**­­A Project Report on**

# “Task 2”

**with**

# Source Code Management

(22CS181)

**Submitted By:**

**Name :** Mann Sharma

**Roll No :** 2210990555

# Submitted to:

**Faculty Name:** Dr.Harveen Kaur

(Department of Computer Science & Engineering, Chitkara University Institute of Engineering and Technology Rajpura ,Punjab)

**Team Member 1 Name :** Manmeet Kaur Roll No. 2210990554

**Team Member 2 Name :** Mann Sharma Roll No. 2210990555

**Team Member 3 Name :** Manav Singh Roll No. 2210990553



**Institute 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:-** 2023-24

**Course Code : -** 22CS003

**Batch :-** 2023

**Vertical Name: -** First Year **Group No: -**  G25-(2)

**Faculty Name: Prof./Dr./Mr./Ms :-** Harveen Kaur

**Signature:-**

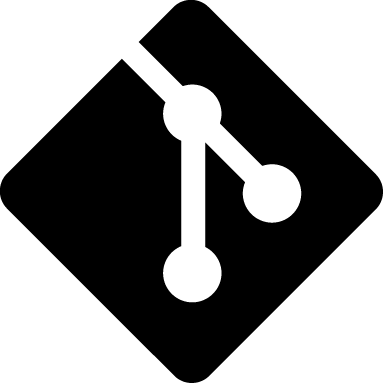
**Date:-**

**Table of Content**

|  |  |  |
| --- | --- | --- |
| **S.NO** | **Title** | **PageNo.** |
| 1. | Version control with Git |  |
| 2. | Problem statement |  |
| 3. | Objective |  |
| 4. | Resources Requirements – Front End/  Back end |  |
| 5. | Concepts and commands |  |
| 6. | Workflow and Discussion |  |
| 7. | Reference |  |

## Git and GitHub Introduction

## What is Git ?



Git is a popular version control system. It was created by Linus Torvalds in 2005, and has been maintained by Junio Hamano since then. It is used for:

* Tracking code changes
* Tracking who made changes
* Coding collaboration

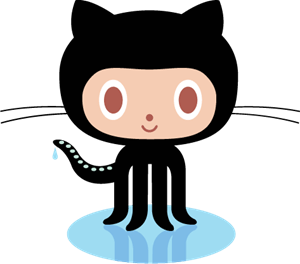
### What does Git do ?

* Manage projects with **Repositories**
* **Clone** a project to work on a local copy
* Control and track changes with **Staging** and **Committing**
* **Branch** and **Merge** to allow for work on different parts and versions of a project
* **Pull** the latest version of the project to a local copy
* **Push** local updates to the main project

### Why Git ?

* Over **70%** of developers use Git!
* Developers can work together from anywhere in the world.
* Developers can see the full history of the project.
* Developers can revert to earlier versions of a project.

**What is GitHub?**



* **Git is not the same as GitHub.**
* GitHub makes tools that use Git.
* GitHub is the largest host of source code in the world, and has been owned by Microsoft since 2018.
* In this , we will focus on using Git with GitHub.

**What is Repository ?**

A repository is a directory or storage space where your projects can live. Sometimes GitHub users shorten this to “repo.” It can be local to a folder on your computer, or it can be a storage space on GitHub or another online host. You can keep code files, text files, image files, you name it, inside a repository.

**What is Version Control System (VCS) ?**

A version control system is a tool that helps you manage “versions” of your code or changes to your code while working with a team over remote distances .Version control keeps track of every modification in a special kind of database that is accessible to the version control software. Version control software (VCS) helps you revert back to an older version just in case a bug or issue is introduced to the system or fixing a mistake without disrupting the work of other team members.

**Types of VCS**

1. Local Version Control System

2. Centralized Version Control System

3. Distributed Version Control System

**Local Version Control System**

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

**Centralized Version Control System**

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

**Distributed Version Control System**

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

**Problem Statement**

Web application for Hotel reservation. We will take the selection criteria from user and display the hotels list for user basing on the criteria. User can book the room if there is availability of the rooms in that particular hotel. There are three different types of user roles for the application they are administrator , hotel agent and normal user. Following are the actions provided for each user.

**Solution**

Normal user :

Can register for the site

Search the hotel details basing on the criteria.

Book the hotel room

Can modify the self details.

Hotel Agent:

Can register for the site

Can add/update the details of the hotel.

Adminstrator:

Will approve the new hotel details added to the application

Can delete the user/hotel details.

Minimal Features :

Adding Hotel information such as hotel name, location, number of rooms , facilities etc. to the database

Listing the hotels based on different criteria selected by the user.

User able to select a hotel and book a room.

Booking permitted only if there are rooms available

Displaying the reservation status.

Registration of users.

Update user details.

Modify hotel details.

Approval of the details entered by the hotel agent.

Delete user/hotel details by admin

Additional Features:

Taking Feedback from user.

Rating the hotels based on the feedback.

Goals:

To complete the hotel booking flow, registration of the users, adding details of the hotels.

Deliverables:

Minimal Features and additional features provided if time permits.

Out of scope:

We are not embedding payment gateway in the current application we will try if time permits after completion of additional features.

**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

.

4. Moreover, we can make our locally available projects, globally available.

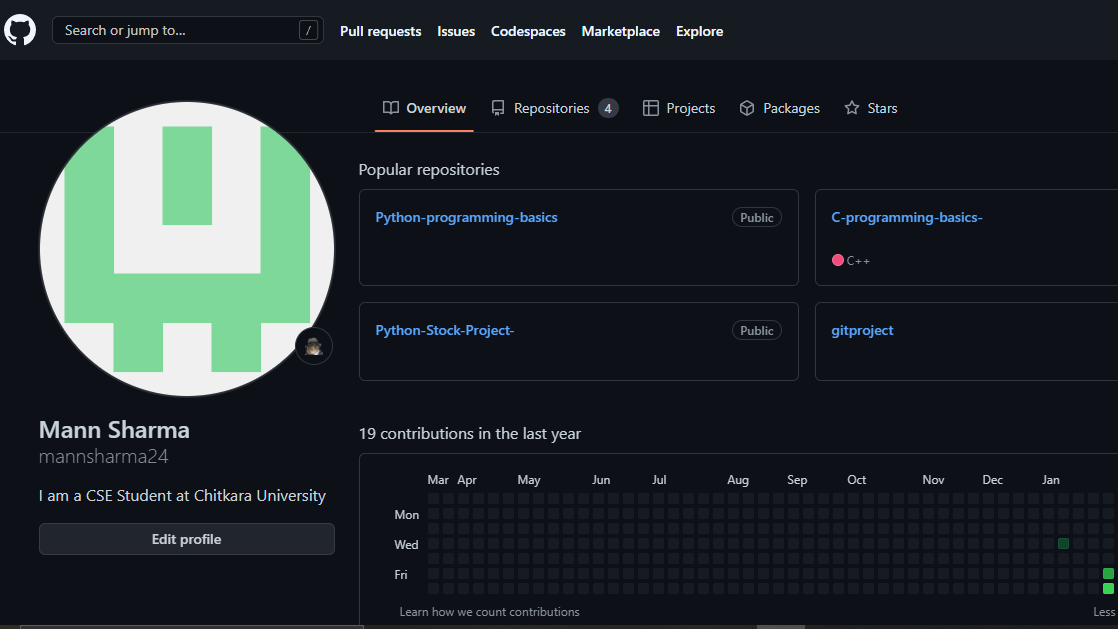
**Experiment No. 01**

**Aim :**

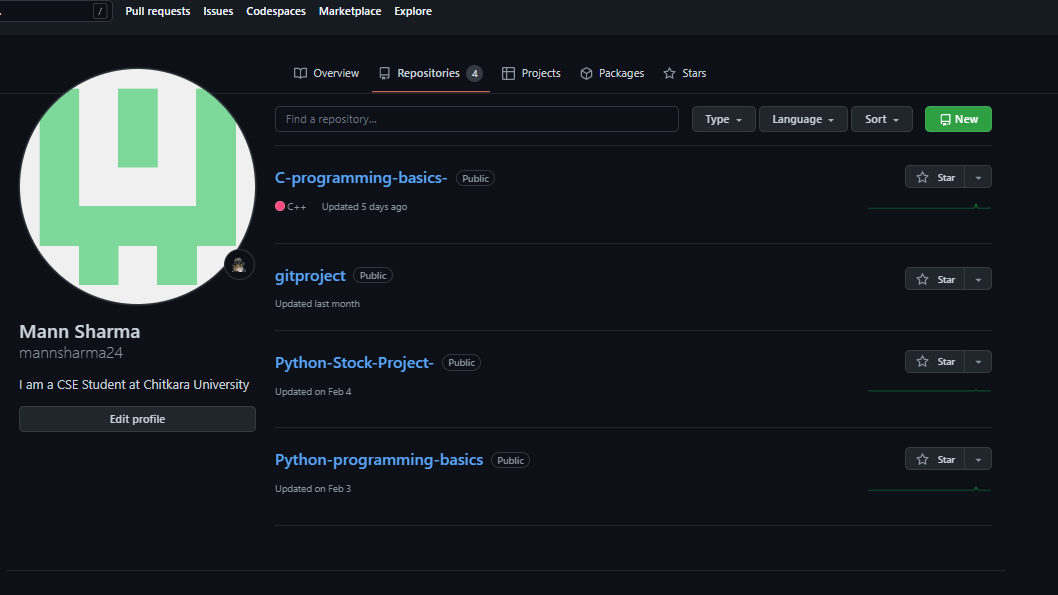
Create a distributed Repository and add members in project team.

