A Project report

on

“Task 2”

with

Source Code Management

(22CS003)

Submitted by:
Name: Bhuvesh Mittal
Roll No.: 2210991450

Submitted To:
Dr. Garima Chopra
Department of Computer Science & Engineering, Chitkara University Institute of Engineering and Technology Rajpura, Punjab

Team Member 1 Name: Bhavya Sood Roll No:2210991443

Team Member 2 Name: Bhoomika Roll No:2210991446

Team Member 3 Name: Bhuvesh Mittal Roll No:2210991450

Institute/School: - Name	**Chitkara University Institute of Engineering and Technology**
Department Name: -	**Department of Computer Science & Engineering**
Program Name: -	**Bachelor of Engineering (B.E.), Computer Science& Engineering**
Course Name: -	**Source Code Management** Session: **2022-23**
Course Code: -	**22CS003** Batch: **1st/2023**
Vertical Name: -	**First Year** Group No: 05/A

Faculty Name: Dr. Garima Chopra

Signature:

Date:

CHITKARA UNIVERSITY

Table of Content

Version Control with Git

▪ Git is a Version Control System that is widely used in the programming world.

▪ It is used for tracking changes in the source code during software development.

▪ Git is a distributed type of VCS which means it allows the user to store data both on a local system and on a server.

▪ It allows the user to make changes in a file without disturbing the original file.

▪ Git shows all the information about the changes made which makes it really easy to detect when an error was made and by whom.

▪ Git allows the user to make different versions of a project and work on them separately and all this is done without changing the original project.

▪ It helps the users to collaborate on a same project even without being in a same place.

▪ It allows the user the user to merge or delete different versions or branches of a project.

▪ Git also helps the user to easily recover a file in case it gets deleted or due to any malfunction.

PROBLEM STATEMENT

1. How to use git as a Version Control.
2. To take care of the Collaboration problem.
3. Using Git to solve basic arithmetic problems.

OBJECTIVE

The objective of this project is to associate programming with git because:

1. This is required because the collaboration makes the team work easy.
2. The code becomes manageable and we can build a clean repository.
3. Tracking and resolving of the errors is quite feasible in this process as we can exactly pinpoint when and where an error was made and by whom.
4. Moreover, we can make our projects locally as well as globally available.

RESOURCE REQUIREMENTS

Back End –

1. C Programming Language
2. Git Bash
3. GitHub

CONCEPTS AND COMMANDS

There are various concepts and commands used in this project.

▪ Firstly, we have used the used the concept of adding collaborators on GitHub.

▪ The concept of push and pull has been used to upload the data to GitHub from the local system and to merge the branches to the main branch respectively.

▪ The concept of cloning has been used to copy the repository made on GitHub to the local computer.

▪ Basic git commands like git add, git config, git commit, git status, git branch, git checkout, notepad, touch etc. have been used to perform their required functions.

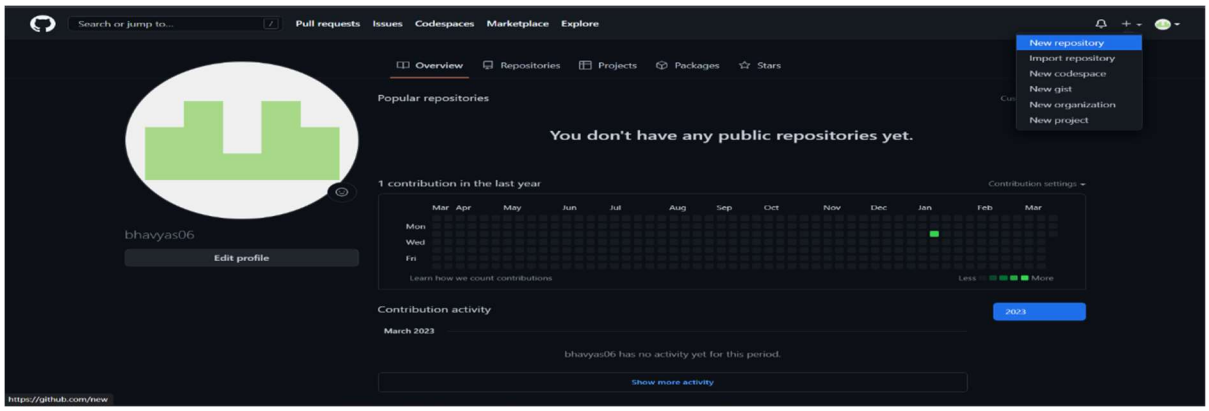
▪ And, we have used to concept of merging to merge the branches and the main branch.

