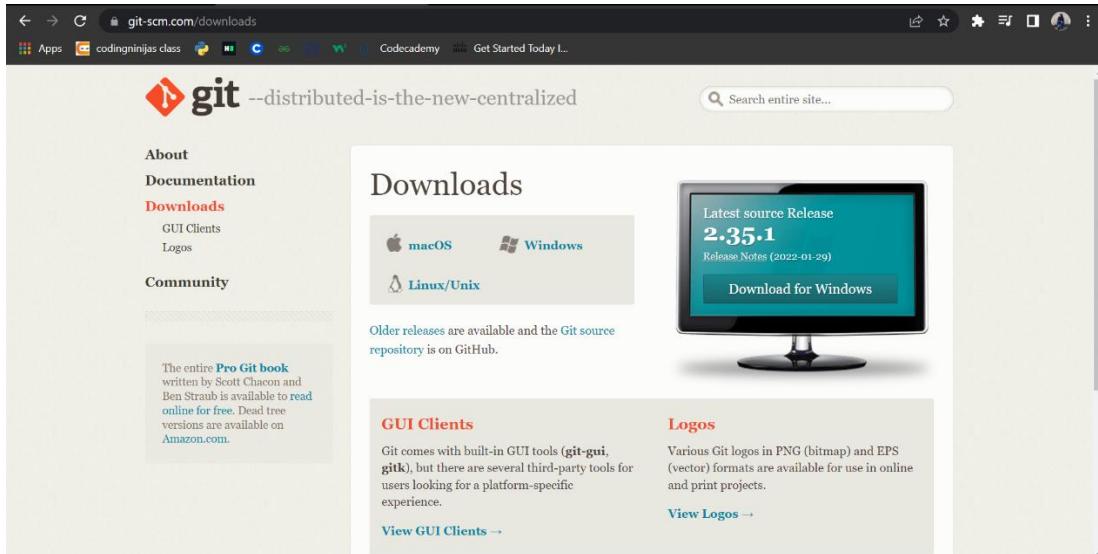
GitHub is a website and cloud-based service that helps developers store and manage their code, as well as track and control changes to their code.

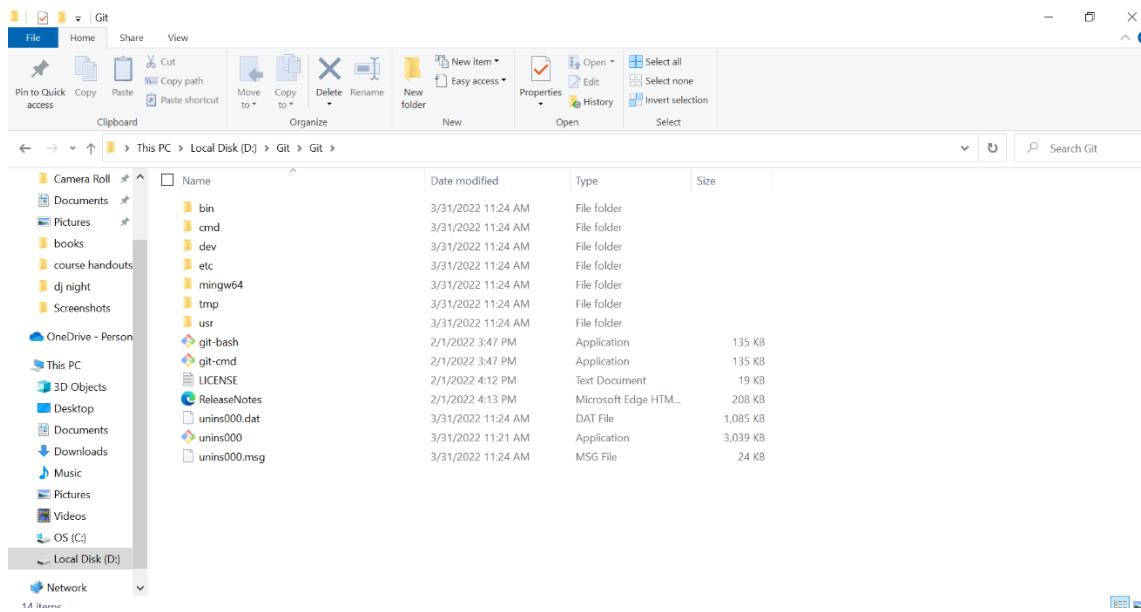
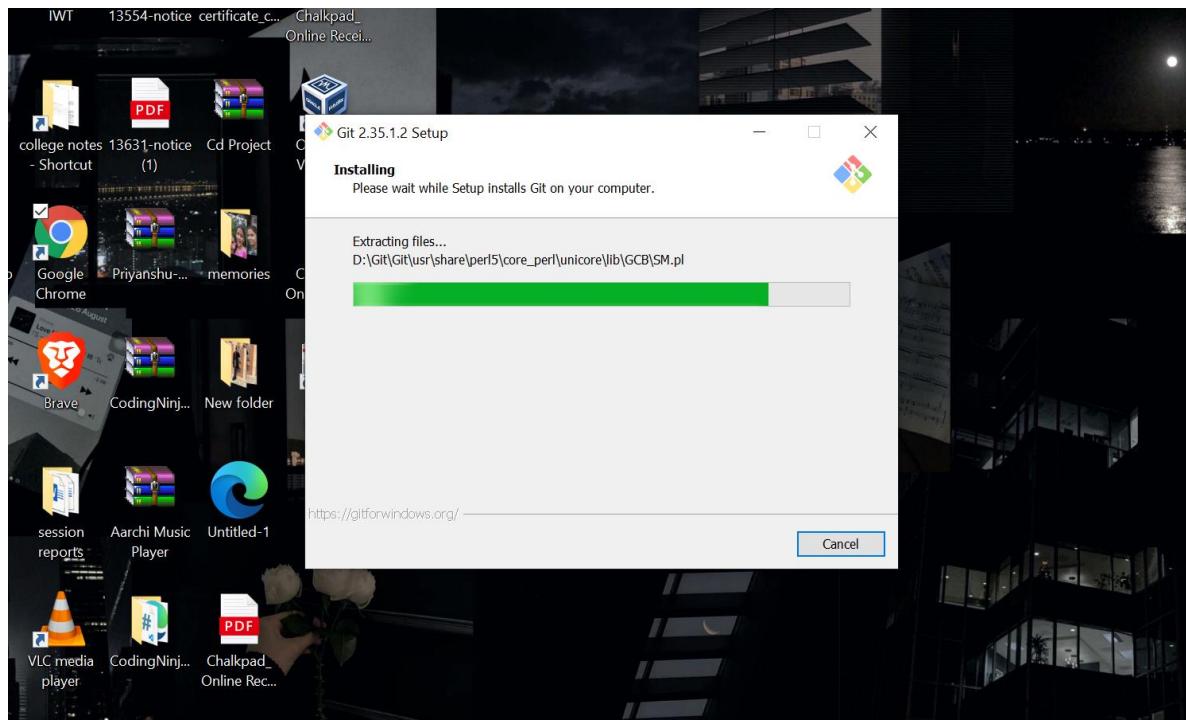
GIT VS GITHUB



INSTALLING GIT

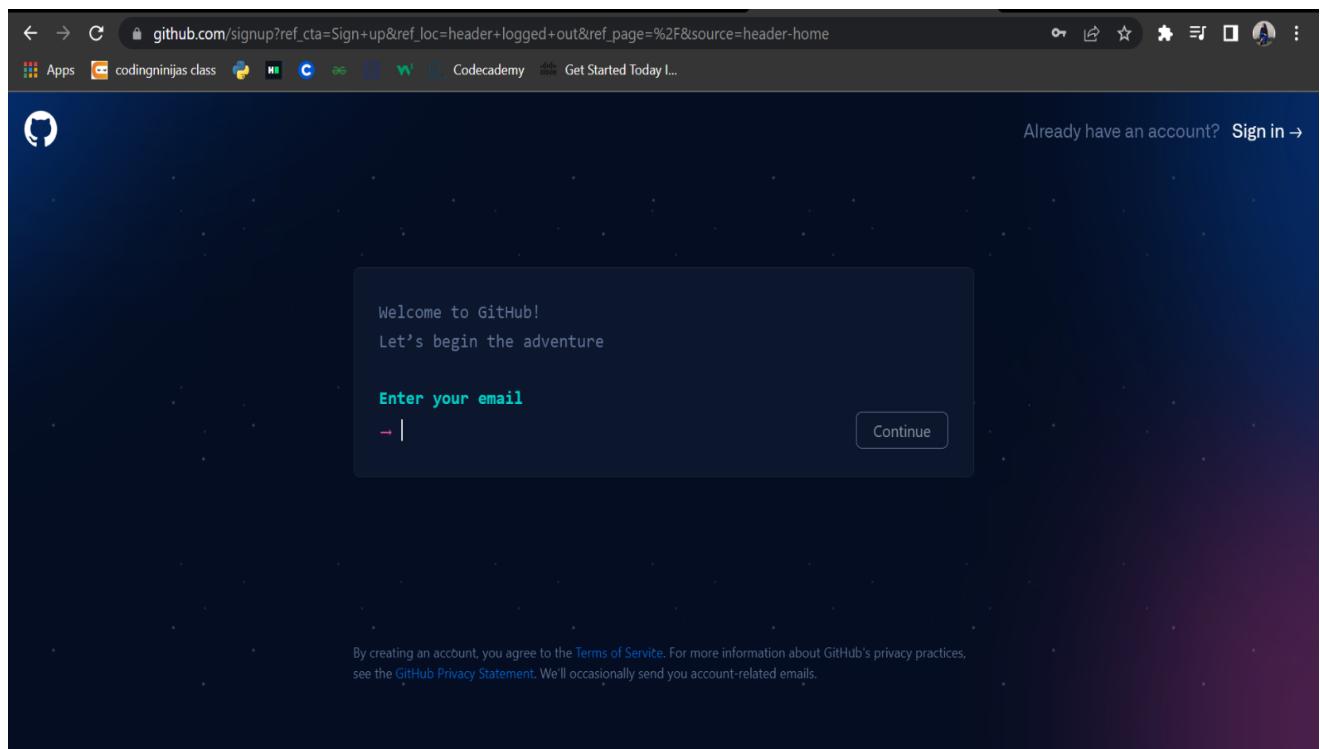
The most official build is available for download on the Git website. Just go to <https://git-scm.com/download/win> and the download will start automatically. Follow the steps listed further to proceed with the installation.