**Procedure :**

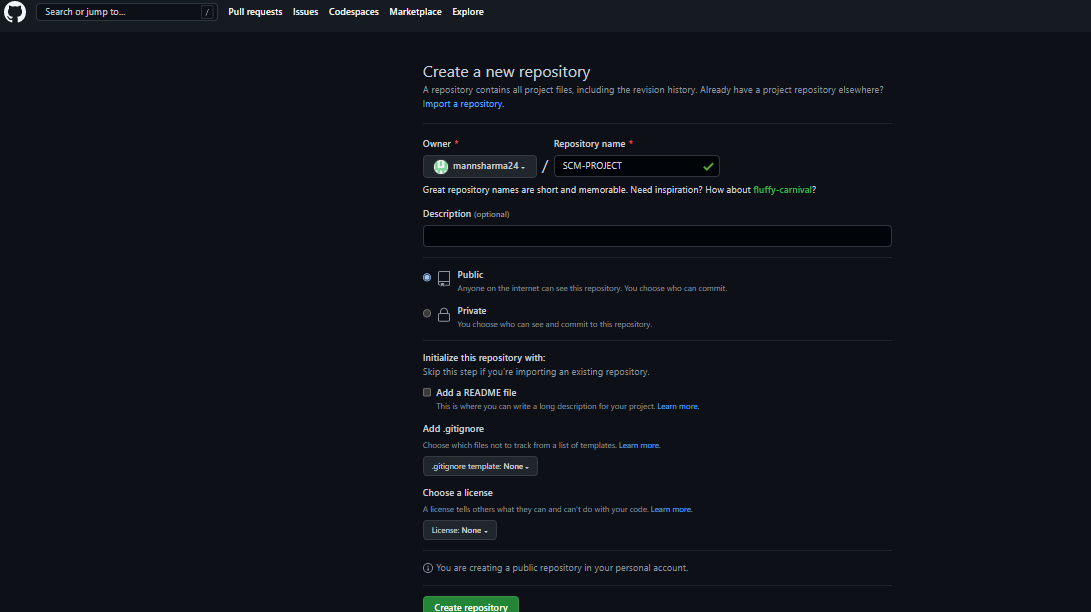
* 1. Login to your GitHub account and you will land on the homepage as shown below. Click on Repositories option in the menu bar.



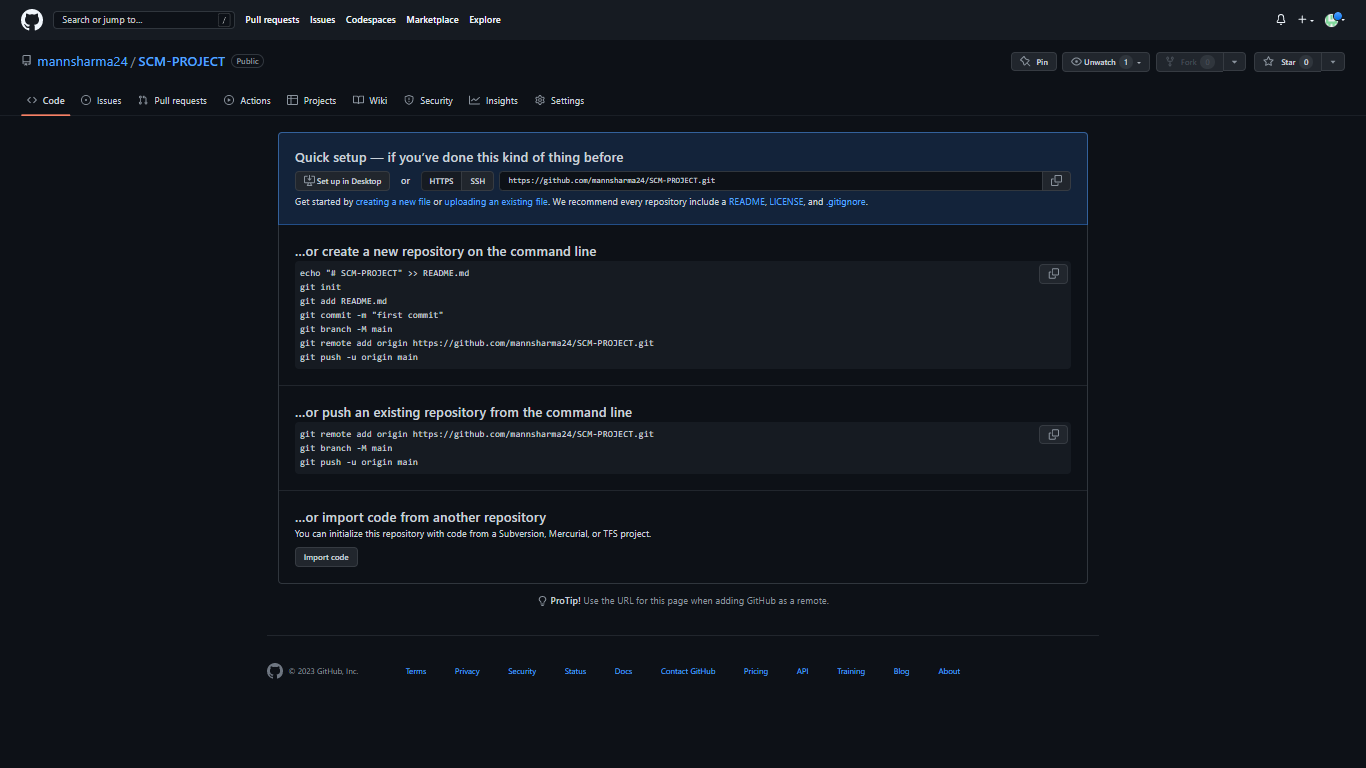
* 1. Click on the ‘New’ button in the top right corner.



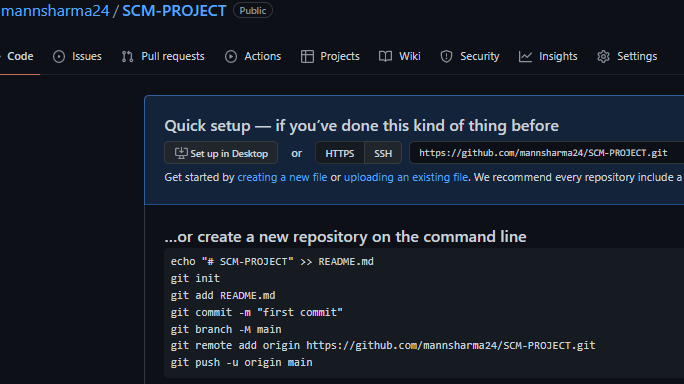
* 1. Enter the Repository name and add the description of the repository.
  2. Select if you want the repository to be public or private.



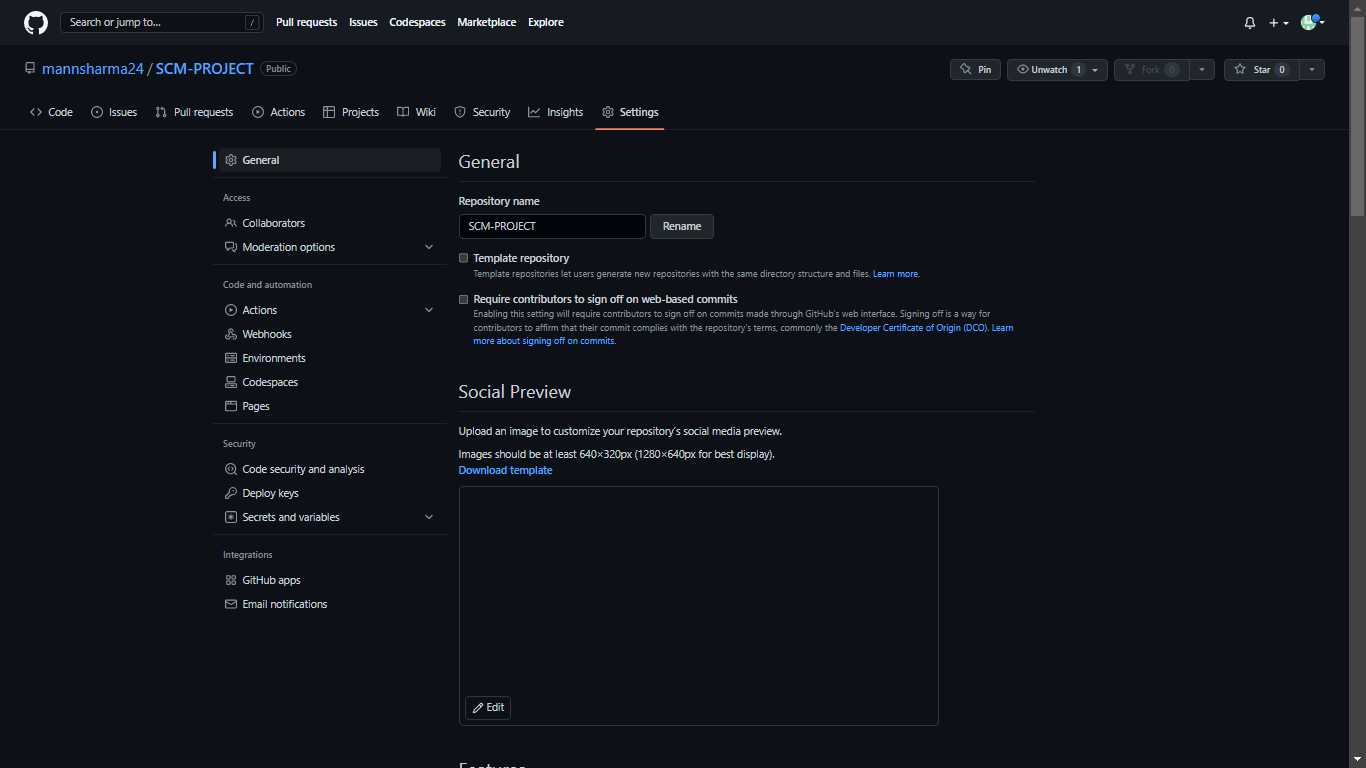
* 1. If you want to import code from an existing repository select the import code option.



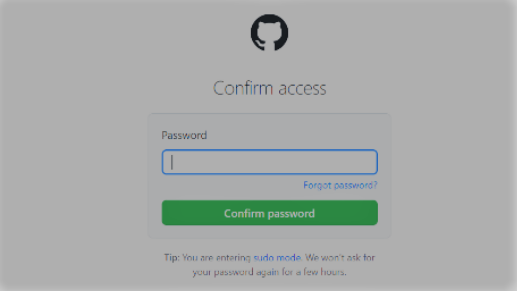
* 1. Now, you have created your repository successfully.
  2. To add members to your repository open your repository and select settings option in the navigation bar.