WORKFLOW AND DISCUSSION

Create a distributed repository and add members of your project team

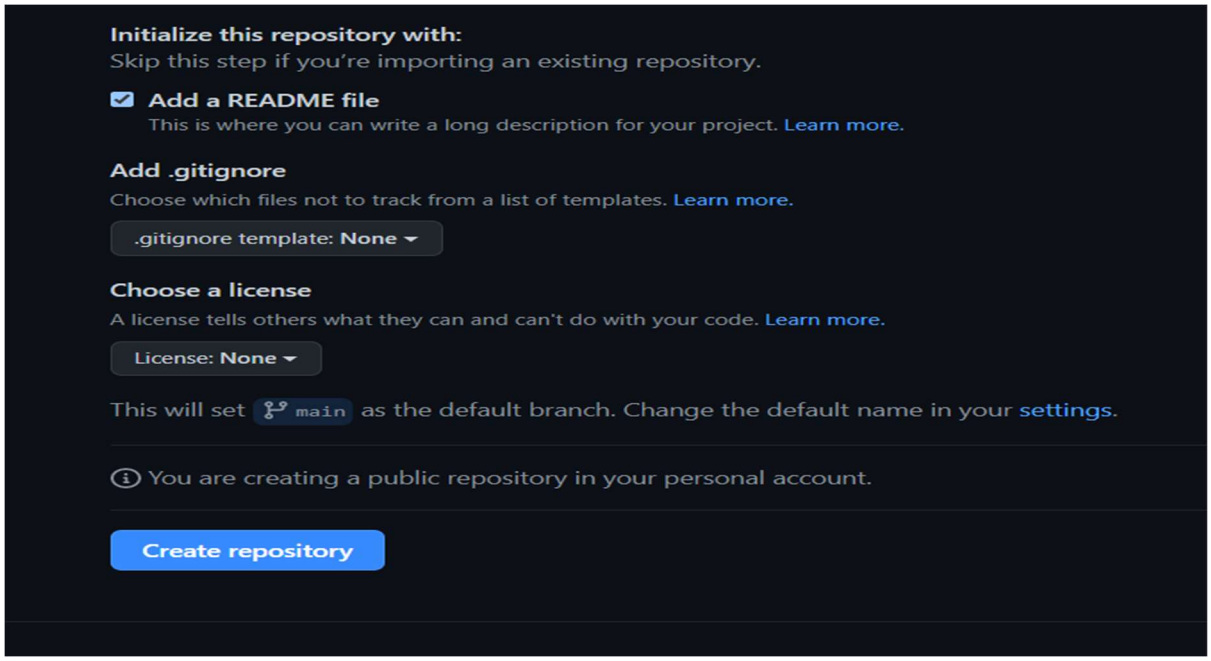
Step 1: Log in to your GitHub account.

Step 2: Create a new repository by clicking on the ‘New’ icon on the top right corner.

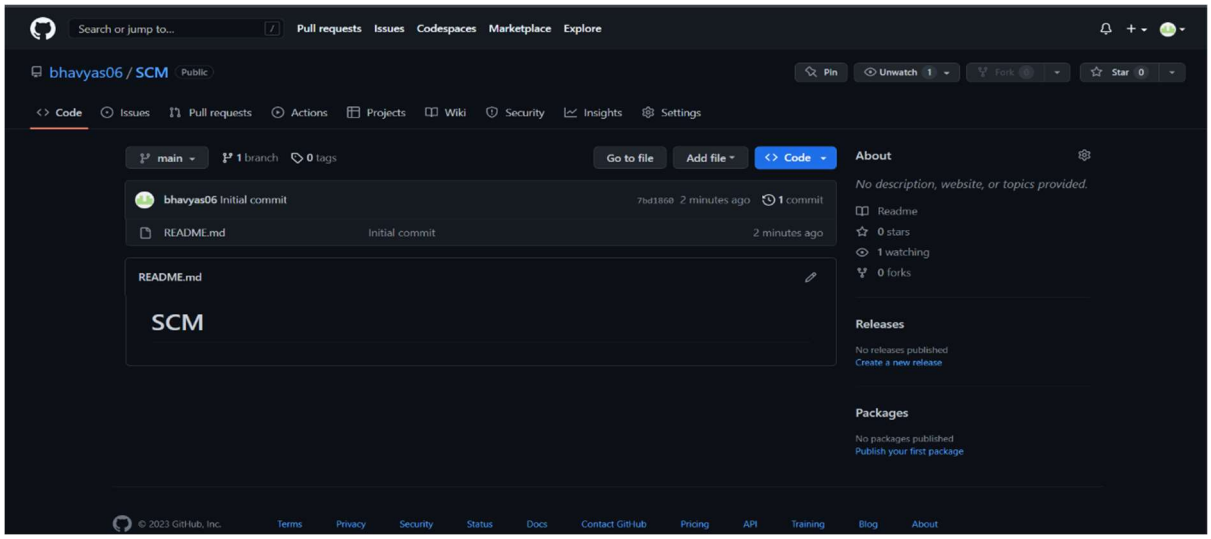


Step 2: Enter the repository name and add the description of the project.

Step 3: Select if you want the repository to be public or private and create a new repository.