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

```
MINGW64:/c/Users/ASUS
ASUS@Aarchi MINGW64 ~ (master)
$ git config --global user.name "aarchi03"
ASUS@Aarchi MINGW64 ~ (master)
$ git config --global user.email "aarchi0012.be21@chitkara.edu.in"
ASUS@Aarchi MINGW64 ~ (master)
$ git config
usage: git config [<options>]

Config file location
  --global      use global config file
  --system     use system config file
  --local       use repository config file
  --worktree   use per-worktree config file
  -f, --file <file> use given config file
  --blob <blob-id> read config from given blob object

Action
  --get        get value: name [value-pattern]
  --get-all    get all values: key [value-pattern]
  --get-regexp get values for regexp: name-regex [value-pattern]
  --get-urlmatch get value specific for the URL: section[.var] URL
  --replace-all replace all matching variables: name value [value-pattern]
  --add         add a new variable: name value
  --unset      remove a variable: name [value-pattern]
  --unset-all  remove all matches: name [value-pattern]
  --rename-section rename section: old-name new-name
  --remove-section remove a section: name
  -l, --list    list all
  --fixed-value use string equality when comparing values to 'value-pattern'
  -e, --edit    open an editor
  --get-color   find the color configured: slot [default]
  --get-colorbool find the color setting: slot [stdout-is-tty]

Type
  -t, --type <t>  value is given this type
  --bool        value is "true" or "false"
  --int         value is decimal number
  --bool-or-int value is --bool or --int
  --bool-or-str value is --bool or string
  --path        value is a path (file or directory name)
  --expiry-date value is an expiry date

Other
  -z, --null    terminate values with NUL byte
  --name-only   show variable names only
  --includes    respect include directives on lookup
```

MAKING A DIRECTORY AND VARIOUS COMMANDS

```
MINGW64:/d/g1/scmproject
ASUS@Aarchi MINGW64 /d/g1
$ git config --global user.name "aarchi03"
ASUS@Aarchi MINGW64 /d/g1
$ git config --global user.email "aarchi0012.be21@chitkara.edu.in"
ASUS@Aarchi MINGW64 /d/g1
$ mkdir scmproject
ASUS@Aarchi MINGW64 /d/g1
$ cd scmproject
ASUS@Aarchi MINGW64 /d/g1/scmproject
$ pwd
/d/g1/scmproject
ASUS@Aarchi MINGW64 /d/g1/scmproject
$ ls
ASUS@Aarchi MINGW64 /d/g1/scmproject
$ vi first.cpp
ASUS@Aarchi MINGW64 /d/g1/scmproject
$ ls
first.cpp
ASUS@Aarchi MINGW64 /d/g1/scmproject
$ cat first.cpp
#include <iostream>
using namespace std;
int main(){
cout<<"Hello World" << endl;
}
ASUS@Aarchi MINGW64 /d/g1/scmproject
$
```

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

```

MINGW64:/d/g1/scmproject
$ git status
On branch master

No commits yet

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

nothing added to commit but untracked files present (use "git add" to track)

ASUS@Aarchi MINGW64 /d/g1/scmproject (master)
$ git add first.cpp
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 (master)
$ git status
On branch master

No commits yet

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

ASUS@Aarchi MINGW64 /d/g1/scmproject (master)
$ git commit -m "first commit | repo demo for class"
[master (root-commit) b91cfad] first commit | repo demo for class
 1 file changed, 5 insertions(+)
 create mode 100644 first.cpp

ASUS@Aarchi MINGW64 /d/g1/scmproject (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.

```

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
```

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
```

```
$ 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:

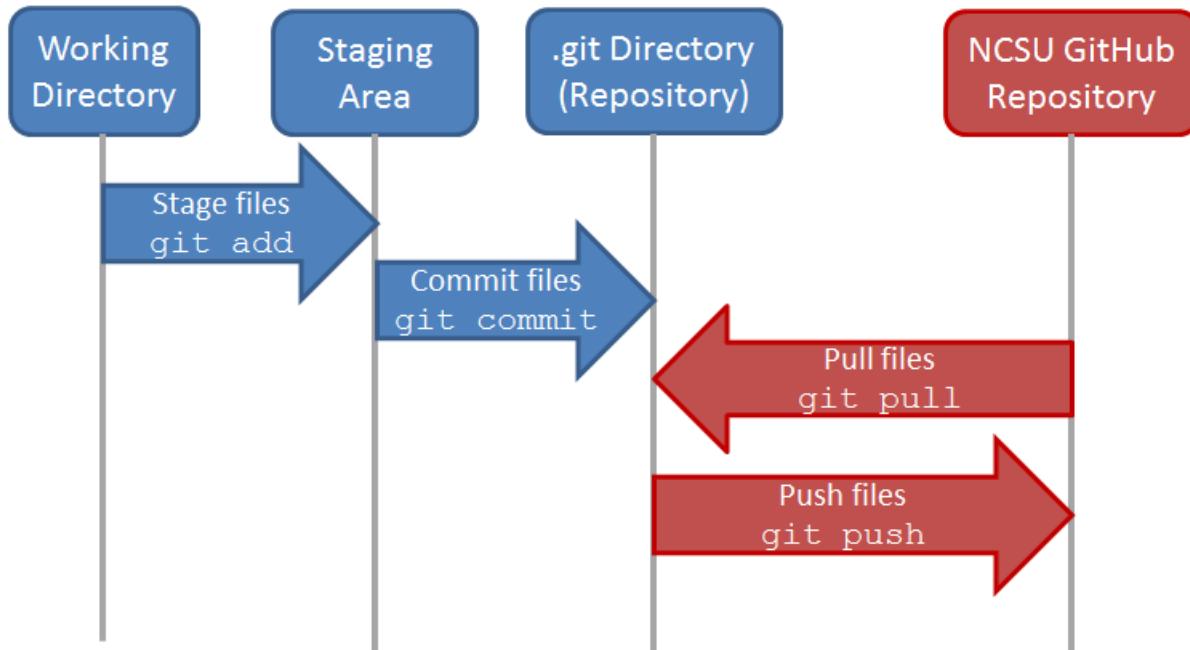
Staging area is the playground where you group, add and organize the files to be committed to Git for tracking their versions.

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

Who has access

Access

- Collaborators**
- Moderation options

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.

Code and automation

- Actions
- Webhooks
- Environments
- Pages

Security

https://github.com/aarchi103/SCMPROJECT/settings/access

github.com/aarchi103/SCMPROJECT/settings/access

Code and automation

- Actions
- Webhooks
- Environments
- Pages

Security

- Code security and analysis
- Deploy keys
- Secrets

Integrations

- GitHub apps
- Email notifications

Add a collaborator to SCMPROJECT

Q ACHINTYA

- Achintya Bhatia • Invite collaborator
- Achintya Sharma • achintya • Invite collaborator
- achintya • achintya-7 • Invite collaborator
- Achintya • Achintya94 • Invite collaborator
- Achintya • achintya20 • Invite collaborator

© 2022 GitHub, Inc. Terms Privacy Security

Training Blog About

Windows Taskbar: File Explorer, Spotify, Mail, Word, Edge, Google Chrome

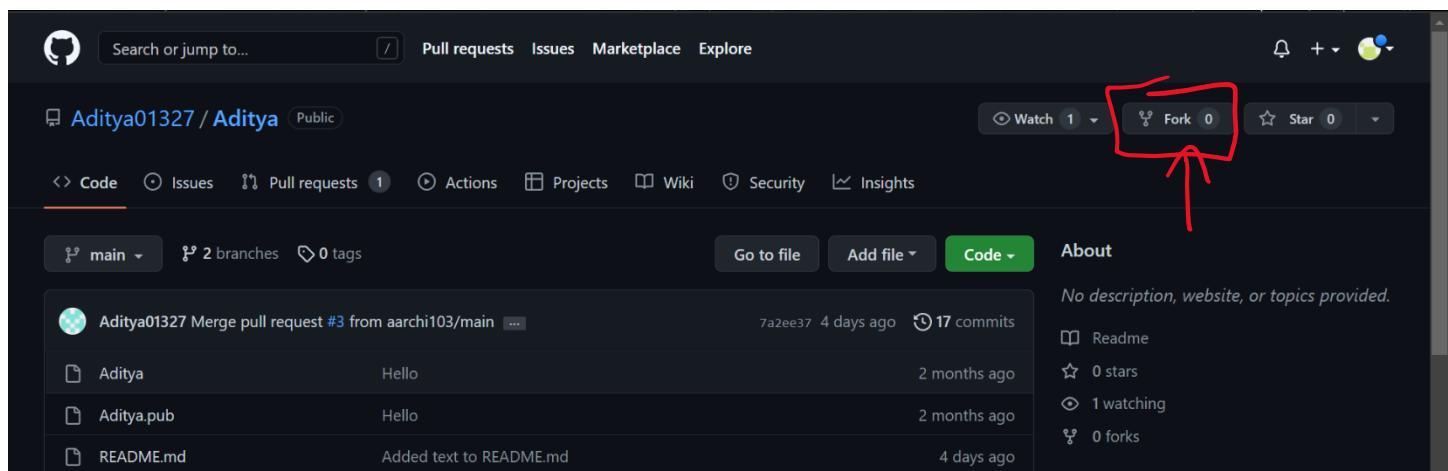
System Tray: Cloud, 33°C Mostly cloudy, ENG IN, 1:45 PM, 5/17/2022

FORK AND COMMIT

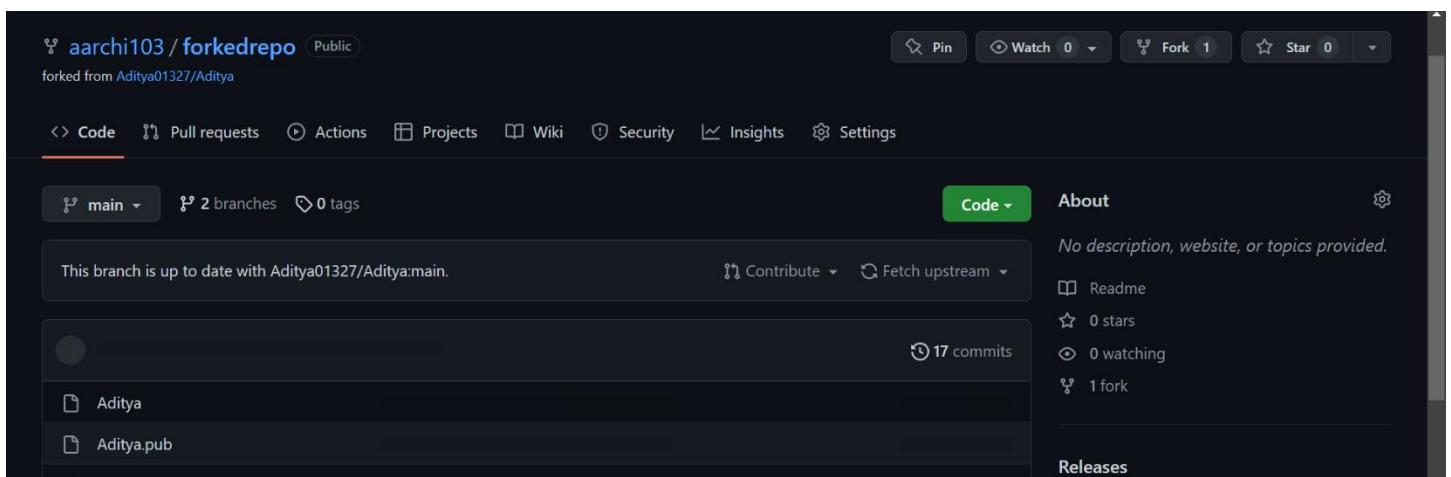
A fork is a copy of a repository. Forking a repository allows you to freely experiment with changes without affecting the original project. Most commonly, forks are used to either propose changes to someone else's project to which you do not have write access, or to use someone else's project as a starting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

```
ASUS@Aarchi MINGW64 /d/forkedrepo
$ git clone https://github.com/aarchi103/forkedrepo.git
Cloning into 'forkedrepo'...
remote: Enumerating objects: 51, done.
remote: Counting objects: 100% (51/51), done.
remote: Compressing objects: 100% (35/35), done.
remote: Total 51 (delta 12), reused 24 (delta 4), pack-reused 0
Receiving objects: 100% (51/51), 12.12 KiB | 376.00 KiB/s, done.
Resolving deltas: 100% (12/12), done.

ASUS@Aarchi MINGW64 /d
$ cd forkedrepo

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

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.

```
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 branch"
}

~<76.cpp [dos] (14:07 11/05/2022)1,1 All ~<76.cpp [dos] (14:07 11/05/2022)1,1 All ~<76.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;
    cout<<"Hello this is soemthing new added"
}

first.cpp [dos] (14:07 11/05/2022)
"first.cpp" [dos] 15L, 264B
10,9-16 All

```

```

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

```

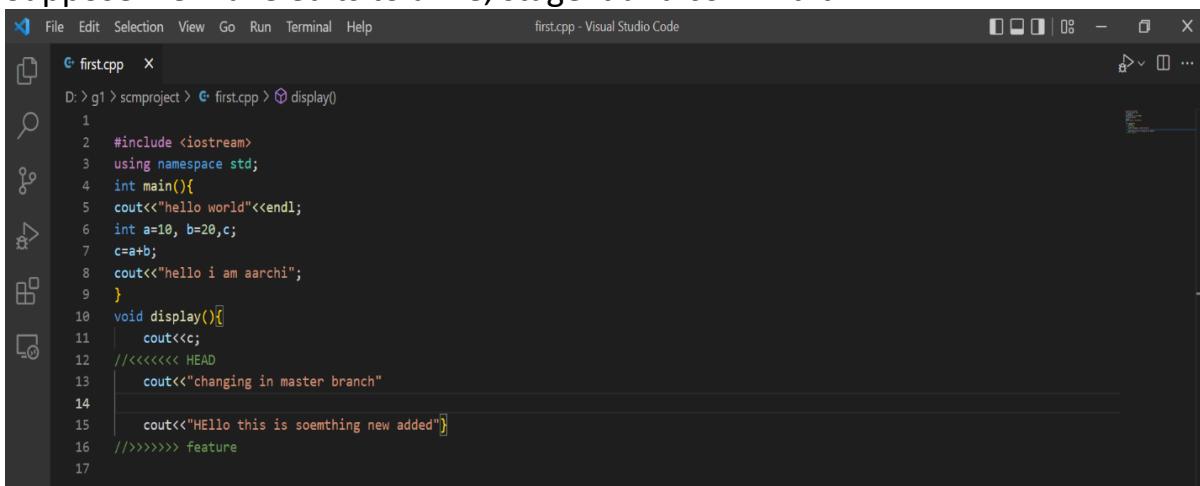
RESET AND REVERT

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.