* 1. Click on Collaborators option under the access tab.

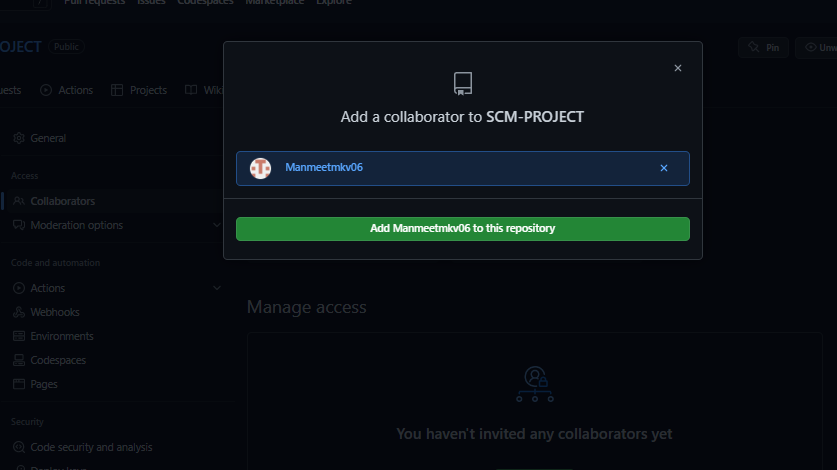


* 1. After clicking on collaborators GitHub asks you to enter your password to confirm the access to the repository.



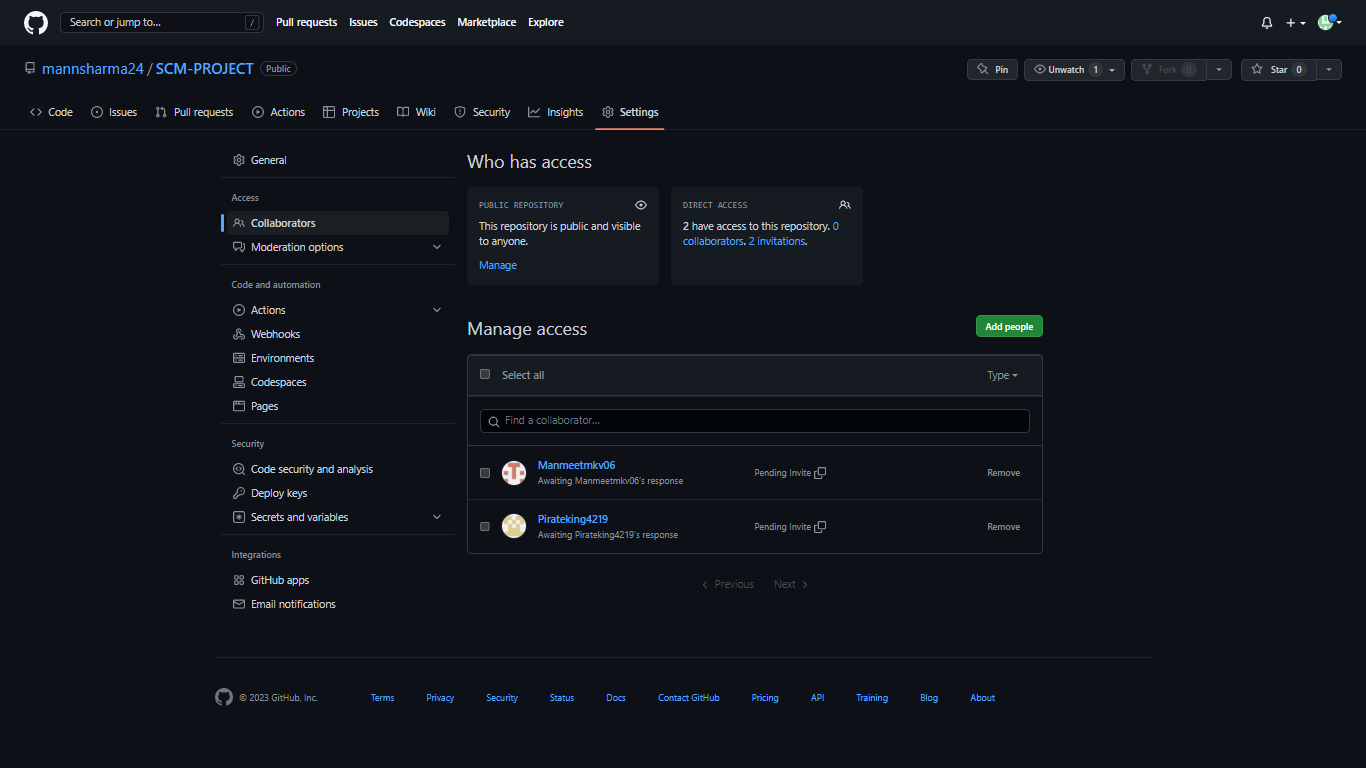
10) After entering the password you can manage access and add/remove team members to your project.

11) To add members click on the add people option and search the id of your respective team member.

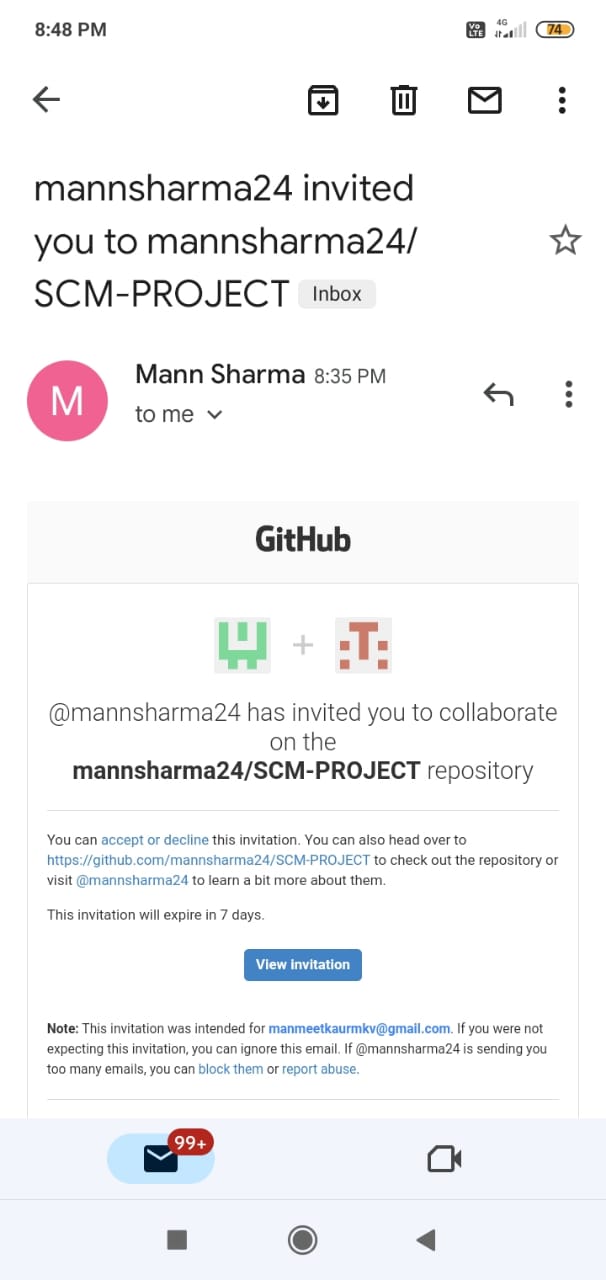


*(Collaborators added by repository owner: team member- Manmeet Kaur,Manav Singh)*

12) To remove any member click on remove option available in the last column of member’s respective row.

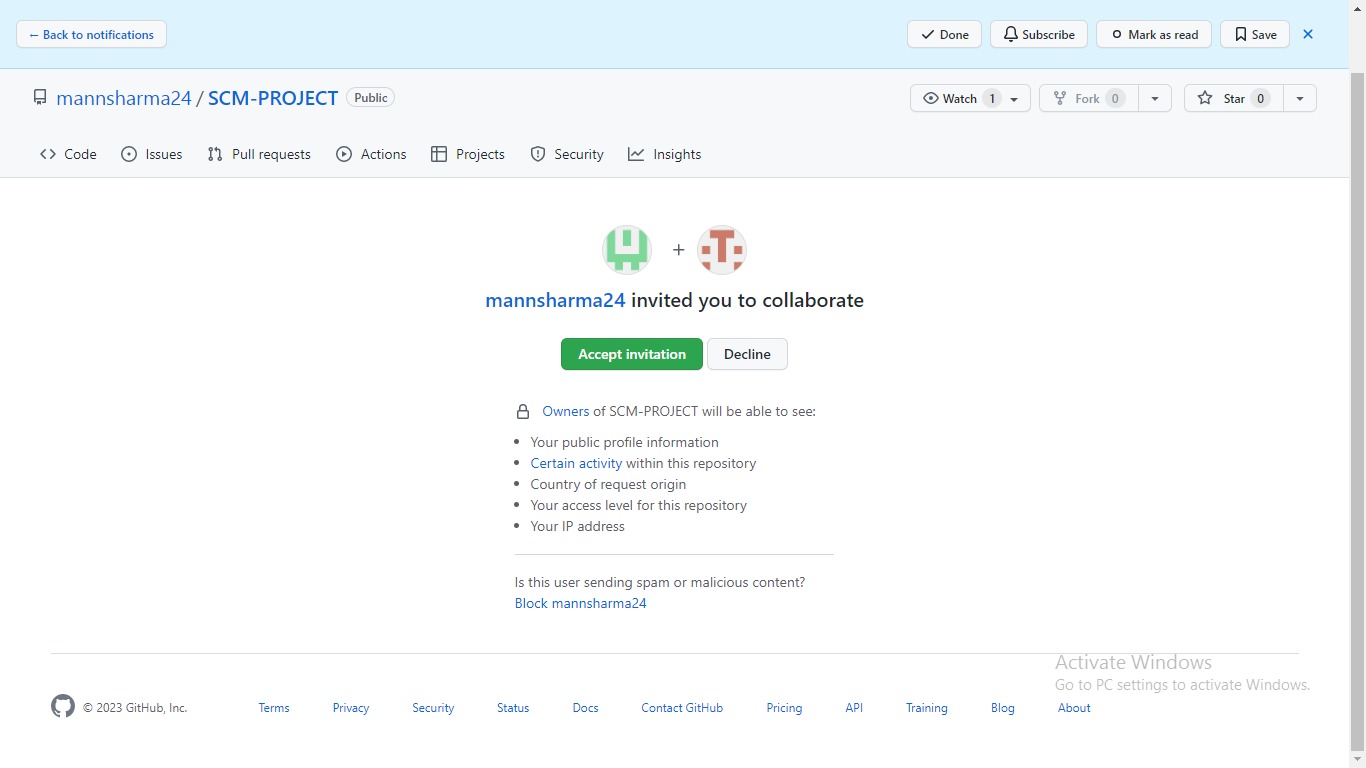


13) To accept the invitation from your team member, open your mail registered with GitHub.