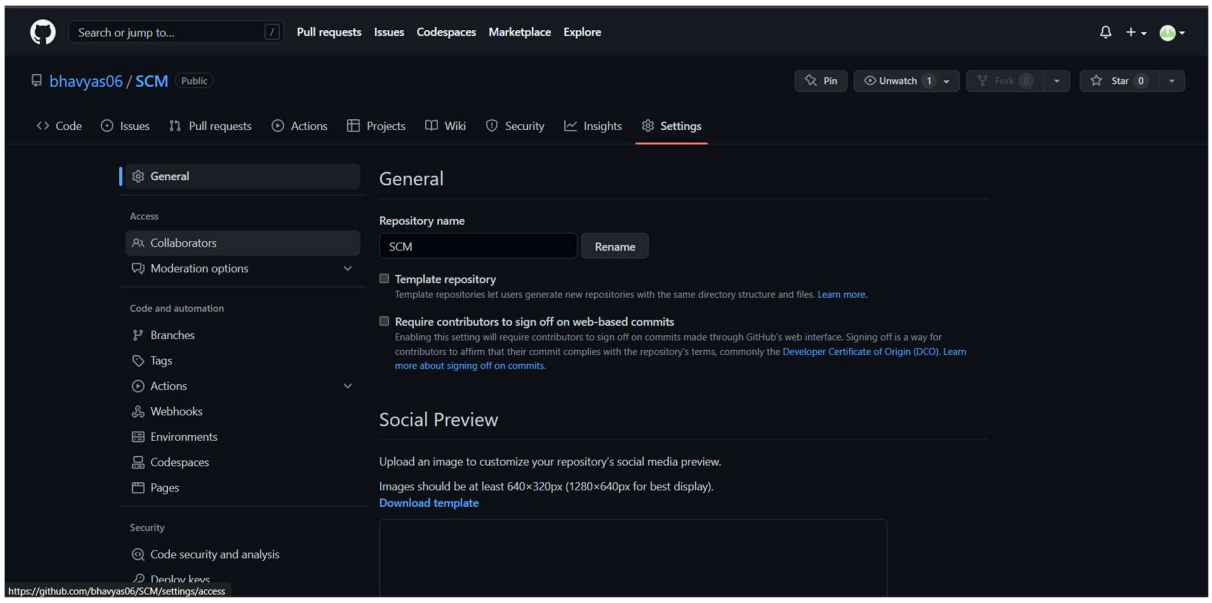


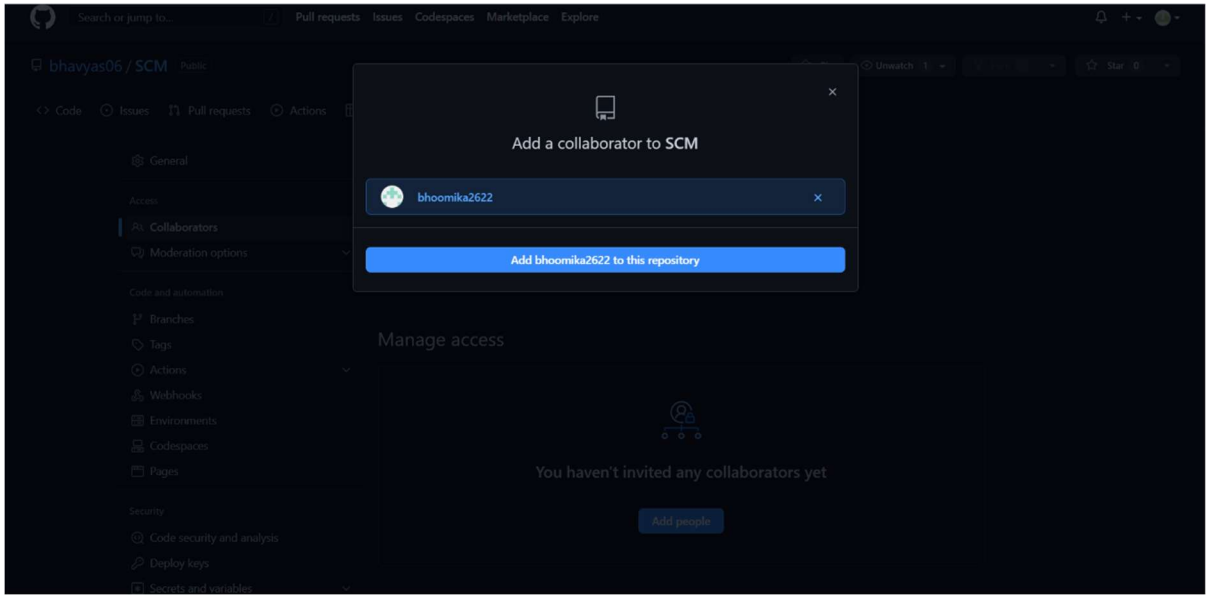
Step 4: Your new repository is available.