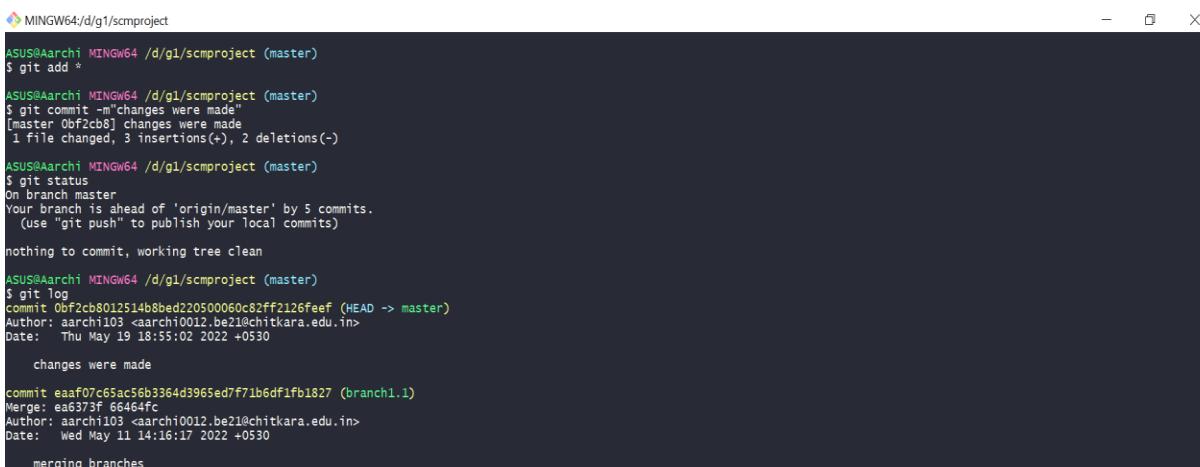
RESET-

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.



```
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<<"hello i am aarchi";
9 }
10 void display(){[REDACTED]
11     cout<<c;
12 //<<<<< HEAD
13     cout<<"changing in master branch"
14
15     cout<<"Hello this is something new added"]
16 //>>>> feature
17 }
```



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

ASUS@Aarchi MINGW64 /d/g1/scmproject (master)
$ git commit -m"changes were made"
[master 0bf2cb6] changes were made
1 file changed, 3 insertions(+), 2 deletions(-)

ASUS@Aarchi MINGW64 /d/g1/scmproject (master)
$ git status
On branch master
Your branch is ahead of 'origin/master' by 5 commits.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean

ASUS@Aarchi MINGW64 /d/g1/scmproject (master)
$ git log
commit 0bf2cb8012514b8bed220500060c82ff2126feef (HEAD -> master)
Author: aarchi103 <aarchi1012.be21@chitkara.edu.in>
Date:   Thu May 19 18:55:02 2022 +0530

    changes were made

commit eaaf07c65ac56b3364d3965ed7f71b6df1fb1827 (branch1.1)
Merge: ea6373f 66464fc
Author: aarchi103 <aarchi1012.be21@chitkara.edu.in>
Date:   Wed May 11 14:16:17 2022 +0530

    merging branches
```

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

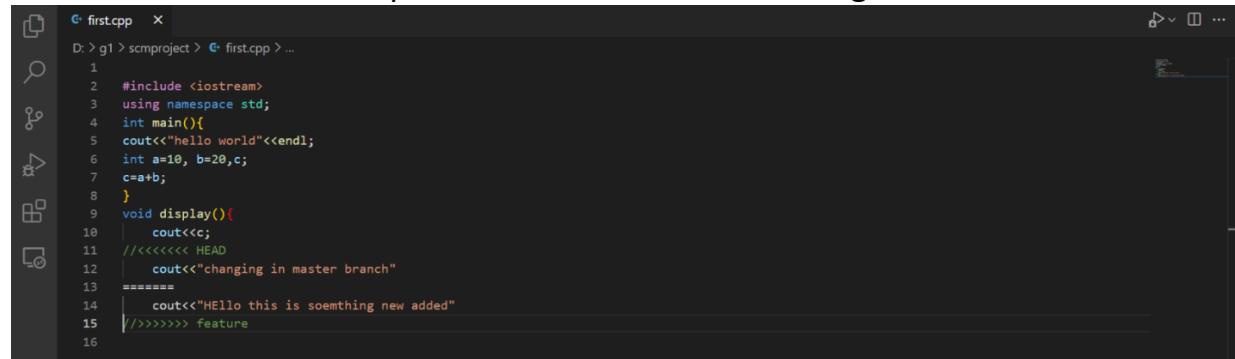
ASUS@Aarchi MINGW64 /d/g1/scmproject (master)
$ git log
commit eaaf07c65ac56b3364d3965ed7f71b6df1fb1827 (HEAD -> master, branch1.1)
Merge: ea6373f 66464fc
Author: aarchi103 <aarchi10012.be21@chitkara.edu.in>
Date:   Wed May 11 14:16:17 2022 +0530

        merging branches

commit ea6373ffd30a15adb1753fb34df0e6355475d51a
Author: aarchi103 <aarchi10012.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  X
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 something 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 X
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 something new added"
16 //>>>>>>> feature
17
```

```
MINGW64:/d/g1/scmproject
ASUS@Aarchi MINGW64 /d/g1/scmproject (master)
$ git add *
ASUS@Aarchi MINGW64 /d/g1/scmproject (master)
$ git commit -m"changes made"
[master cdf7f75] changes made
1 file changed, 2 insertions(+), 1 deletion(-)
ASUS@Aarchi MINGW64 /d/g1/scmproject (master)
$ git log
commit cdf7f75a2b0897fe0b267f07bcc1032e6d056559 (HEAD -> master)
Author: aarchi103 <aarchi10012.be21@chitkara.edu.in>
Date: Thu May 19 19:01:01 2022 +0530
    changes made
commit eaaf07c65ac56b3364d3965ed7f71b6df1fb1827 (branch1.1)
Merge: ea6373f 66464fc
Author: aarchi103 <aarchi10012.be21@chitkara.edu.in>
Date: Wed May 11 14:16:17 2022 +0530
merging branches
```

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

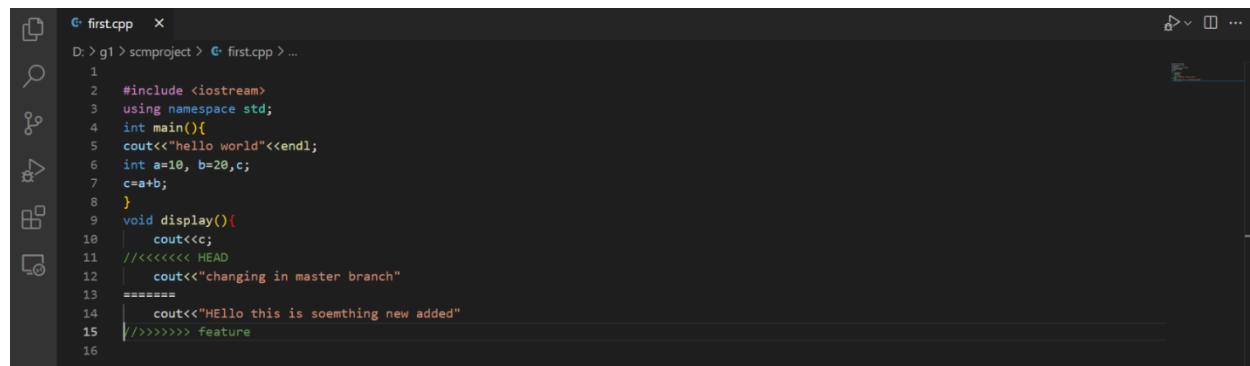
```
ASUS@Aarchi MINGW64 /d/g1/scmproject (master)
$ git revert cdf7f75a2b0897fe0b267f07bcc1032e6d056559
[master baa5a37] Revert "changes made"
1 file changed, 1 insertion(+), 2 deletions(-)

ASUS@Aarchi MINGW64 /d/g1/scmproject (master)
$ git log
commit baa5a370fd219bdc5c37707c74e725c15c5679e (HEAD -> master)
Author: aarchi103 <aarchi10012.be21@chitkara.edu.in>
Date: Thu May 19 19:01:38 2022 +0530
    Revert "changes made"

This reverts commit cdf7f75a2b0897fe0b267f07bcc1032e6d056559.

commit cdf7f75a2b0897fe0b267f07bcc1032e6d056559
Author: aarchi103 <aarchi10012.be21@chitkara.edu.in>
Date: Thu May 19 19:01:01 2022 +0530
    changes made
commit eaaf07c65ac56b3364d3965ed7f71b6df1fb1827 (branch1.1)
Merge: ea6373f 66464fc
Author: aarchi103 <aarchi10012.be21@chitkara.edu.in>
Date: Wed May 11 14:16:17 2022 +0530
merging branches
```

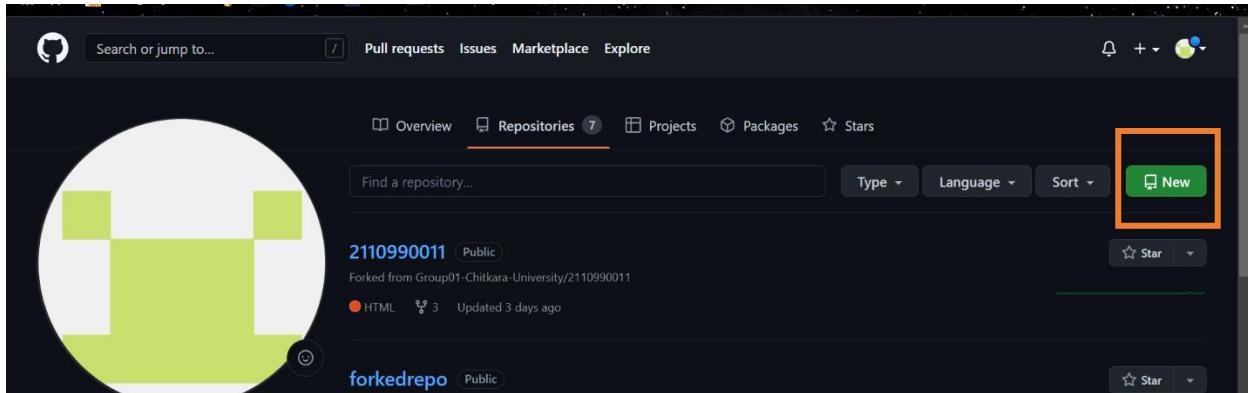
You can see that a new commit as ‘revert “changes made”’ is there and the file has returned to its previous state.