**

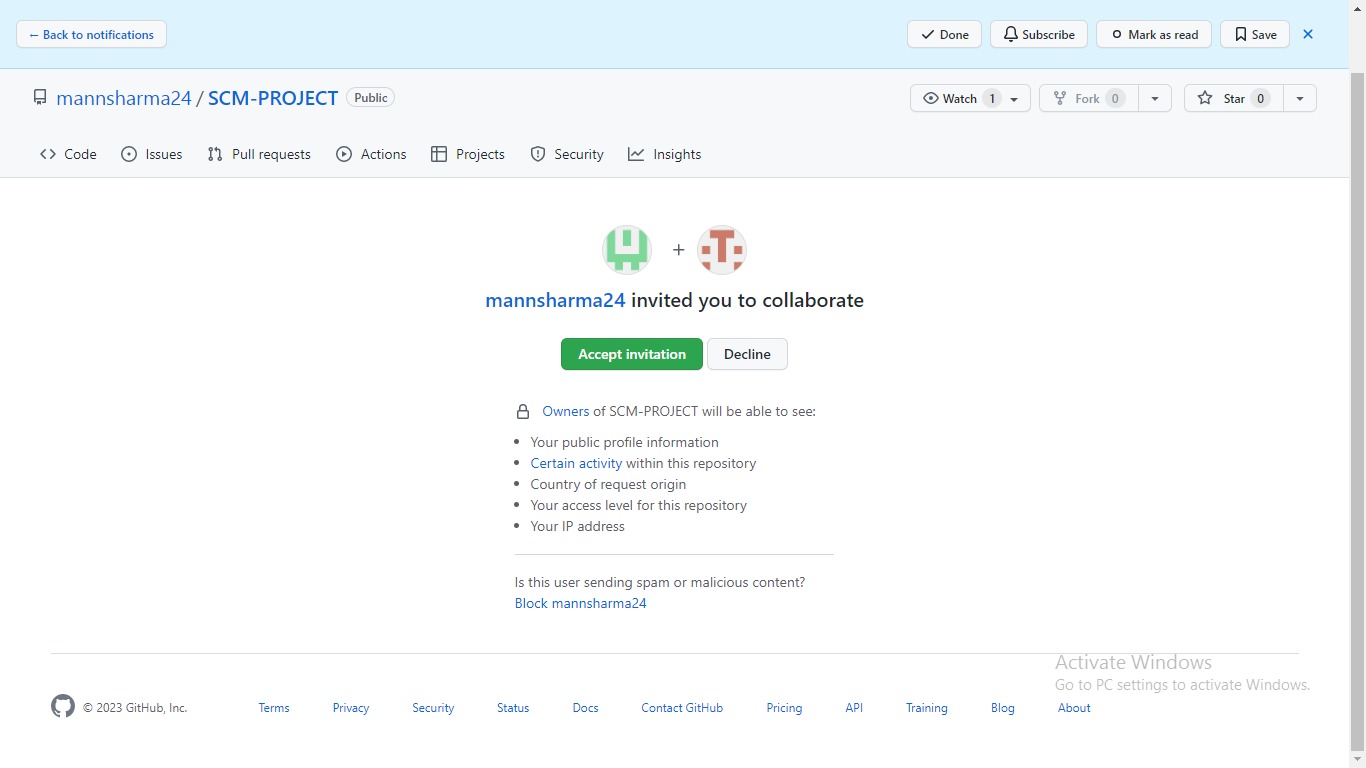
14) You will receive an invitation mail from the repository owner. Open the email and click on accept invitation.

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



16) You will be shown the option that you are now allowed to push.

17) Now all members are ready to contribute to the project.



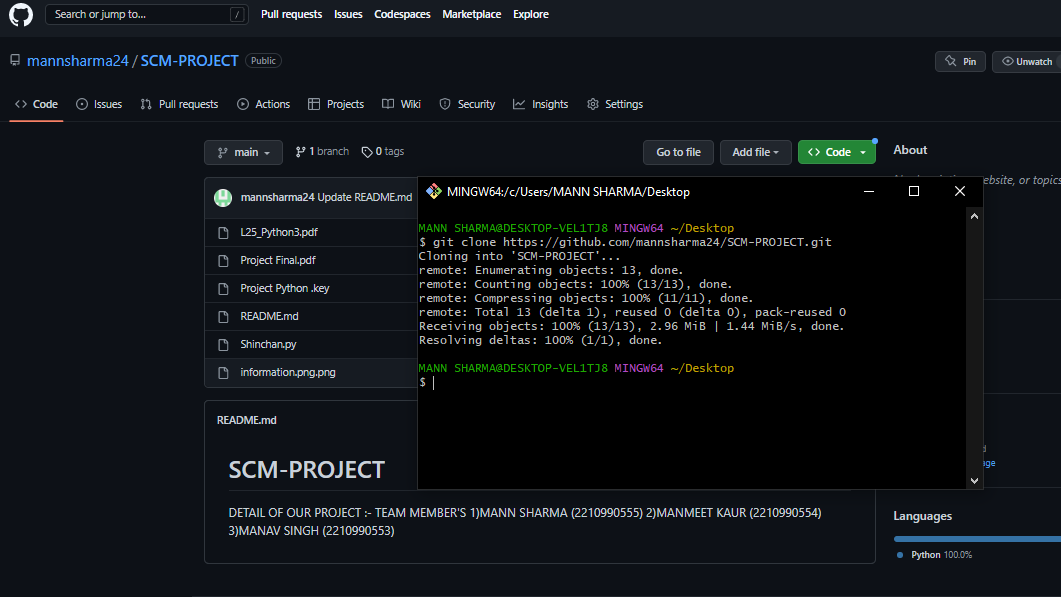
**Experiment No. 02**

**Aim:**

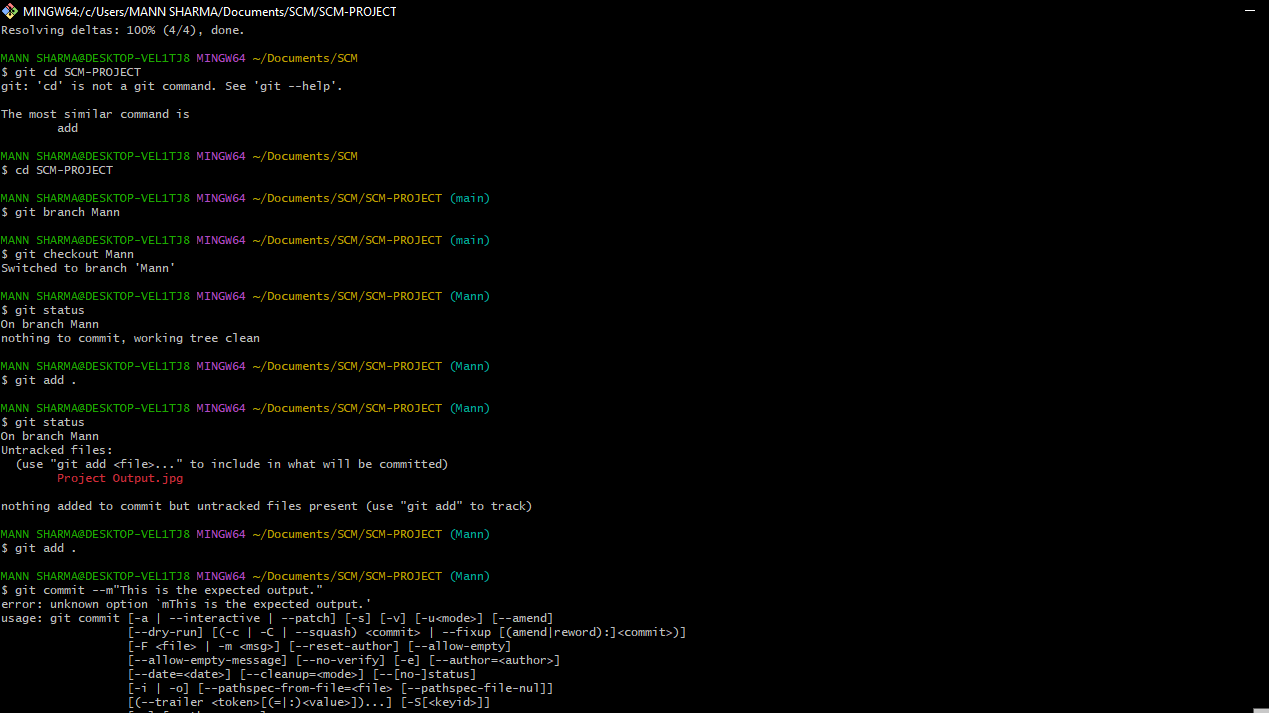
Open and Close a Pull Request

**Procedure:**

* 1. To open a pull request we first have to make a new branch, by using git branch branchname option.