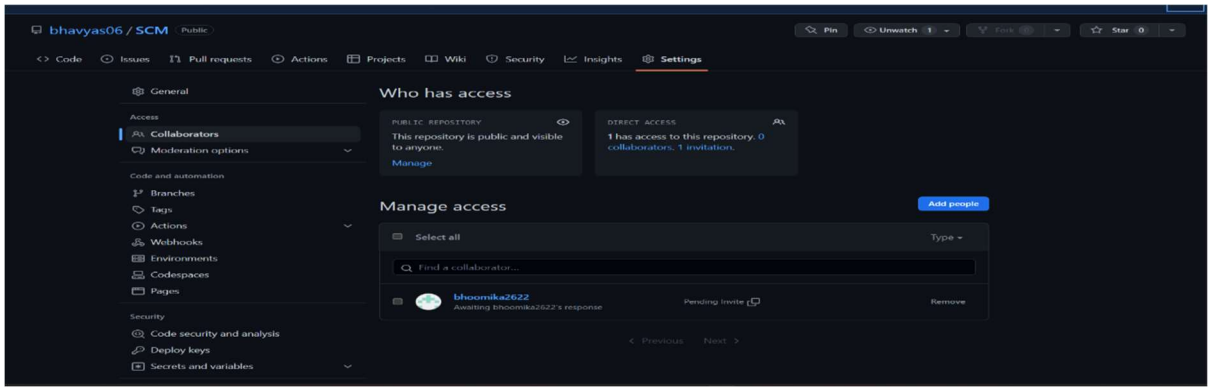


Step 5: Click on the ‘Settings’ tool and you will see a ‘Collaborators’ option on the left side.

Step 6: Click on that and now you can add your team members by writing their respective usernames.

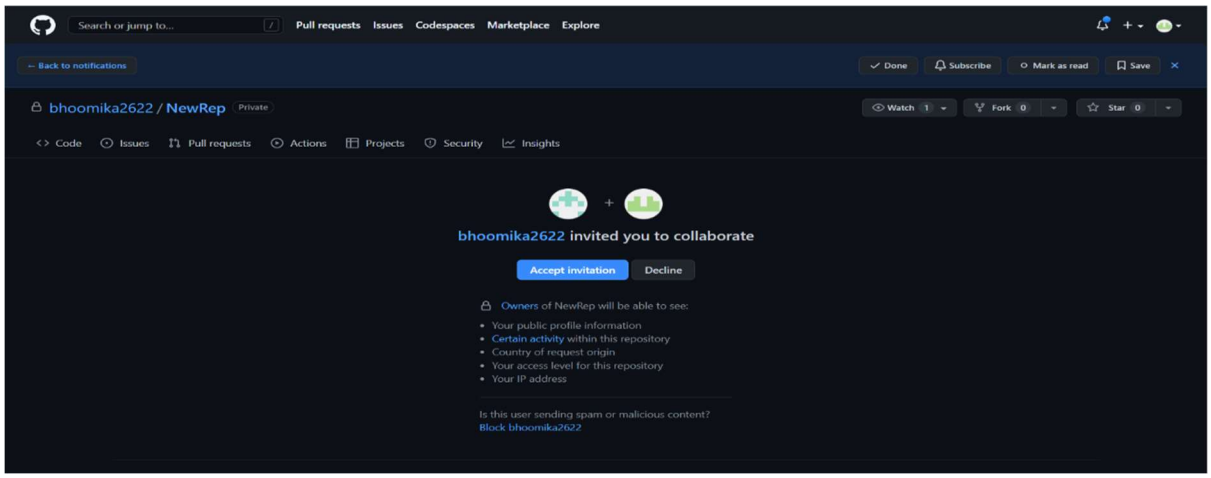






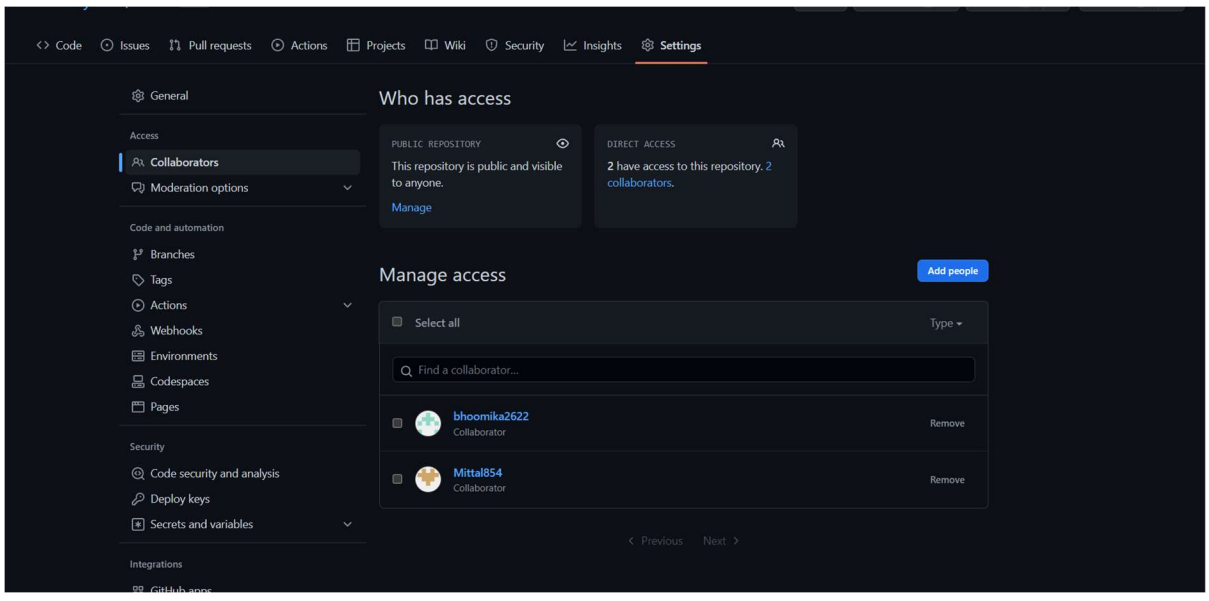
Step 7: Once you sent out the requests, they will receive the request in their ‘notifications’ tool at the top right corner and can either accept or decline the request.