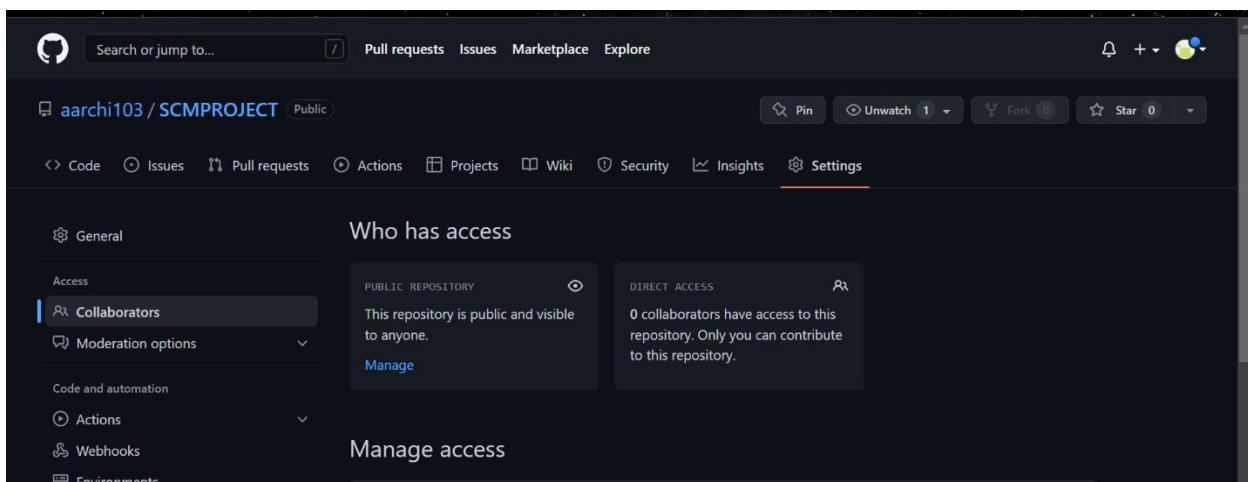
```
D: > g1 > scmproject > firstcpp > ...
1
2 #include <iostream>
3 using namespace std;
4 int main(){
5     cout<<"Hello world"<<endl;
6     int a=10, b=20,c;
7     c=a+b;
8 }
9 void display(){
10    cout<<c;
11 //<<<<< HEAD
12 //    cout<<"changing in master branch"
13 =====
14 //    cout<<"HEllo this is soemthing new added"
15 //>>>> feature
16
```

CREATE A DISTRIBUTED REPOSITORY AND ADD MEMBERS IN PROJECT TEAM

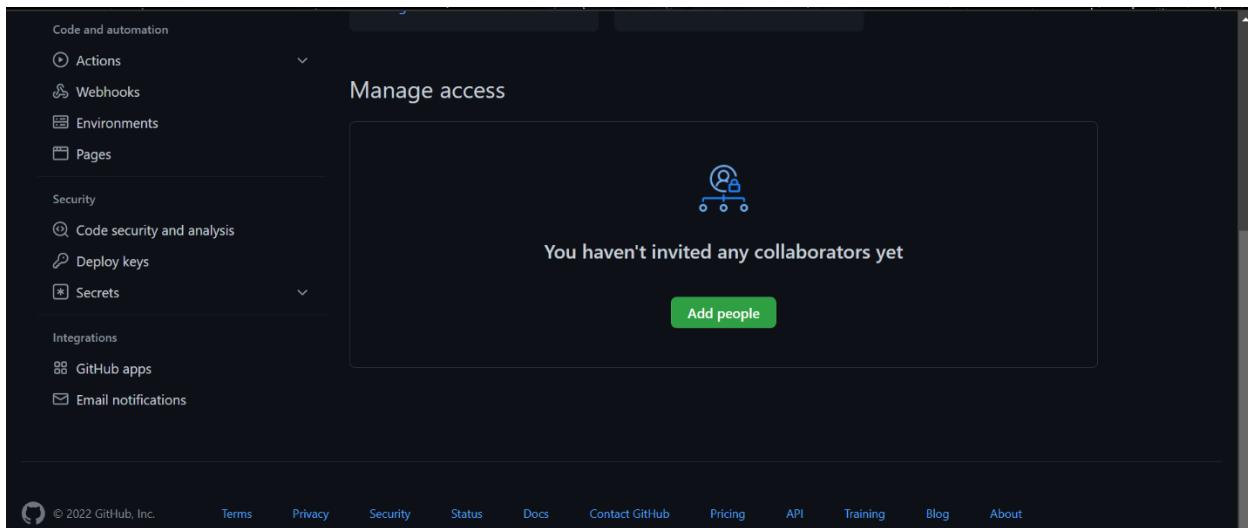
1. On the homepage of your GitHub account, click on Repositories option in the menu bar.
2. Click on the ‘New’ button in the top right corner.



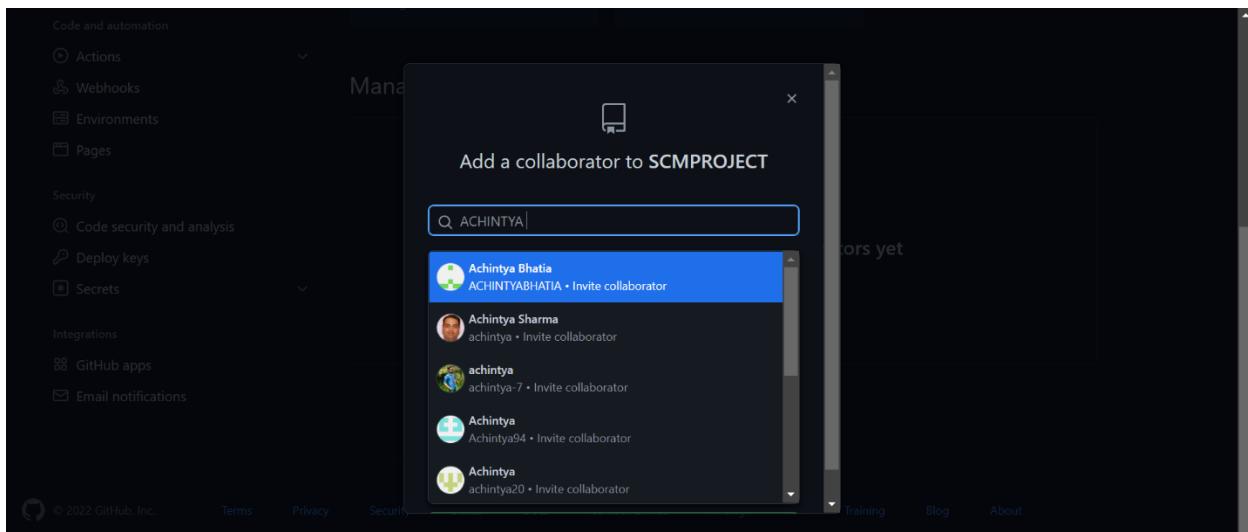
3. Enter the Repository name and add the description of the repository.
4. To add members to your repository, open your repository and select settings option in the navigation bar.
5. Click on Collaborators option under the access tab.



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.

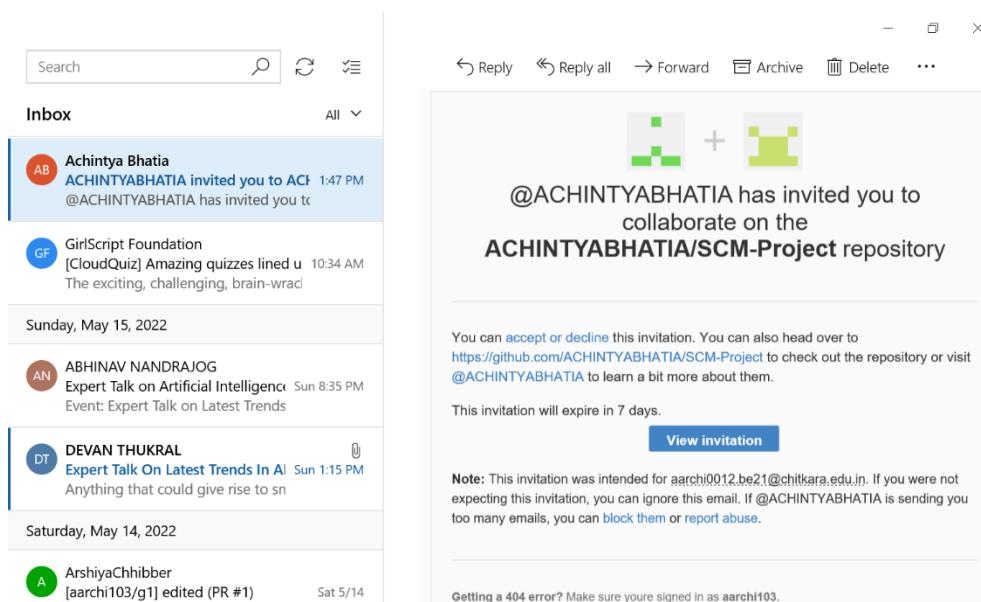


The screenshot shows the 'Manage access' section of a GitHub repository. On the left, a sidebar lists various repository settings: Actions, Webhooks, Environments, Pages, Security (Code security and analysis, Deploy keys, Secrets), Integrations (GitHub apps, Email notifications), and a general 'Code and automation' section. The main area displays a message: 'You haven't invited any collaborators yet' with a 'Add people' button.



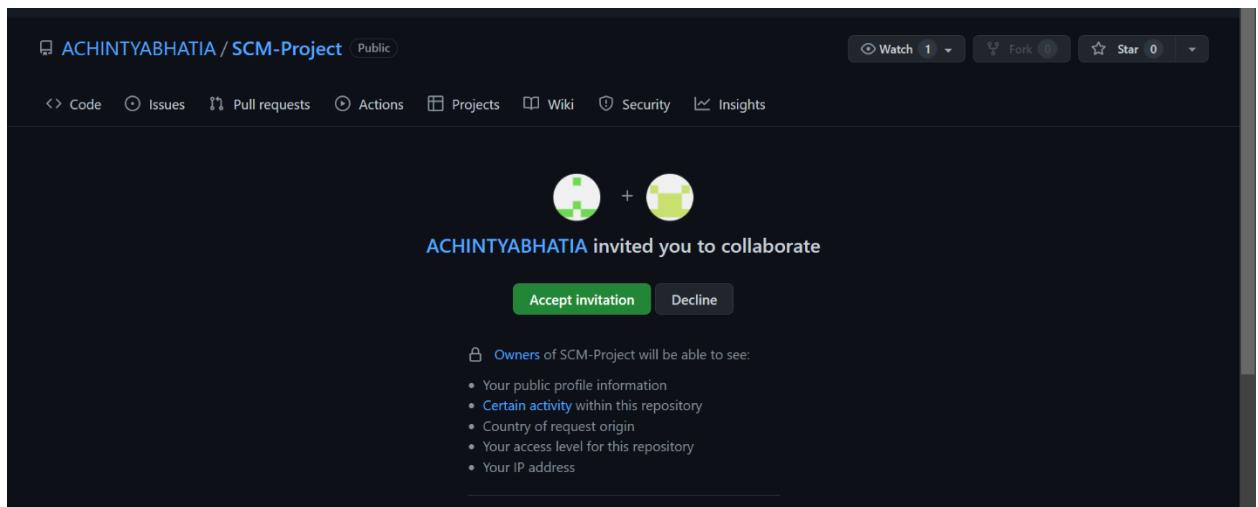
The screenshot shows the 'Add a collaborator' modal. A search bar at the top contains the text 'Q ACHINTYA'. Below it, a list of users is displayed, each with a profile picture, name, and a link to invite them: 'Achintya Bhatia ACHINTYABHATIA + Invite collaborator', 'Achintya Sharma achintya + Invite collaborator', 'achintya achintya-7 + Invite collaborator', 'Achintya Achintya94 + Invite collaborator', and 'Achintya achintya20 + Invite collaborator'.

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.



The screenshot shows an email inbox with several messages. One message from 'Achintya Bhatia' at 1:47 PM invites the user to a GitHub repository. The message includes a note about accepting or declining the invitation, a 'View invitation' button, and a note about the invitation expiring in 7 days. The GitHub URL is provided as <https://github.com/ACHINTYABHATIA/SCM-Project>.