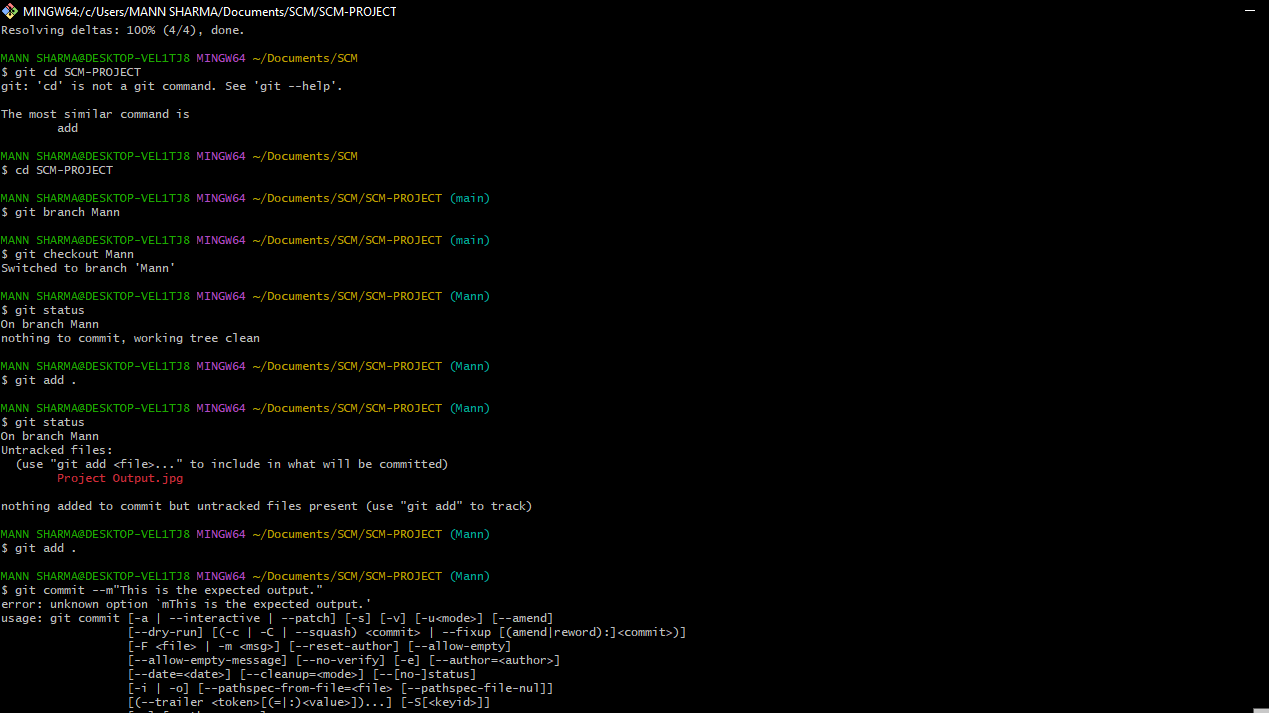


* 1. After making new branch we add a file to the branch or make changes in the existing file.

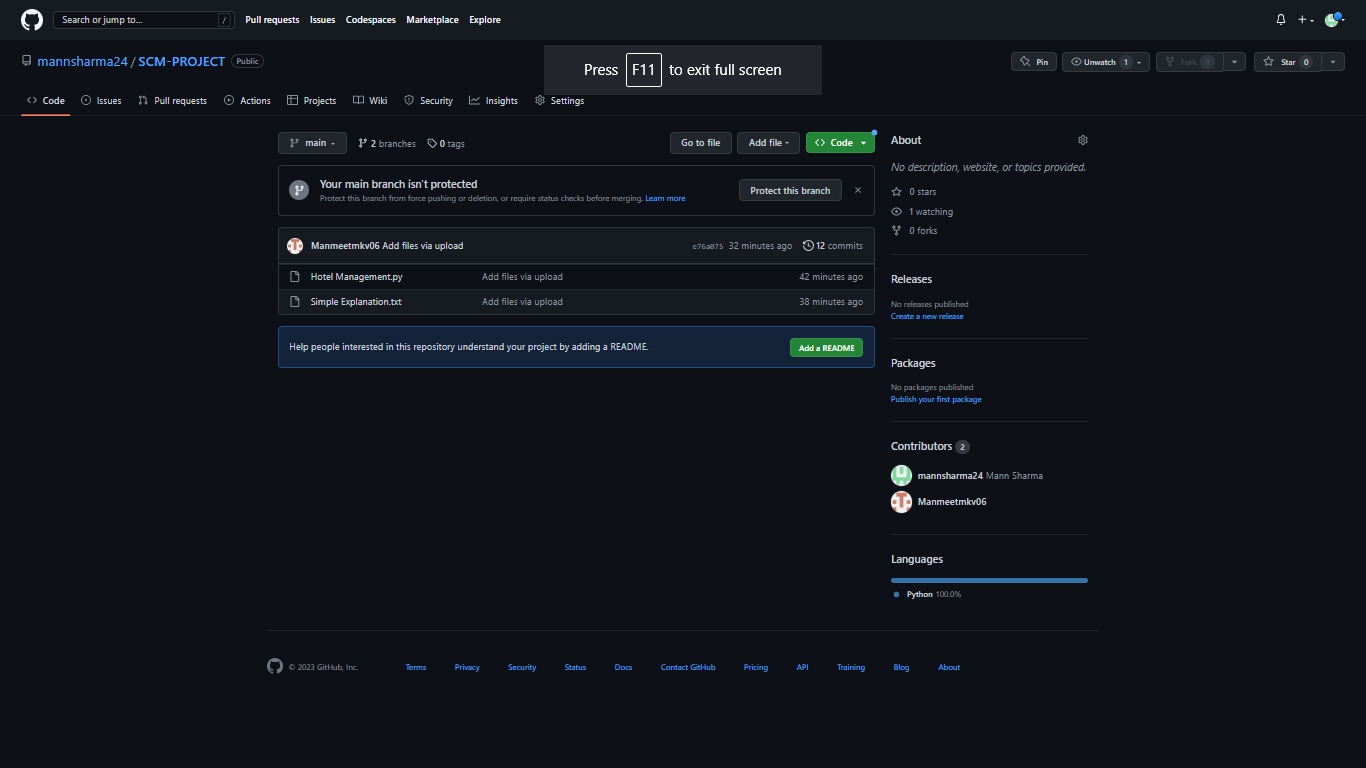


* 1. Add and commit the changes to the local repository.

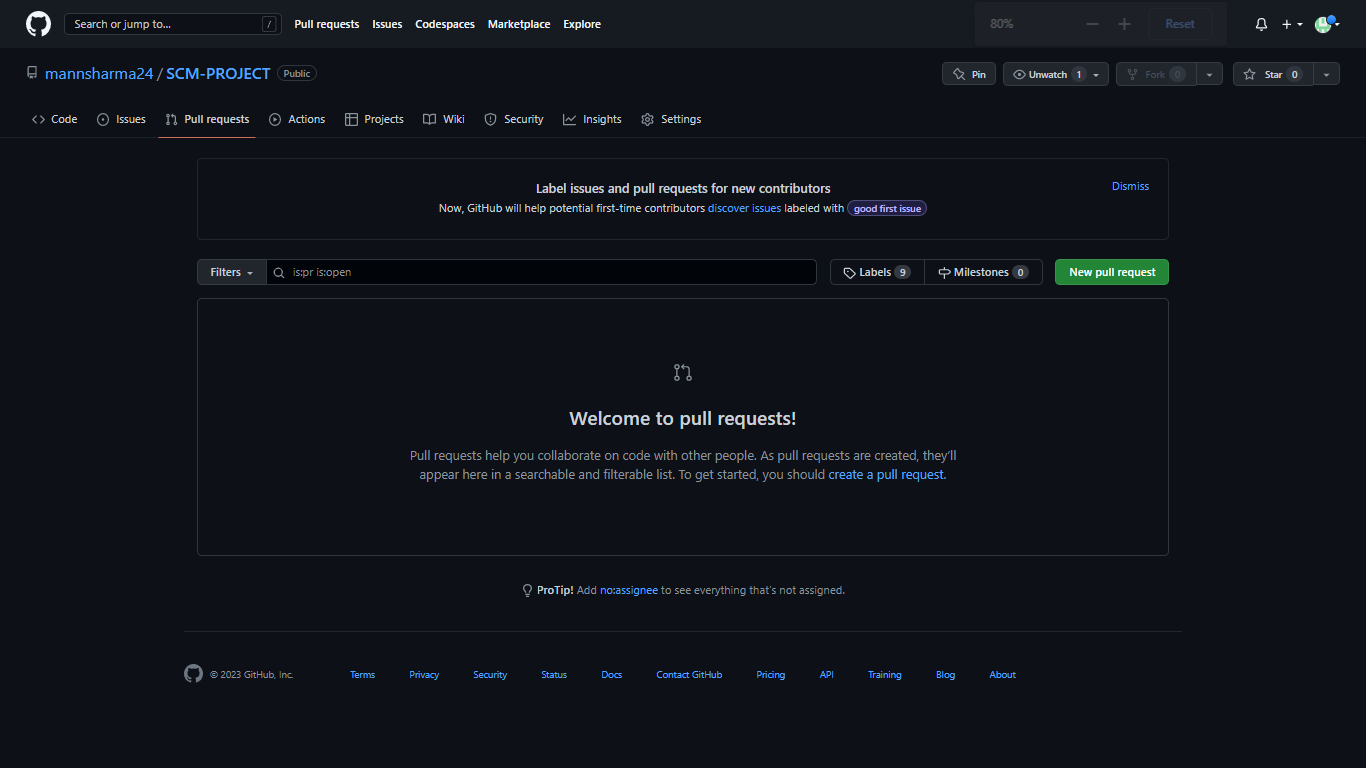
4) Use git push origin branch name option to push the new branch to the main repository.