Here’s how you will receive an invitation.



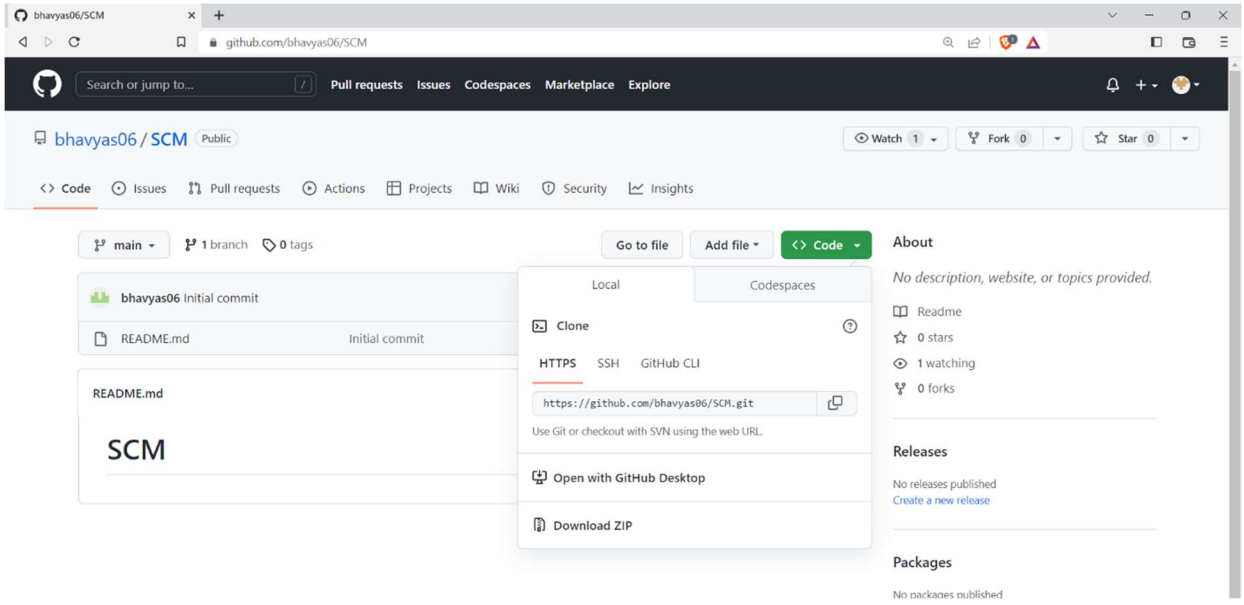
Step 8: You can also remove a collaborator.

Step 9: Once everyone is added, you can now start your project.



Open and Close a Pull Request

Step 1: Copy the 'url' of your created repository from the Code tool.



Step 2: Open Git Bash Here in your working directory.

Step 3: Use the *$ git clone 'url'* command to clone the repository into your local system.

```
bhuve@mittal MINGW64 /d/scm project
$ git clone https://github.com/bhavyas06/SCM.git
Cloning into 'SCM'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (3/3), done.
```

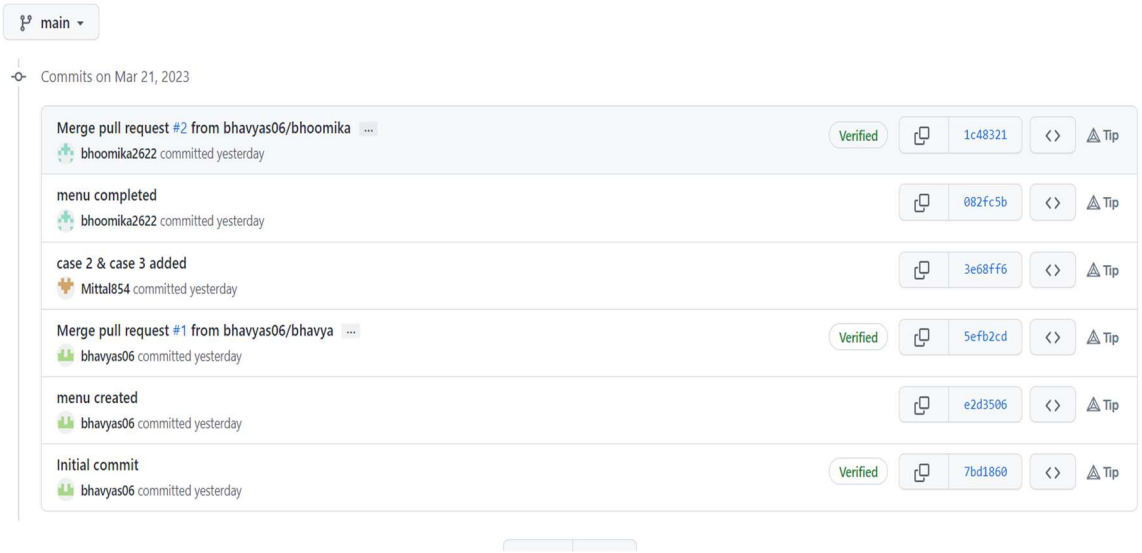
Step 4: Once the cloning is completed, Open the editor and make the changes, add them and commit them.

```
bhuve@mittal MINGW64 /d/scm project/SCM (main)
$ notepad project.c

bhuve@mittal MINGW64 /d/scm project/SCM (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   project.c

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

bhuve@mittal MINGW64 /d/scm project/SCM (main)
$ git add project.c

bhuve@mittal MINGW64 /d/scm project/SCM (main)
$ git commit -m "case 2 & case 3 added"
[main 3e68ff6] case 2 & case 3 added
 1 file changed, 10 insertions(+), 2 deletions(-)
```