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



The screenshot shows a GitHub repository page for 'ACHINTYABHATIA / SCM-Project'. A collaboration invitation is displayed, showing two user icons with a plus sign between them. The text reads 'ACHINTYABHATIA invited you to collaborate'. Below this are two buttons: 'Accept invitation' (green) and 'Decline' (grey). A note below the buttons states: 'Owners of SCM-Project will be able to see:'. It lists five items: 'Your public profile information', 'Certain activity within this repository', 'Country of request origin', 'Your access level for this repository', and 'Your IP address'.

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.

```
MINGW64:/d/forkedrepo
ASUS@aarchi MINGW64 /d
$ git clone https://github.com/aarchi103/forkedrepo.git
Cloning into 'forkedrepo'...
remote: Enumerating objects: 51, done.
remote: Counting objects: 100% (51/51), done.
remote: Compressing objects: 100% (35/35), done.
remote: Total 51 (delta 12), reused 24 (delta 4), pack-reused 0
Receiving objects: 100% (51/51), 12.12 KiB | 376.00 KiB/s, done.
Resolving deltas: 100% (12/12), done.

ASUS@aarchi MINGW64 /d
$ cd forkedrepo

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")

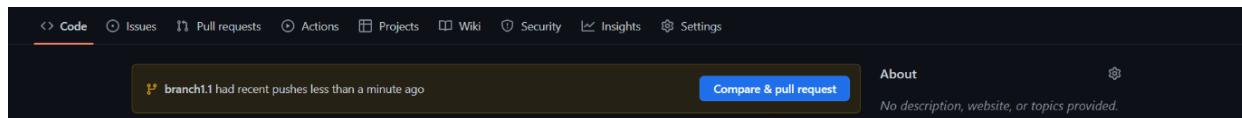
ENG 7:15 PM  
IN 5/18/2022
```

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

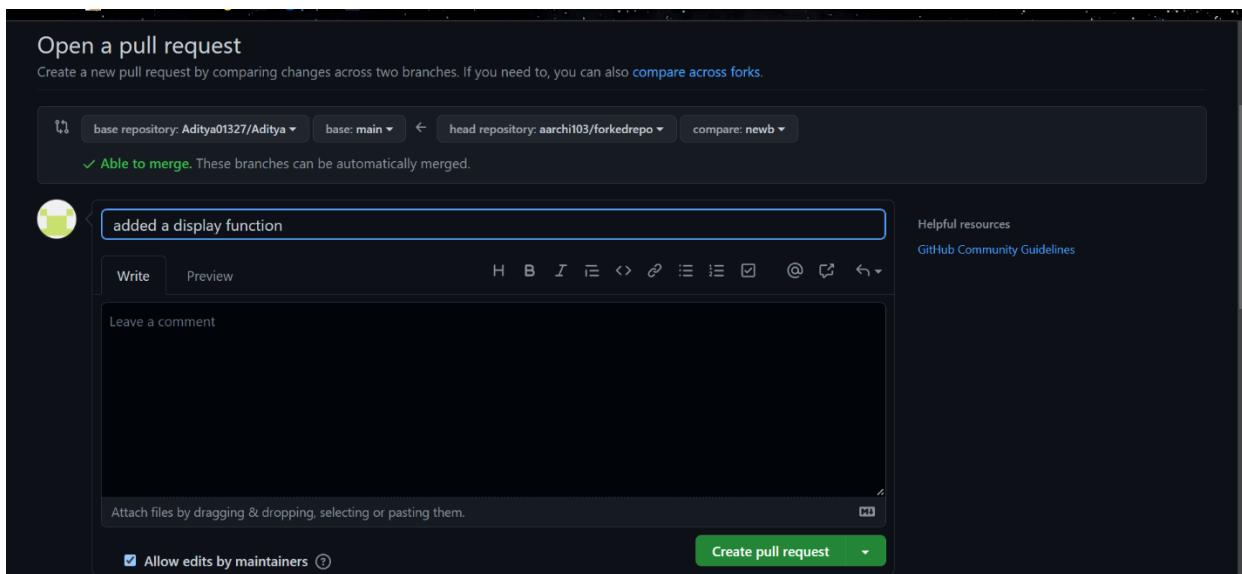
```
ASUS@Aarchi MINGW64 /d/forkedrepo (newb)
$ git push -u origin newb
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 12 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 356 bytes | 356.00 KiB/s, done.
Total 3 (delta 2), reused 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)
$ |
```

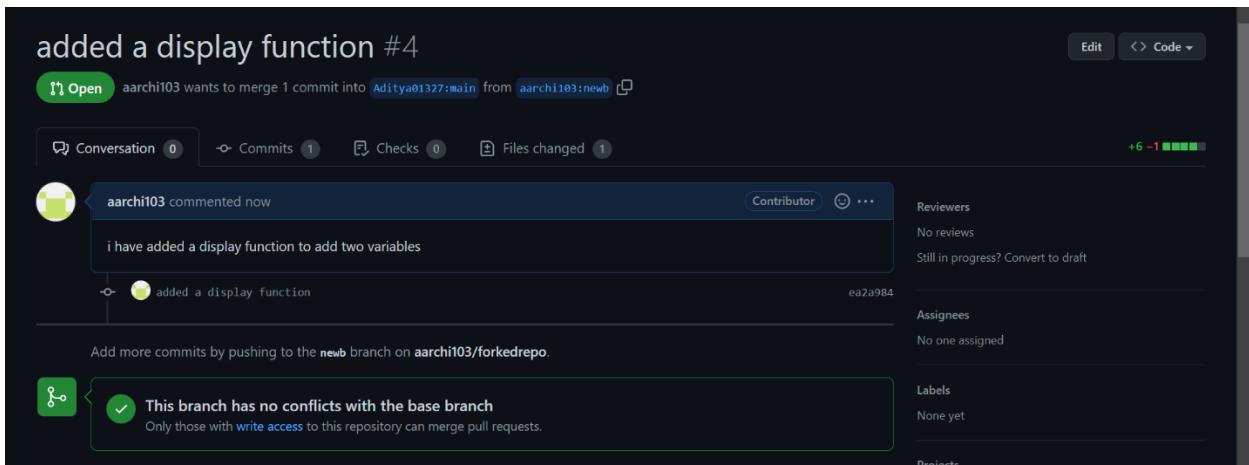
- After pushing new branch GitHub will either automatically ask you to create a pull request or you can create your own pull request.



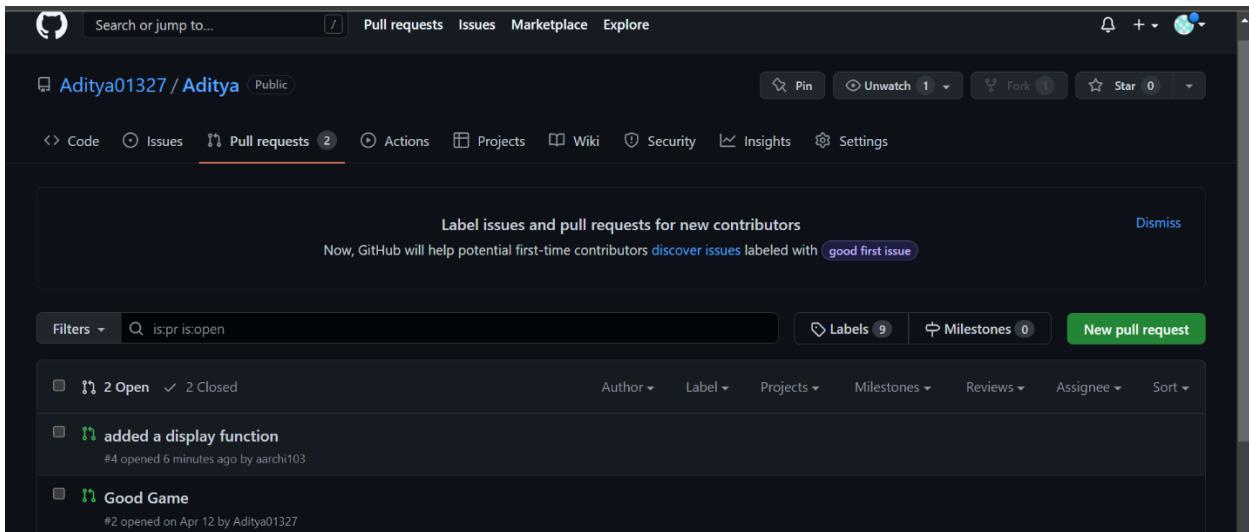
- To create your own pull request, click on pull request option.



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



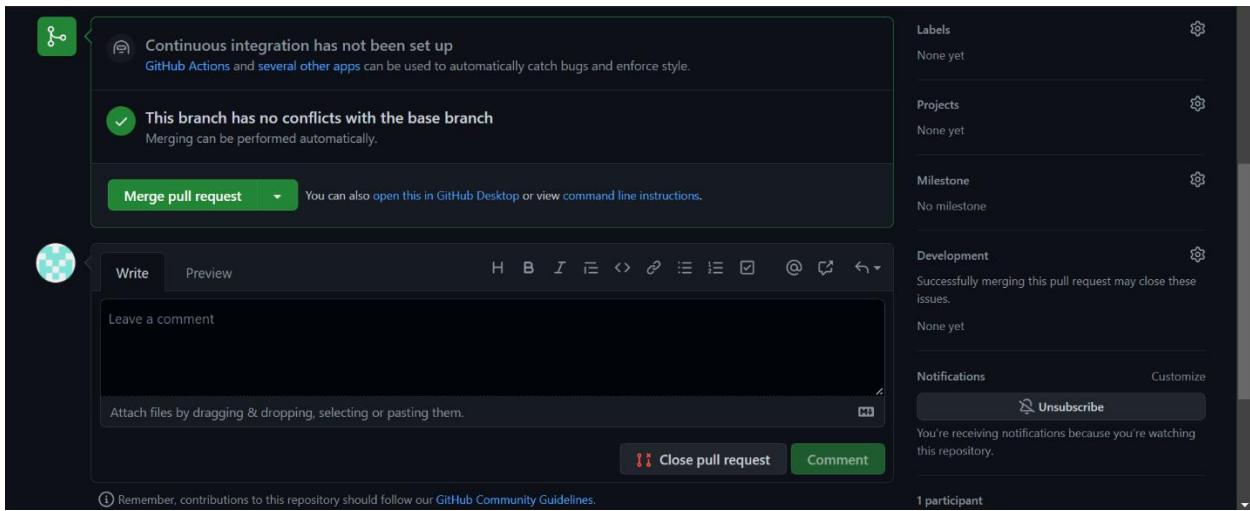
10. After opening a pull request the owner of the original repository will be sent the request if they want to merge or close the request.



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 H B I E <> P E M D @ ↻ Leave a comment

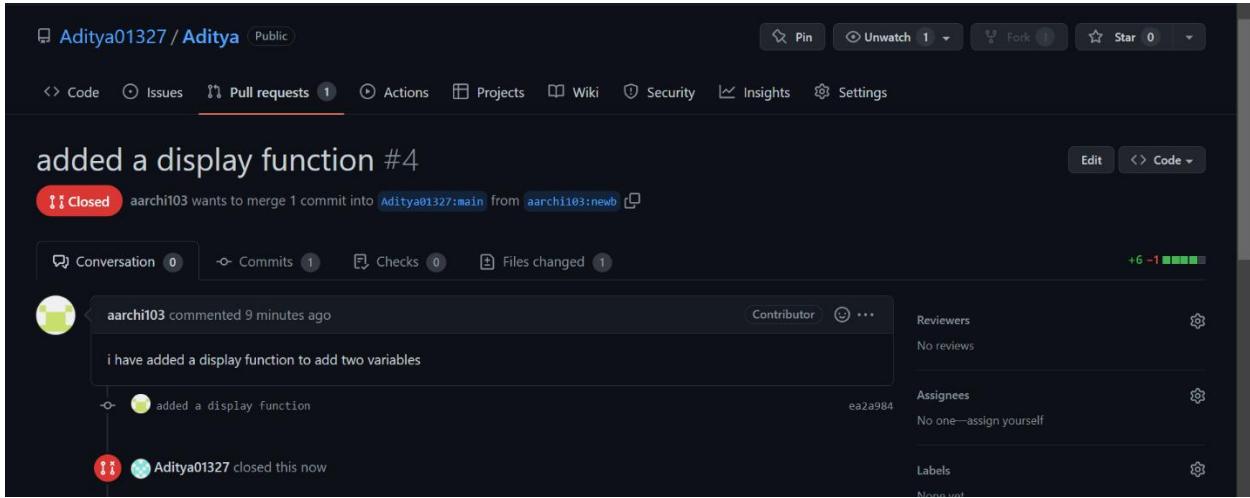
Attach files by dragging & dropping, selecting or pasting them.