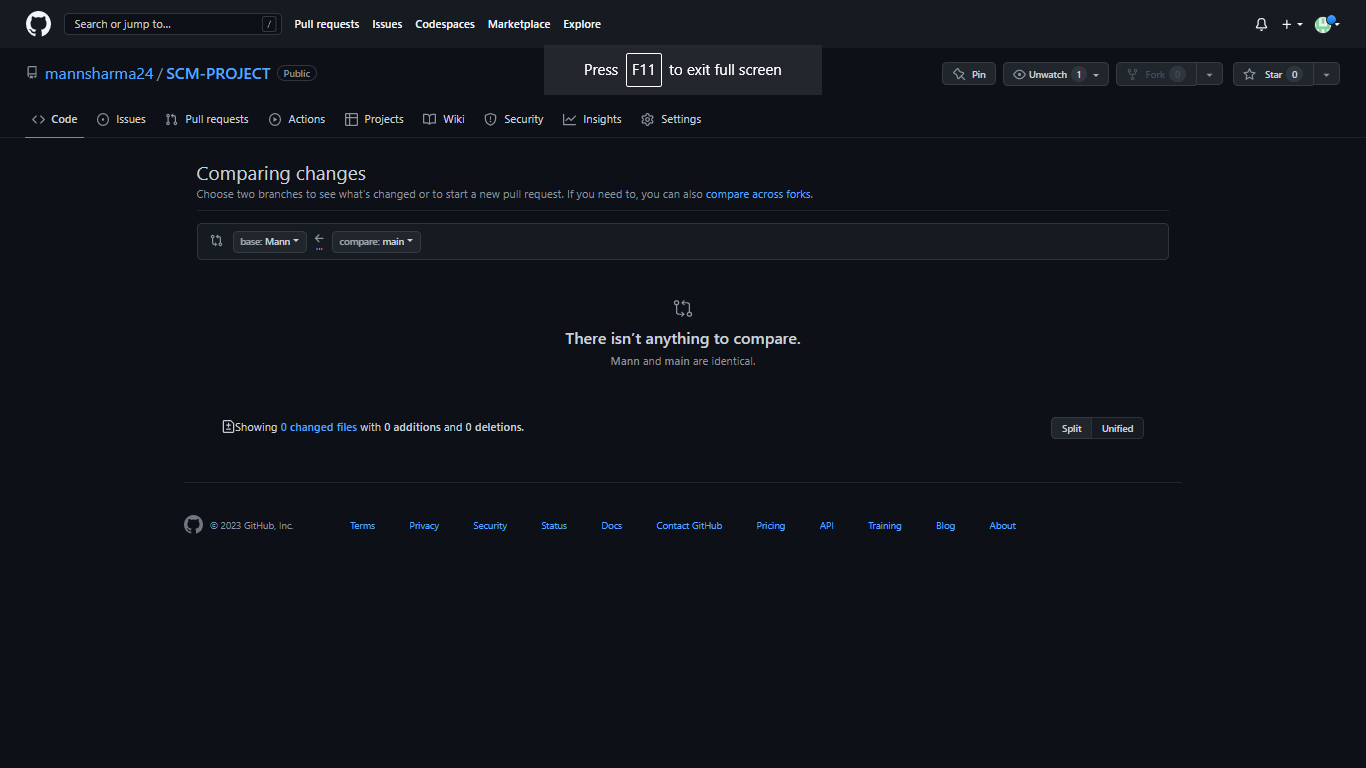
5) After pushing new branch GitHub will either automatically ask you to create a pull request or you can create your own pull request **.**



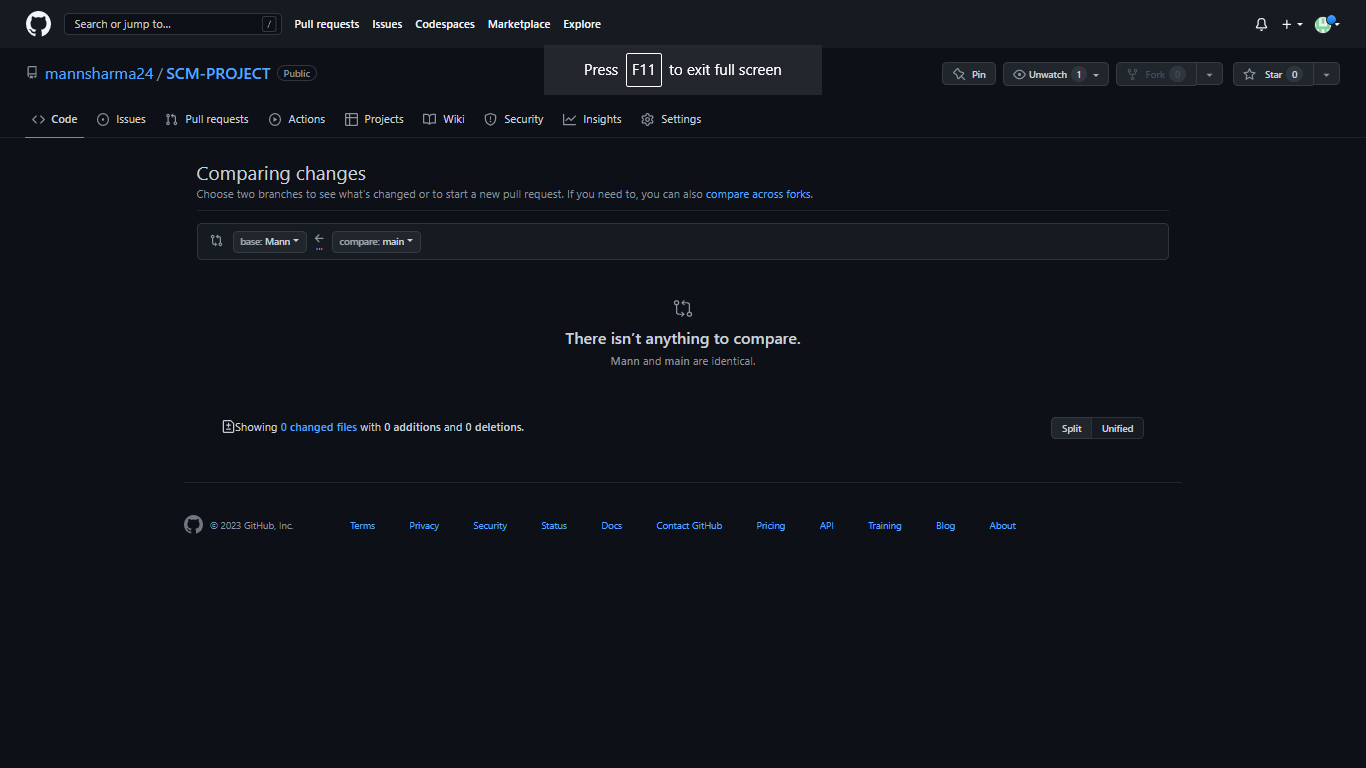
6) To create your own pull request, click on pull request option.



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



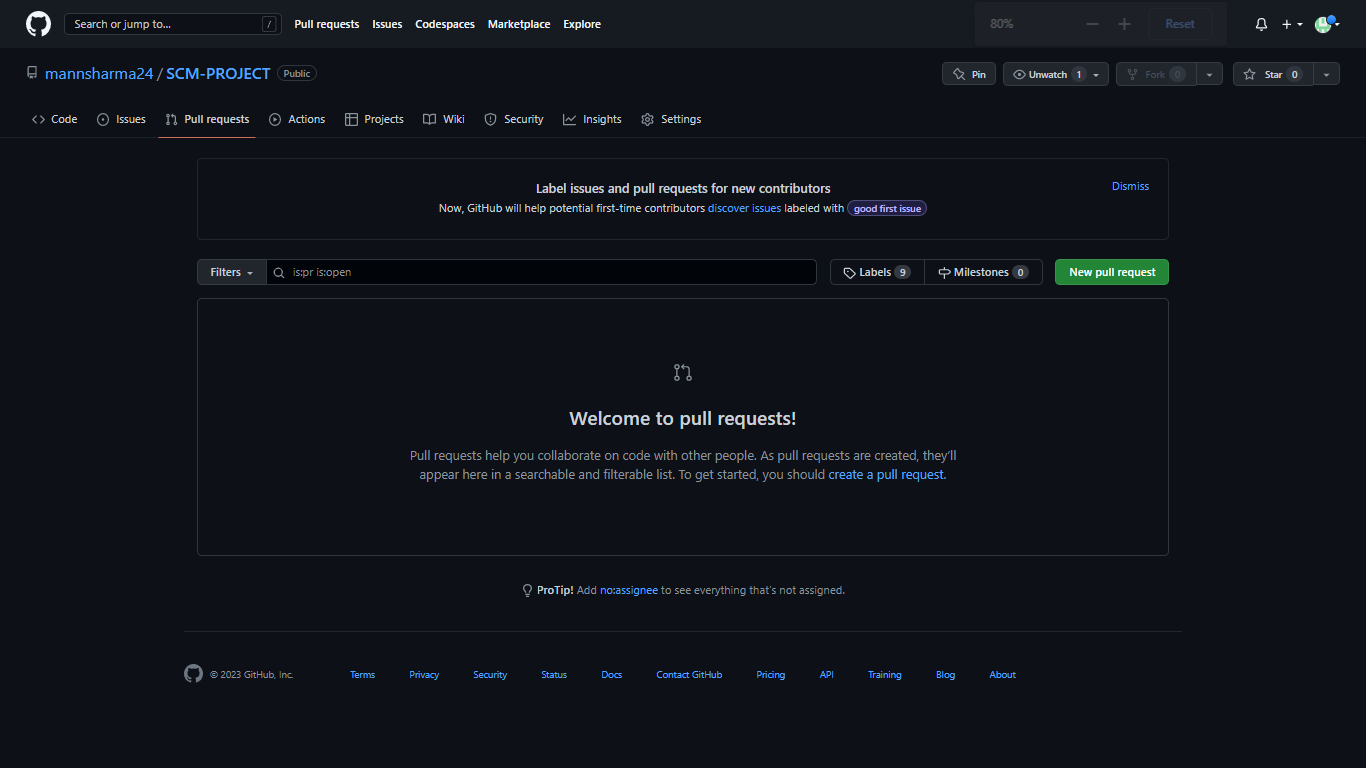
8) After opening a pull request all the team members will be sent the request if they want to merge or close the request.



9) If the team member chooses not to merge your pull request they will close your pull request.

10) To close the pull request simply click on close pull request and addcomment/ reason why you closed the pull request.

11) You can see all the pull request generated and how they were dealt withby clicking on pull request option.