Step 5: After all your changes are done, use the *$ git push origin 'branch name'* to push these changes into the GitHub repository.

```
bhuve@mittal MINGW64 /d/scm project/SCM (main)
$ git push origin main
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 16 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 403 bytes | 403.00 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/bhavyas06/SCM.git
   5efb2cd..3e68ff6  main -> main

bhuve@mittal MINGW64 /d/scm project/SCM (main)
$
```

Step 6: After this, our changes will be added in the main branch.



Publish and Print Network Graphs

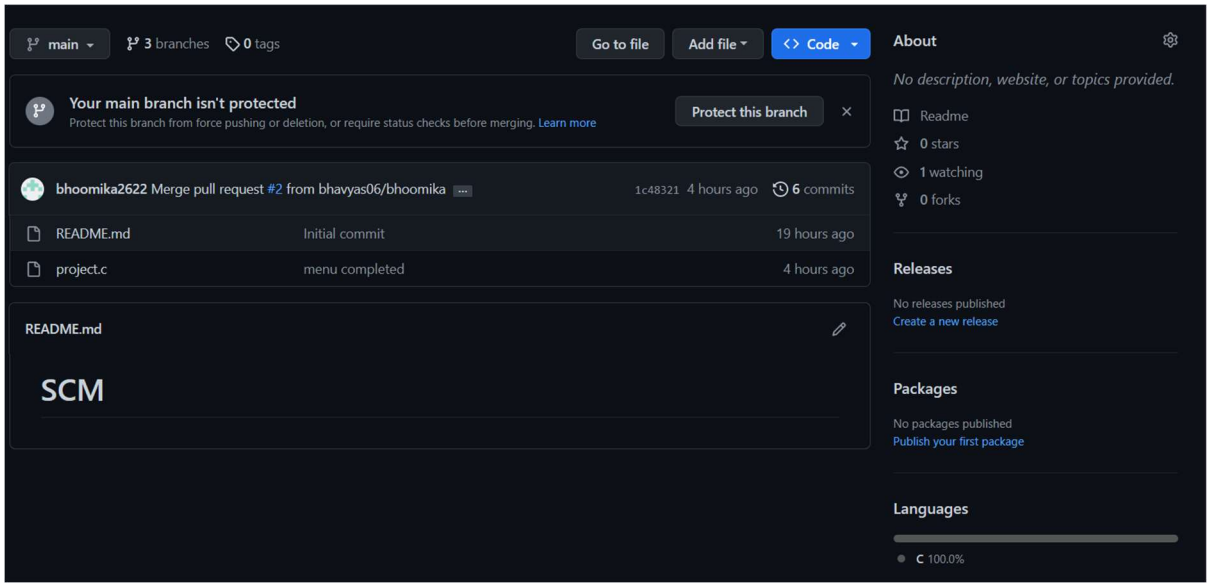
The network graph is one of the useful features for developers on GitHub. It is used to display the branch history of the entire repository network, including branches of the root repository and branches of forks that contain commits unique to the network.

A repository's graph gives you information on traffic, projects that depend on the repository, contributors and commits to the repository and a repository's forks and network.

If you maintain a repository, you can use this data to get a better understanding of who is using your repository and why they are using it.