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

Code Issues Pull requests 1 Actions Projects Wiki Security Insights Settings

added a display function #4

Closed aarchi103 wants to merge 1 commit into Aditya01327:main from aarchi103:newb

Conversation 0 Commits 1 Checks 0 Files changed 1

aarchi103 commented 9 minutes ago i have added a display function to add two variables

Contributor

ea2a984 +6 -1

aarchi103 added a display function

Aditya01327 closed this now

Reviewers No reviews Assignees No one—assign yourself Labels None yet

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/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")

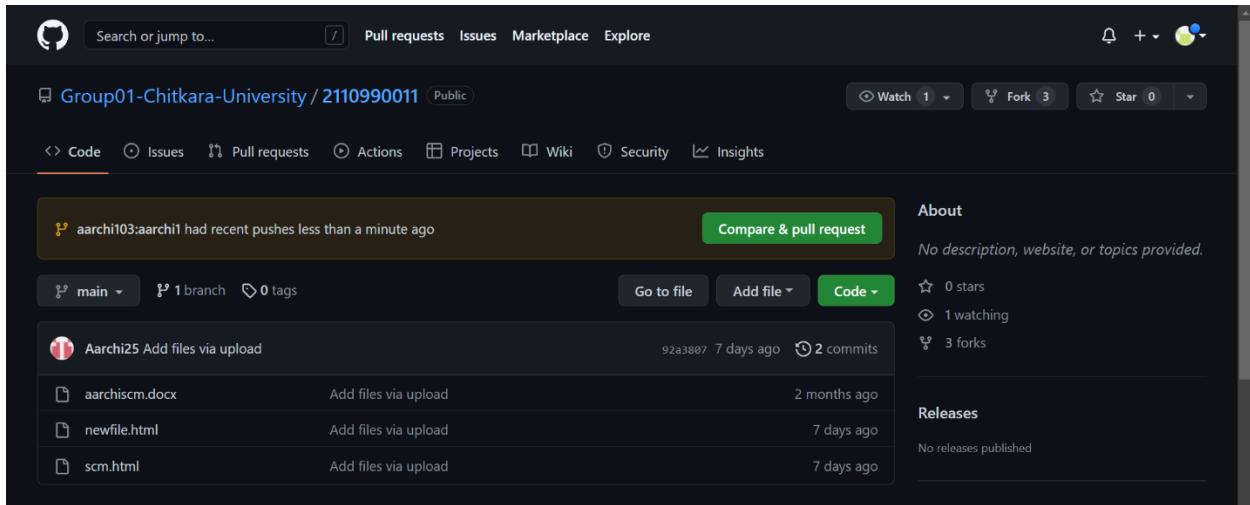
ASUS@Aarchi MINGW64 /d/2110990011 (aarchi)
$ git commit -a - "added styling"
[aarchi 12cbae4] added styling
1 file changed, 87 insertions(+), 77 deletions(-)
 rewrite newfile.html (64%)

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 ++++++-----+
-----
```

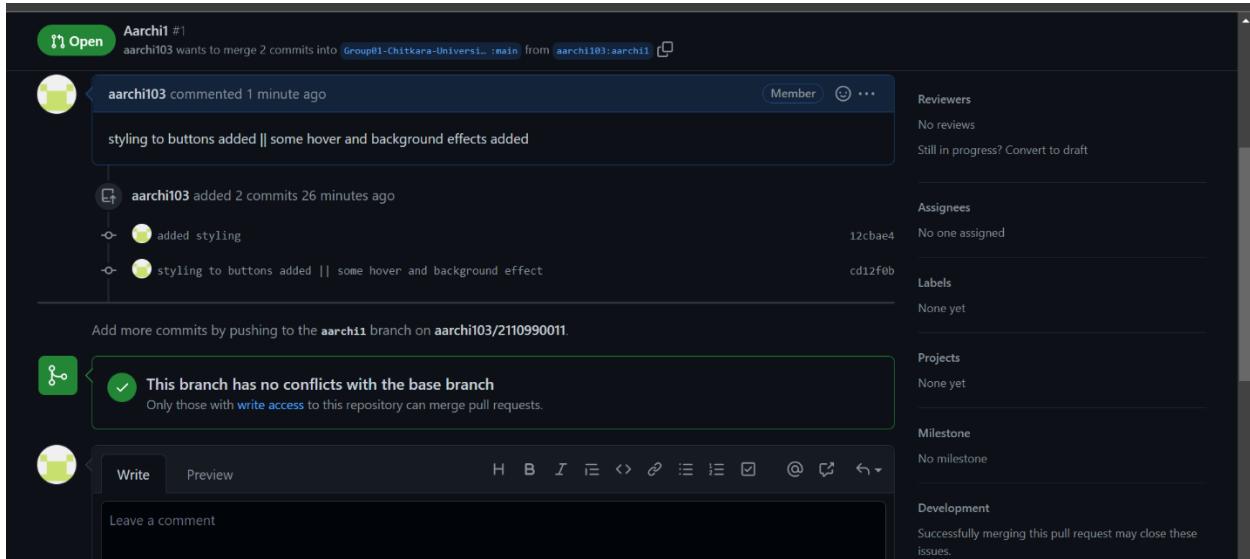
```
ASUS@Aarchi MINGW64 /d/2110990011 (main)
$ git push -u origin main
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 12 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 805 bytes | 402.00 KiB/s, done.
Total 3 (delta 1), reused 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.



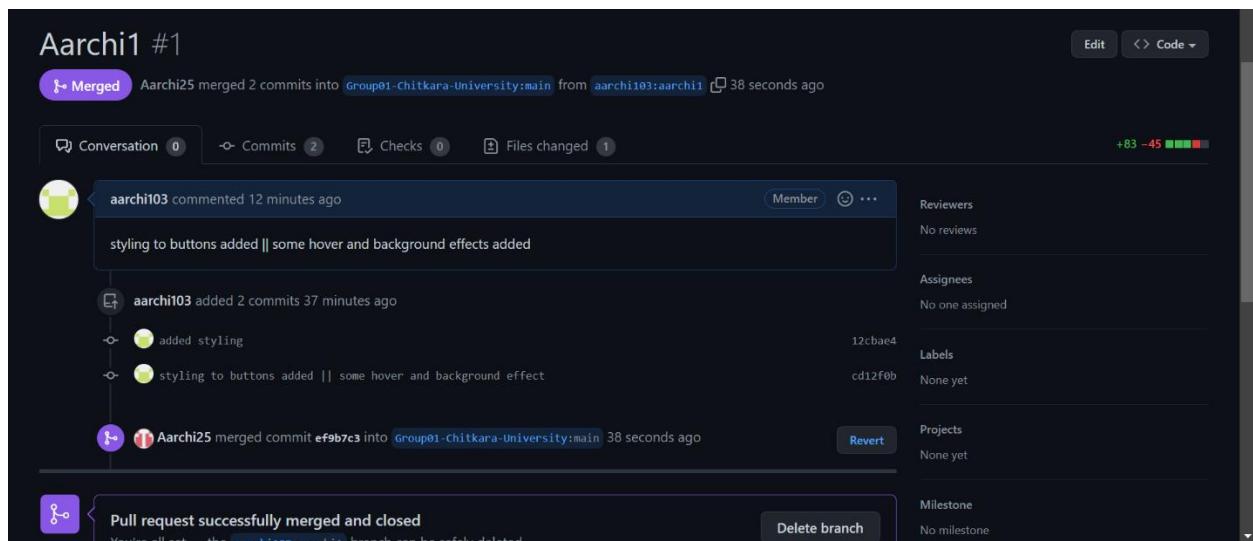
The screenshot shows a GitHub repository page. At the top, there's a search bar and navigation links for Pull requests, Issues, Marketplace, and Explore. Below the header, the repository name is "Group01-Chitkara-University / 2110990011" (Public). To the right are Watch (1), Fork (3), and Star (0) buttons. The main content area shows recent pushes from "aarchi103:aarchi1". A green button labeled "Compare & pull request" is visible. Below the pushes, there are buttons for "Go to file", "Add file", and "Code". A sidebar on the right contains sections for "About" (no description, website, or topics provided), "Releases" (no releases published), and statistics like 0 stars, 1 watching, and 3 forks.

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



The screenshot shows a GitHub pull request titled "Archit #1". It indicates that "aarchi103 wants to merge 2 commits into Group01-Chitkara-University :main from aarchi103:aarchi1". The pull request has one commit from "aarchi103 commented 1 minute ago": "styling to buttons added || some hover and background effects added". Below this, another commit from "aarchi103 added 2 commits 26 minutes ago" is shown, with two sub-commits: "added styling" and "styling to buttons added || some hover and background effect". A note at the bottom says "This branch has no conflicts with the base branch". The right side of the screen displays sections for Reviewers, Assignees, Labels, Projects, Milestone, and Development.

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.



Aarchi1 #1

Merged Aarchi25 merged 2 commits into Group01-Chitkara-University:main from aarchi103:aarchi1 38 seconds ago

Conversation 0 Commits 2 Checks 0 Files changed 1 +83 -45

aarchi103 commented 12 minutes ago styling to buttons added || some hover and background effects added

aarchi103 added 2 commits 37 minutes ago

- added styling 12cae4
- styling to buttons added || some hover and background effect cd12f0b

Aarchi25 merged commit ef9b7c3 into Group01-Chitkara-University:main 38 seconds ago Revert

Pull request successfully merged and closed Delete branch

Reviewers: No reviews

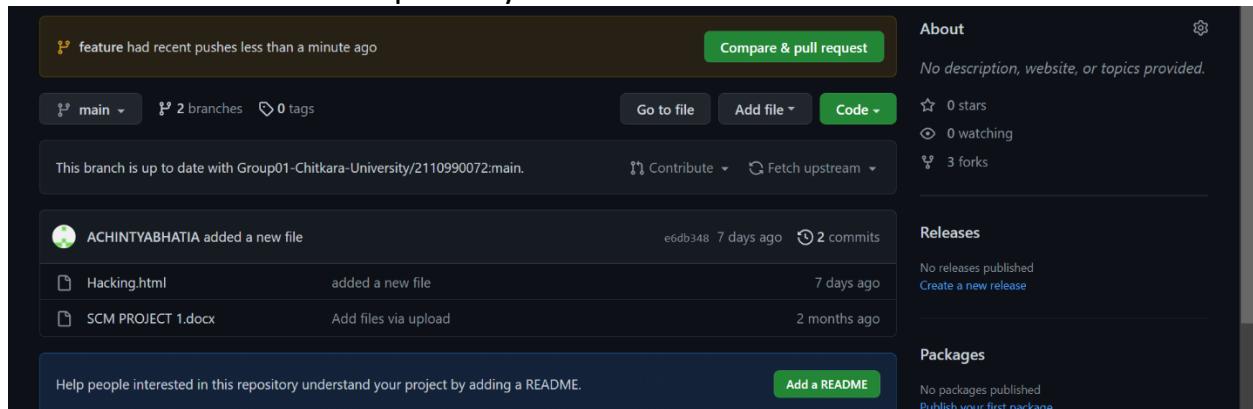
Assignees: No one assigned

Labels: None yet

Projects: None yet

Milestone: No milestone

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



feature had recent pushes less than a minute ago

Compare & pull request

main 2 branches 0 tags Go to file Add file Code

This branch is up to date with Group01-Chitkara-University/2110990072:main. Contribute Fetch upstream