****

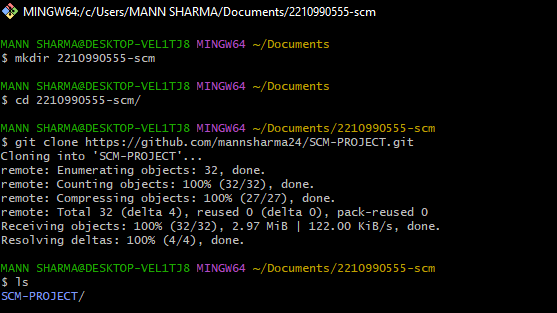
**Experiment No. 03**

**Aim:**

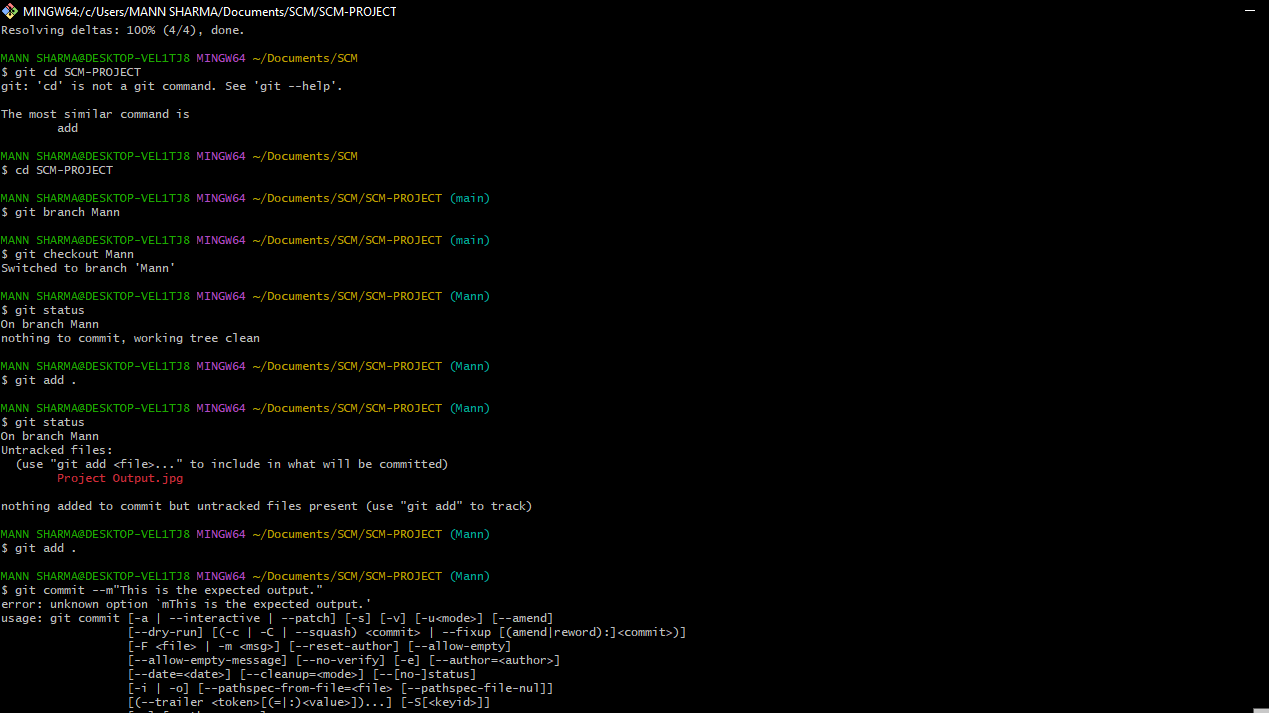
Create a pull request on a team member’s repo and close pull requests generated by team members on own Repo as a maintainer.

To create a pull request on a team member’s repository and close requests by any other team members as a maintainer follow the procedure given below:

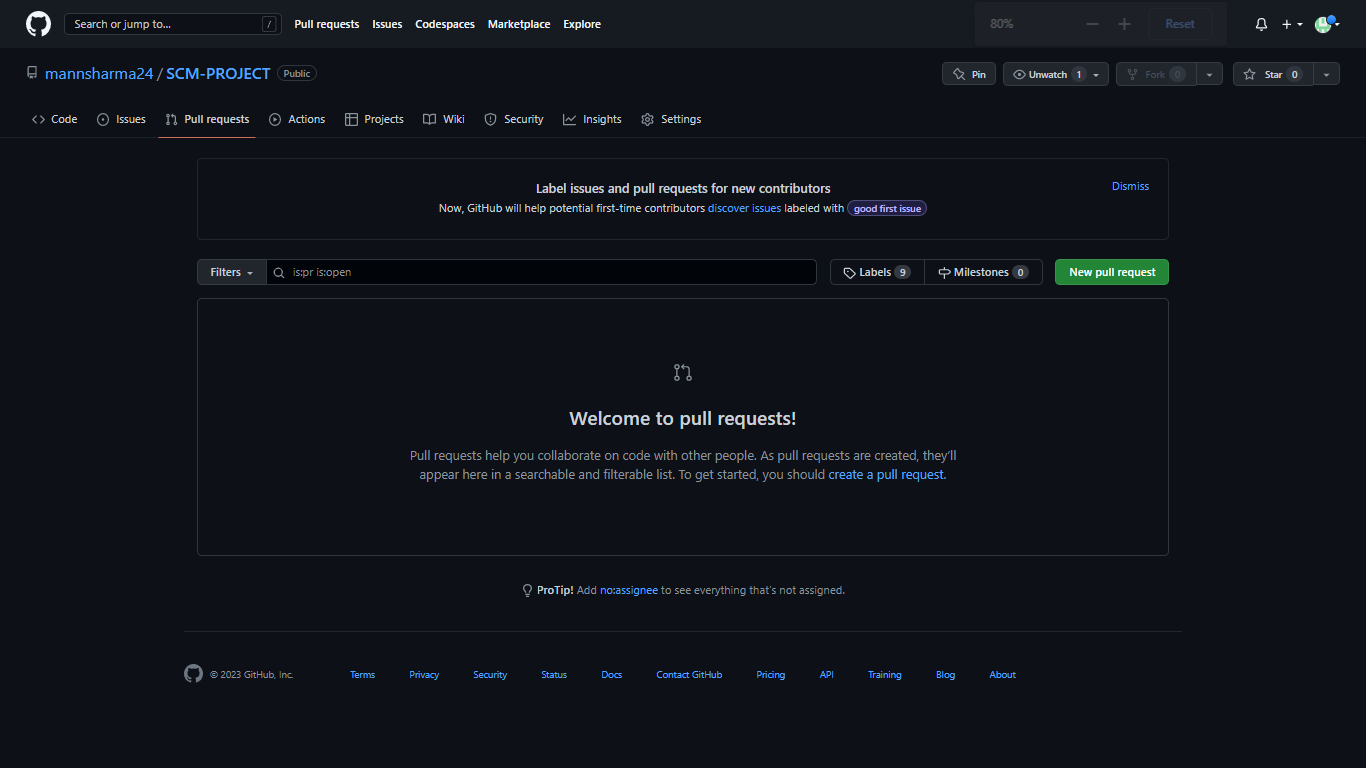
1). Do the required changes in the repository, add and commit these changes in the local repository in a new branch.



2). Push the modified branch using git push origin branch-name.



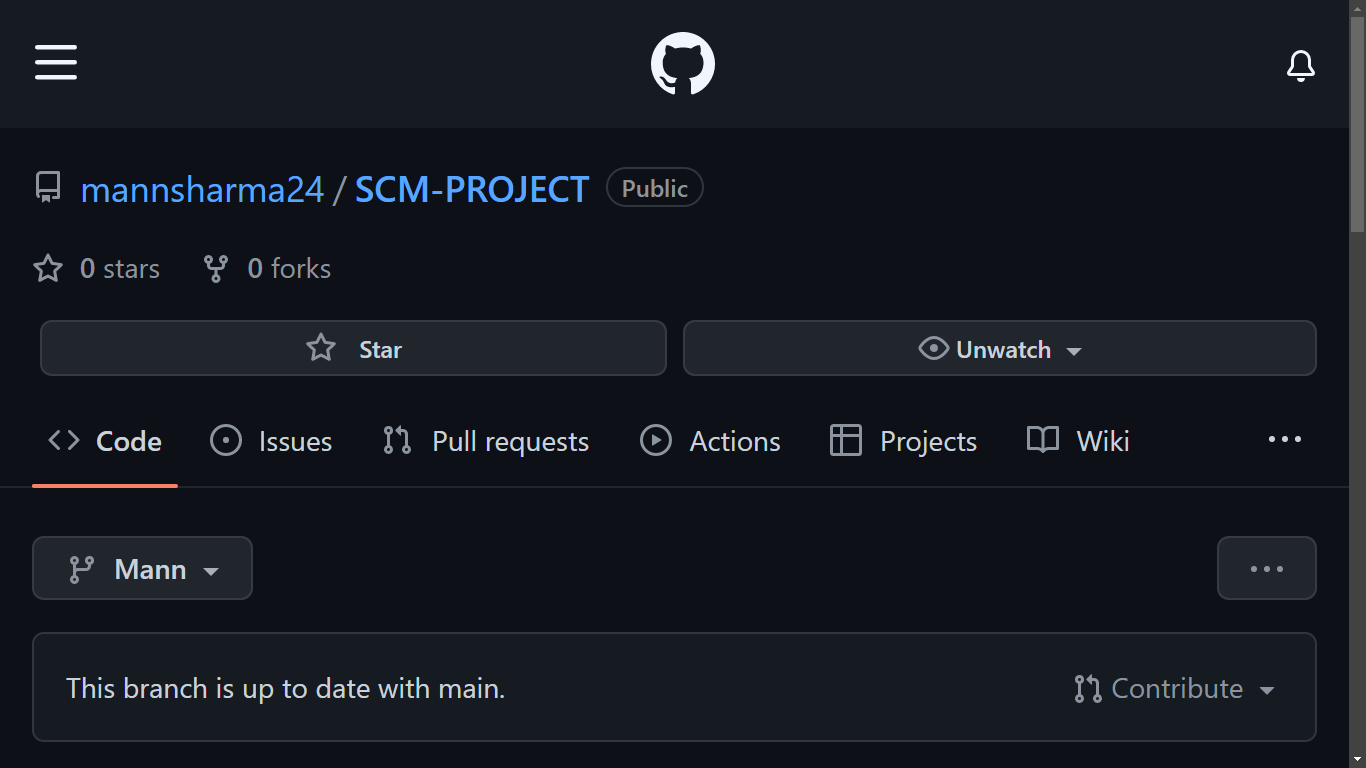
3). Open a pull request by following the procedure from the above experiment.