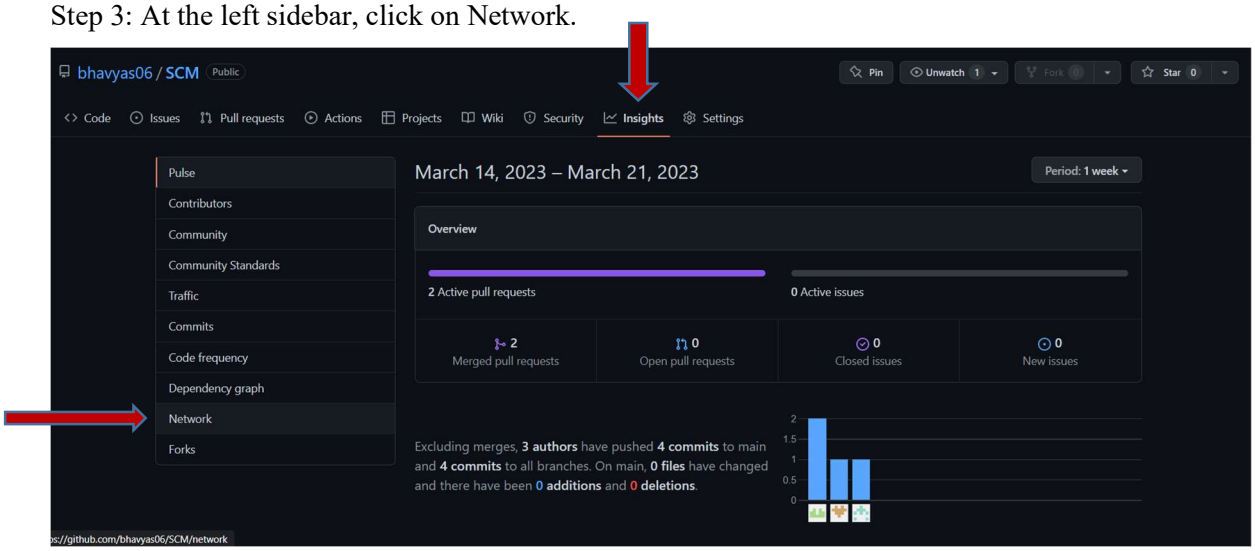
Here's how you can access the network graphs of your respective repository--

Step 1: On GitHub.com, navigate to the main page of the repository.

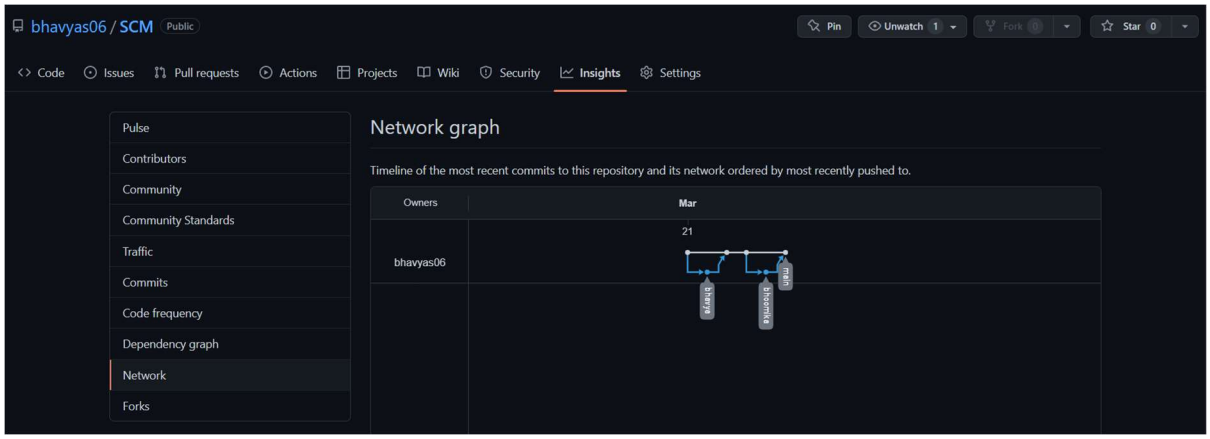


Step 2: Under your repository name, click on insights.

Step 3: At the left sidebar, click on Network.



Step 4: You will get the network graph of your repository which displays the branch history of the entire repository network, including branches of the root repository and branches of the forks that contain commits unique to the network.

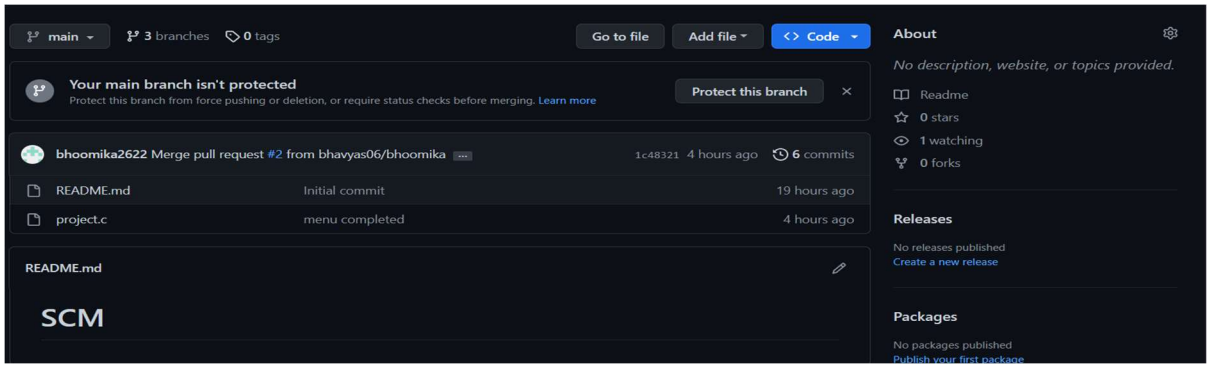


Listing the Forks of a Repository—

Forks are listed alphabetically by the username of the person who forked the repository.