ACHINTYABHATIA added a new file e6db348 7 days ago 2 commits

- Hacking.html added a new file 7 days ago
- SCM PROJECT 1.docx Add files via upload 2 months ago

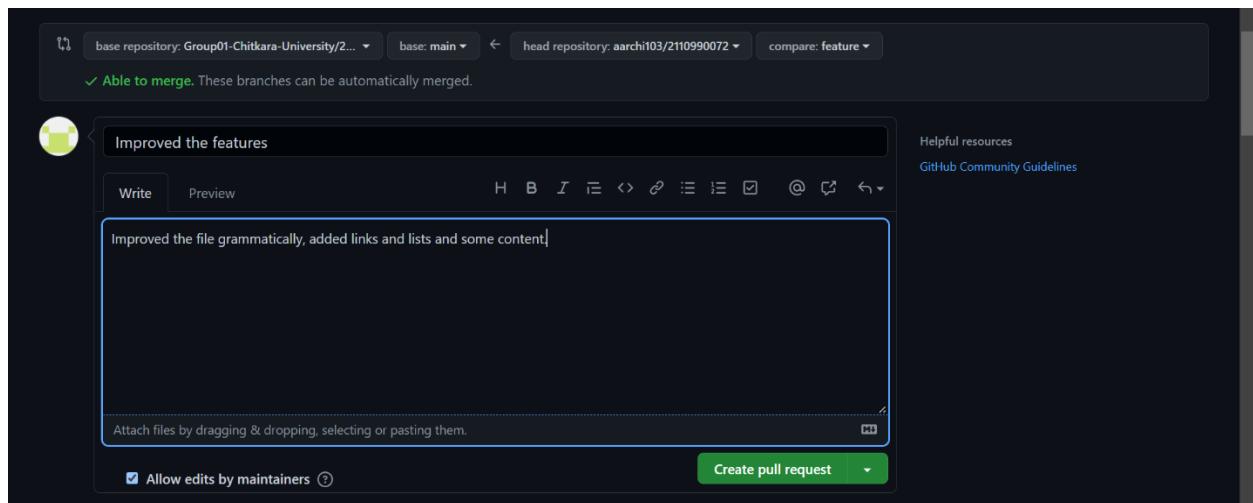
Add a README

About: No description, website, or topics provided.

0 stars 0 watching 3 forks

Releases: No releases published Create a new release

Packages: No packages published Publish your first package



base repository: Group01-Chitkara-University/2... base: main head repository: aarchi103/2110990072 compare: feature Able to merge. These branches can be automatically merged.

Improved the features

Write Preview

Improved the file grammatically, added links and lists and some content!

Attach files by dragging & dropping, selecting or pasting them.

Create pull request

Allow edits by maintainers

Improved the features #1

Merged ACHINTYABHATIA merged 1 commit into Group01-Chitkara-University:main from aarchi103:feature now

Conversation 0 Commits 1 Checks 0 Files changed 1 +16 -20

aarchi103 commented 2 minutes ago Member ...
Improved the file grammatically, added links and lists and some content.

Improved the features 69a51eb

ACHINTYABHATIA merged commit bedff03 into Group01-Chitkara-University:main now Revert

Pull request successfully merged and closed
You're all set — the aarchi103:feature branch can be safely deleted.
If you wish, you can also delete this fork of Group01-Chitkara-University/2110990072 in the settings.

Reviewers
No reviews

Assignees
No one assigned

Labels
None yet

Projects
None yet

Milestone

On 3rd team member's repository

aarchi103 / 2110990023 Public

forked from Group01-Chitkara-University/2110990023

Code Pull requests Actions Projects Wiki Security Insights Settings

branch1.1 had recent pushes less than a minute ago Compare & pull request

main 1 branch 0 tags Go to file Add file Code

This branch is up to date with Group01-Chitkara-University/2110990023:main. Contribute Fetch upstream

aroraashman Added a suggestion and contact form 36162f5 8 days ago 2 commits
2110990023-Aashman Arora-SCM.pdf SCM_1.1 2 months ago
Suggestions and contact .html Added a suggestion and contact form 8 days ago

About
No description, website, or topics provided.

0 stars 0 watching 3 forks

Releases
No releases published Create a new release

Packages
No packages published Publish your first package

Group01-Chitkara-University / 2110990023 Public

Edit Pins Watch 1 Fork 3 Star 0

Code Issues Pull requests 1 Actions Projects Wiki Security Insights

Added address input field #1

Open aarchi103 wants to merge 1 commit into Group01-Chitkara-University:main from aarchi103:branch1.1

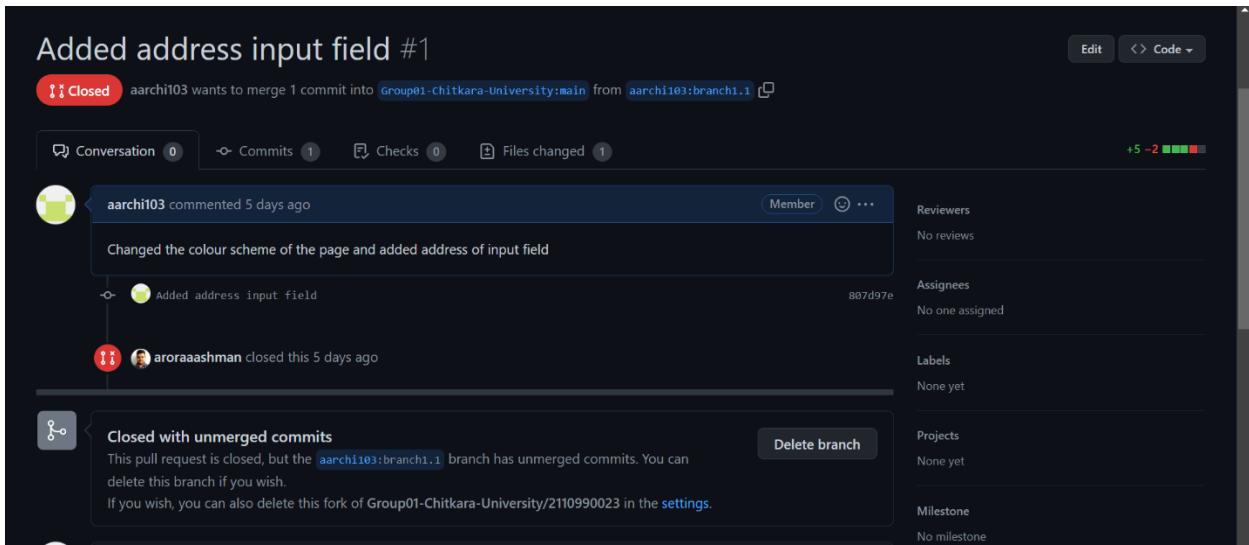
Conversation 0 Commits 1 Checks 0 Files changed 1 +5 -2

aarchi103 commented now Member ...
Changed the colour scheme of the page and added address of input field

Added address input field 807d97e

Reviewers
No reviews

Still in progress? Convert to draft

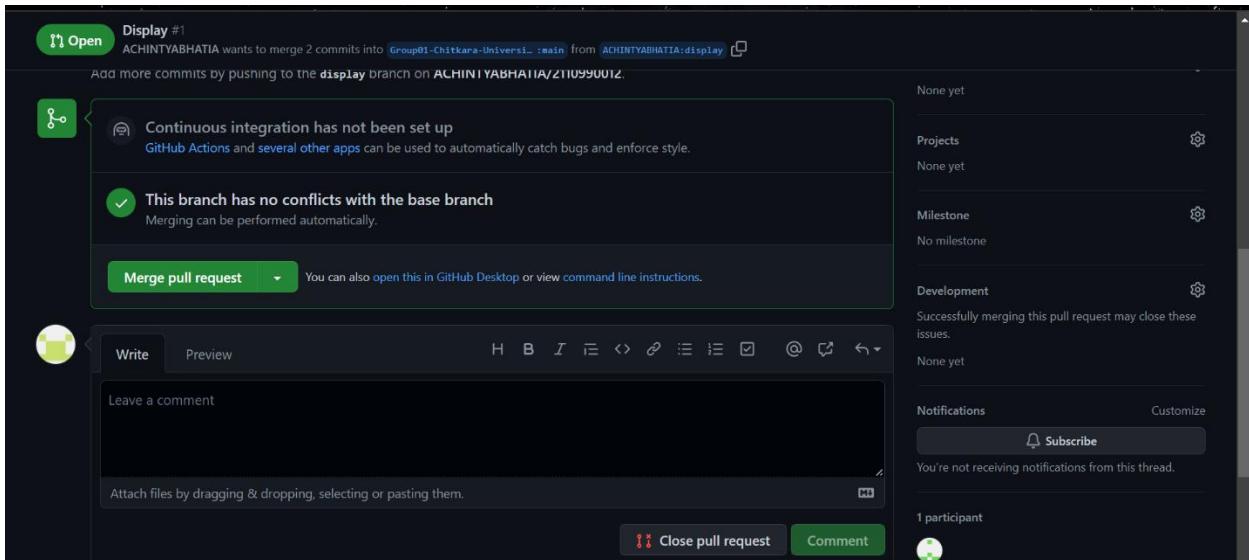


The screenshot shows a GitHub pull request page. At the top, a red button says "Closed" with a note: "aarchi103 wants to merge 1 commit into Group01-Chitkara-University:main from aarchi103:branch1.1". Below this, there are tabs for Conversation (0), Commits (1), Checks (0), and Files changed (1). A comment from "aarchi103" is visible, stating "Changed the colour scheme of the page and added address of input field" and "Added address input field". Another comment from "aroraashman" says "closed this 5 days ago". On the right side, there are sections for Reviewers (No reviews), Assignees (No one assigned), Labels (None yet), Projects (None yet), and Milestone (No milestone). A message at the bottom states: "Closed with unmerged commits. This pull request is closed, but the aarchi103:branch1.1 branch has unmerged commits. You can delete this branch if you wish. If you wish, you can also delete this fork of Group01-Chitkara-University/2110990023 in the settings." There is a "Delete branch" button.

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

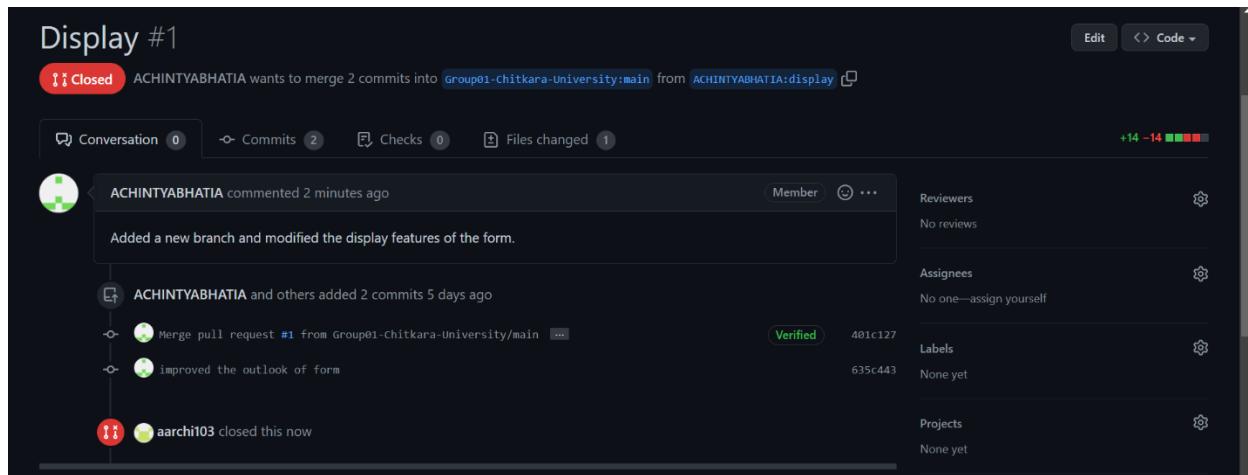
#CLOSING/MERGING PULL REQUESTS GENERATED BY TEAM MEMBERS

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.



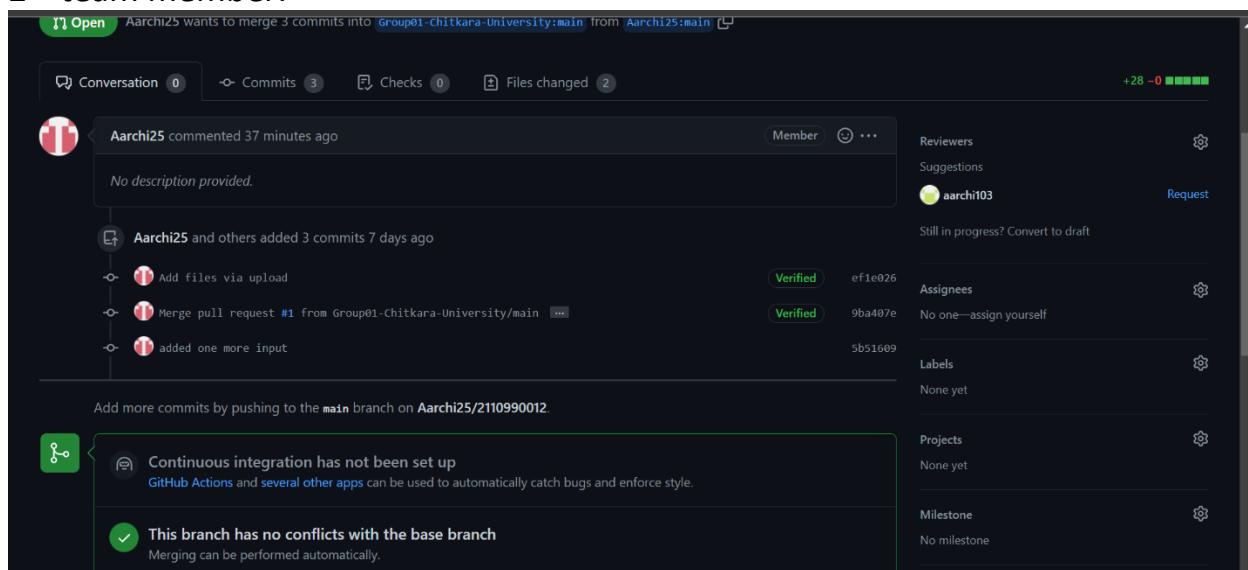
The screenshot shows a GitHub pull request page for a branch named "display". At the top, a green button says "Open" with a note: "ACHINTYABHATIA wants to merge 2 commits into Group01-Chitkara-University:main from ACHINTYABHATIA:display". Below this, there is a note: "Add more commits by pushing to the display branch on ACHINTYABHATIA/2110990012." The main content area shows two status messages: "Continuous integration has not been set up" (with a note about GitHub Actions) and "This branch has no conflicts with the base branch" (with a note about automatic merging). A green "Merge pull request" button is present. To the right, there are sections for None yet (under Development, Notifications, and Participants). Below the main content, there is a "Leave a comment" text area and a "Comment" button.

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.

added one more input field #2

Merged archi103 merged 3 commits into Group01-Chitkara-University:main from Archi25:main now

Conversation 0 Commits 3 Checks 0 Files changed 2 +28 -0

Aarchi25 commented 37 minutes ago
No description provided.

Aarchi25 and others added 3 commits 7 days ago

- Add files via upload
- Merge pull request #1 from Group01-Chitkara-University/main ...
- added one more input

aarchi103 merged commit ebe0ec into Group01-chitkara-University:main now

Revert

Reviewers: No reviews

Assignees: No one—assign yourself

Labels: None yet

Projects: None yet

Milestone: None

3rd team member:

Open Made changes on Webpage #3

aroraashman wants to merge 1 commit into Group01-Chitkara-University:main from aroraashman:Aashman now

No description provided.

Made changes on Webpage

Suggestions: aarchi103 Request
Still in progress? Convert to draft

Add more commits by pushing to the Aashman branch on aroraashman/2110990012

Merge pull request #3 from aroraashman/Aashman

Made changes on Webpage

Confirm merge Cancel

Assignees: No one—assign yourself

Labels: None yet

Projects: None yet

Made changes on Webpage #3

Merged archi103 merged 1 commit into Group01-Chitkara-University:main from aroraashman:Aashman now

Conversation 0 Commits 1 Checks 0 Files changed 1 +3 -3

aroraashman commented 2 minutes ago
No description provided.

Made changes on Webpage

0d13bd9

aarchi103 merged commit e285418 into Group01-chitkara-University:main now

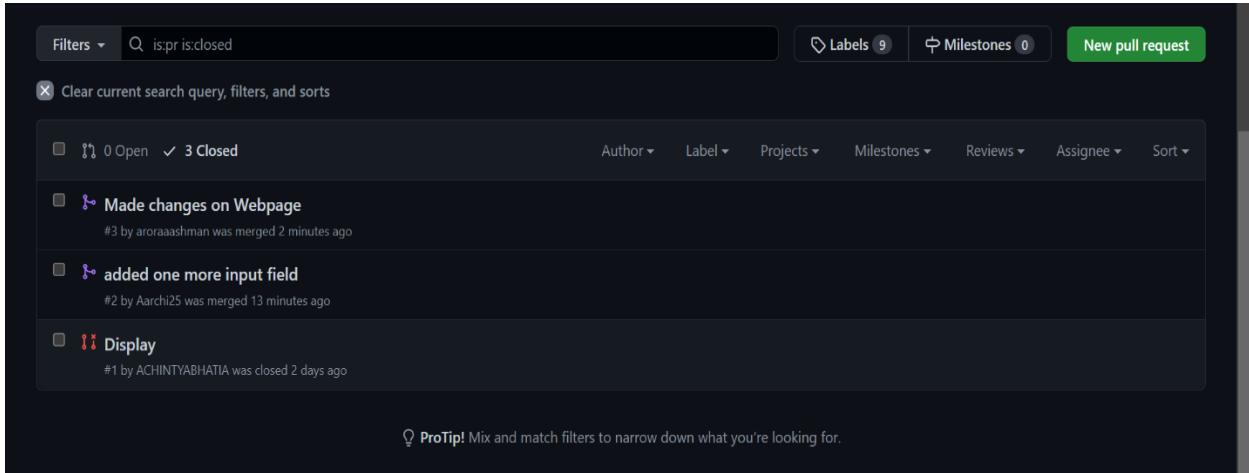
Revert

Reviewers: No reviews

Assignees: No one—assign yourself

Labels: None yet

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.



The screenshot shows a GitHub interface for a pull request history. At the top, there are filters: 'Filters' (dropdown), a search bar containing 'Q is:pr is:closed', and buttons for 'Labels 9', 'Milestones 0', and 'New pull request'. Below the filters is a link to 'Clear current search query, filters, and sorts'. The main area displays three closed pull requests:

- #3 by arorraashman was merged 2 minutes ago. Description: Made changes on Webpage.
- #2 by Aarchi25 was merged 13 minutes ago. Description: added one more input field.
- #1 by ACHINTYABHATIA was closed 2 days ago. Description: Display.

At the bottom, there is a 'ProTip!' message: 'Mix and match filters to narrow down what you're looking for.'

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 in 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.

[Code](#) [Issues](#) [Pull requests](#) [Actions](#) [Projects](#) [Wiki](#) [Security](#) [Insights](#) [Settings](#)

Pulse

May 24, 2022 – May 31, 2022 Period: 1 week ▾

Contributors

Community

Community Standards

Traffic

Commits

Code frequency

Dependency graph

Network

Forks

Overview

2 Active pull requests 0 Active issues

↳ 2 Merged pull requests	↳ 0 Open pull requests	⌚ 0 Closed issues	🕒 0 New issues
--------------------------	------------------------	-------------------	----------------

Excluding merges, **2 authors** have pushed **2 commits** to main and **2 commits** to all branches. On main, **1 file** has changed and there have been **9 additions** and **3 deletions**.



↳ 2 Pull requests merged by 2 people

The figure shows a timeline of commits for two GitHub users. The x-axis represents dates from April 8 to May 27. The y-axis lists repository names. For each repository, a horizontal line of dots indicates commit history, with arrows showing merges between branches.

- Group01-Chikara-University:** Commits on April 8, 9, 18, 20, 25, and 27. A merge arrow points from the April 20 branch to the May 27 branch.
- ACHINTYABHATIA:** Commits on April 19, 20, 21, 22, 23, 24, and 25. A merge arrow points from the April 21 branch to the April 24 branch. Another merge arrow points from the April 24 branch to the May 25 branch.

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.

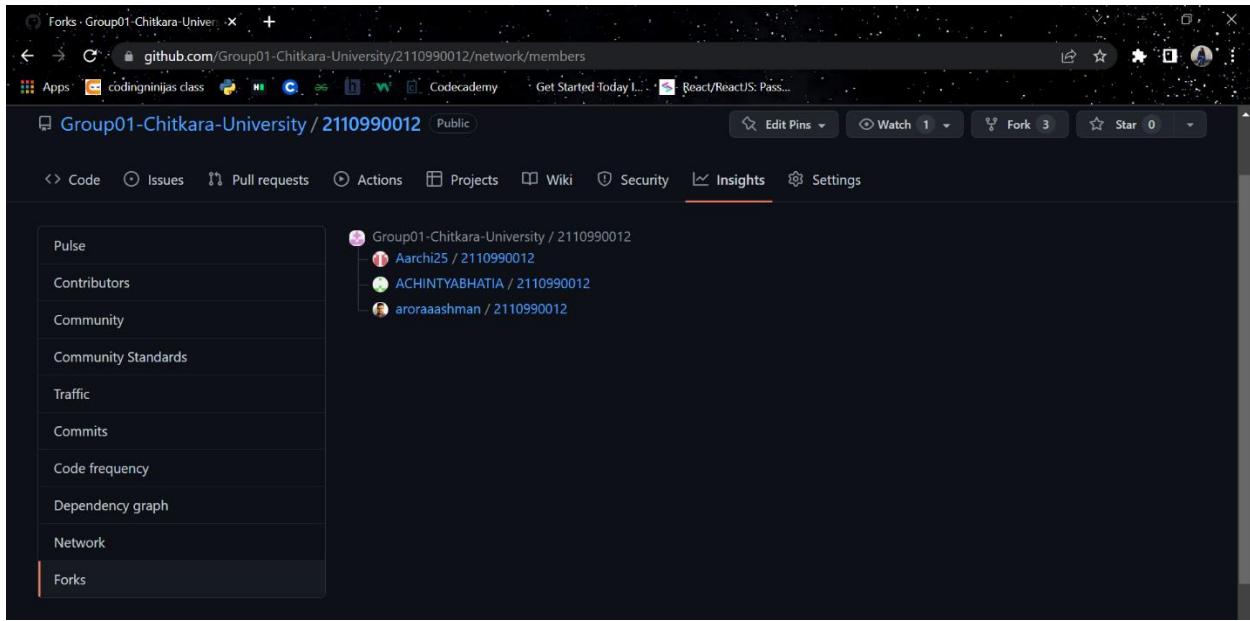
Community	Owners	Apr	May	
Community Standards		8 9 18	20 25 27	
Traffic				
Commits	Group01-Chitara-University			
Code frequency	ACHINTYABHATIA			
Dependency graph				
Network				
Ends				

The timeline shows commits from April 8 to May 27. A red box highlights a commit by user 'aroraashman' on May 26, which is described as 'Made changes on Webpage'. The commit is timestamped at 0d13bd9.

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.



The screenshot shows a GitHub repository page for "Group01-Chitkara-University / 2110990012". The "Forks" tab is selected in the navigation bar. On the left, there is a sidebar with various metrics: Pulse, Contributors, Community, Community Standards, Traffic, Commits, Code frequency, Dependency graph, Network, and Forks. The "Forks" option is highlighted with a red border. The main content area displays a list of forks, each with a user icon, name, and repository link:

- Group01-Chitkara-University / 2110990012
- Aarchi25 / 2110990012
- ACHINTYABHATIA / 2110990012
- arorraashman / 2110990012

TEAM MEMBERS

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