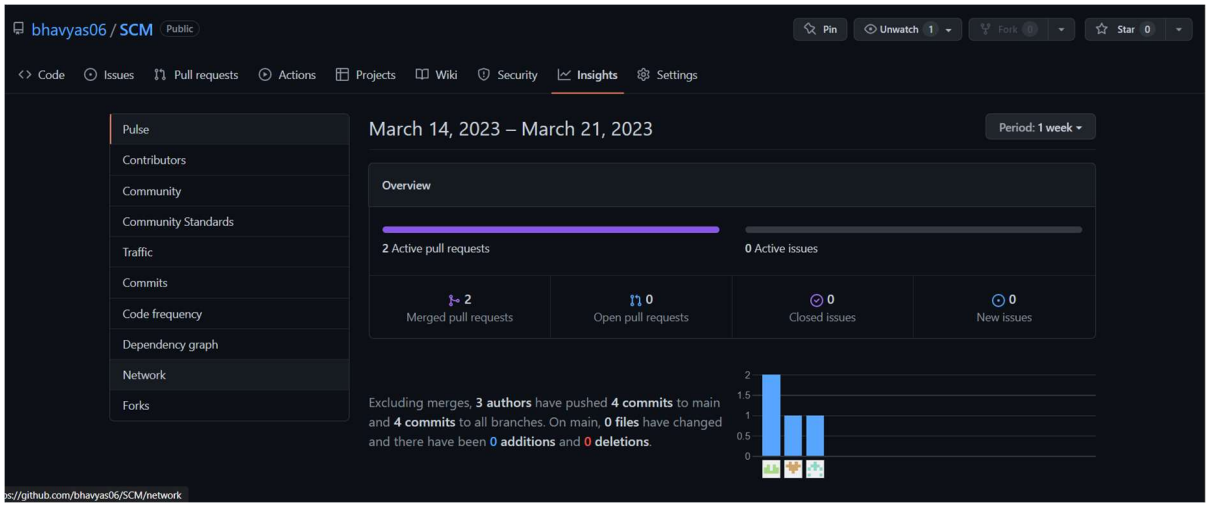
Clicking the number of forks shows you the full network. From there you can click ‘members’ to see who forked the repo.

Step 1: On GitHub.com, navigate the main page of your repo.

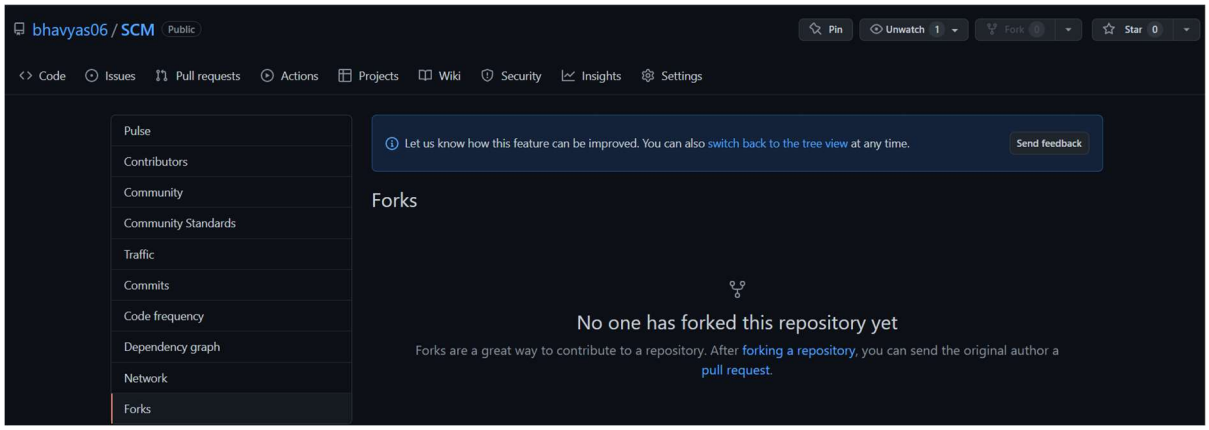


Step 2: Under your repo name, click on Insights.

Step 3: At the left sidebar, click on Forks.



Step 4: Here you can see all the forks.



Viewing the Dependencies of a Repository—

You can use the dependency graph to explore the code of your repository depends on:

Almost all software relies on code developed and maintained by other developers, often known as a supply chain. For example – utilities, libraries and frameworks. These dependencies are an integral part of your code and any bugs or vulnerabilities in them may affect your code. It's important to review and maintain these dependencies.

REFERENCE

To complete this project, we took the help of Git Documentation.

Also, the book, Pro Git written by Scott Chacon was used.