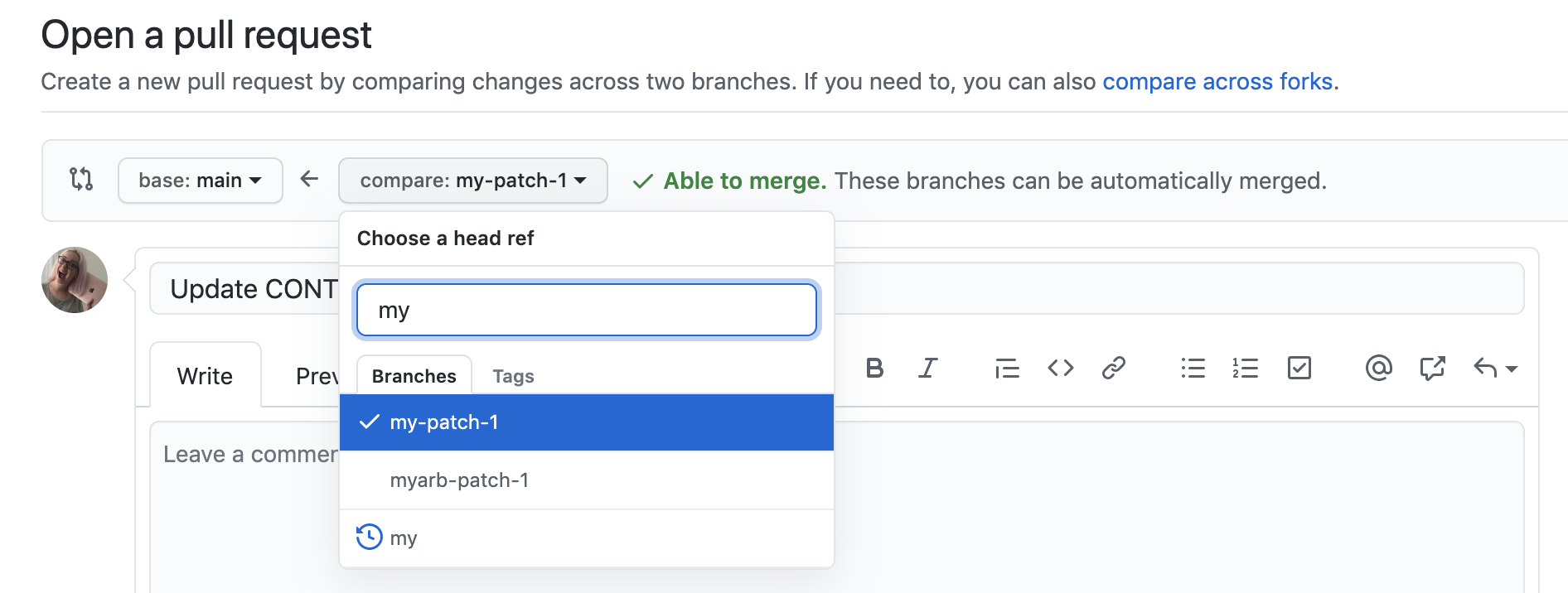
4). The pull request will be created and will be visible to all the team members

5). Ask your team member to login to his/her Github account.

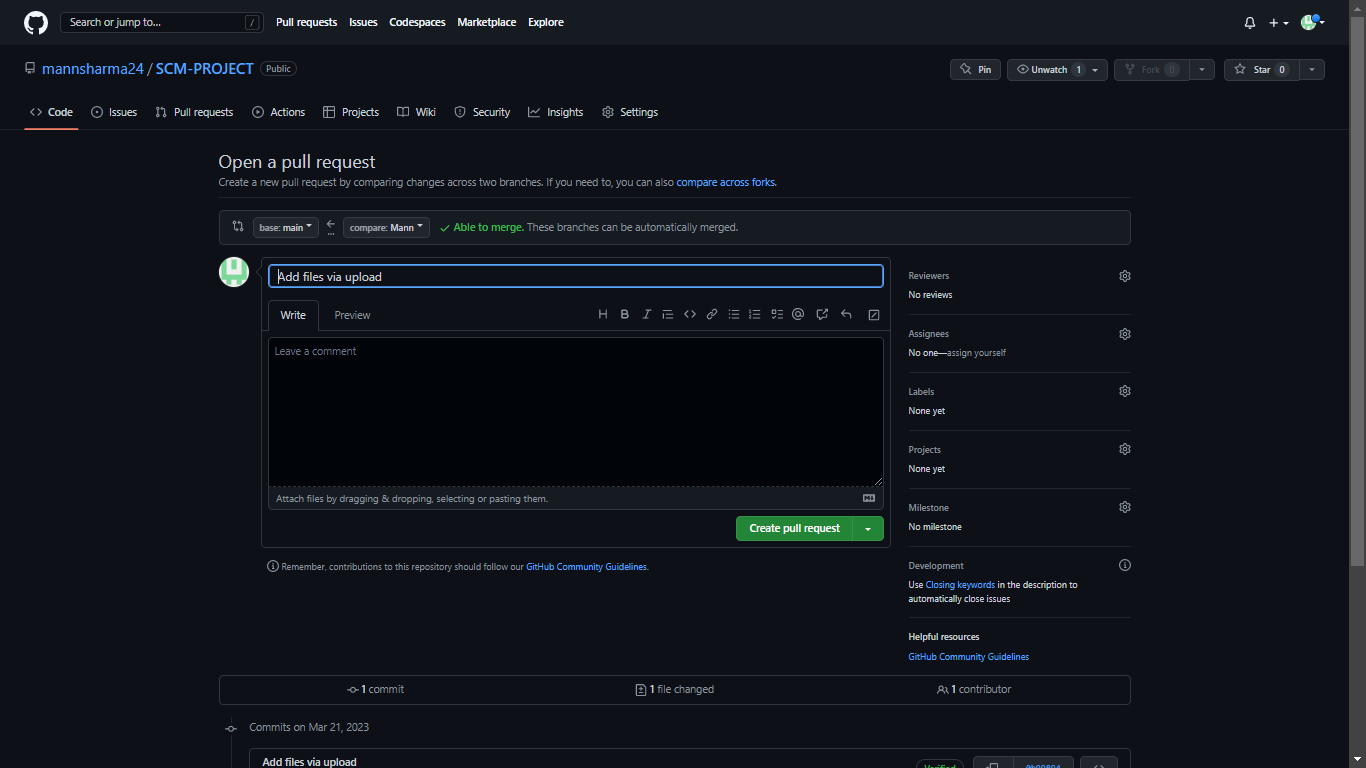
6). They will notice a new notification in the pull request menu.



7). Click on it. The pull request generated by you will be visible to them.



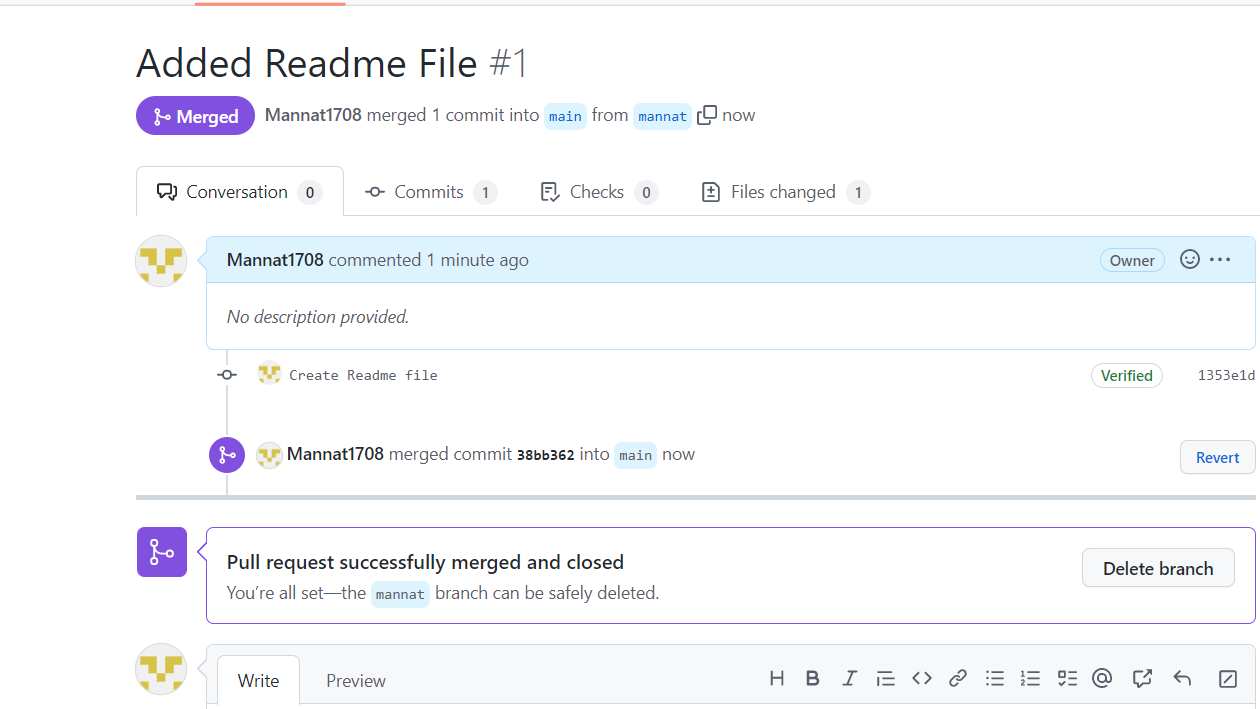
8). Click on the pull request. Two options will be available, either to close the pull request or Merge the request with the main branch.



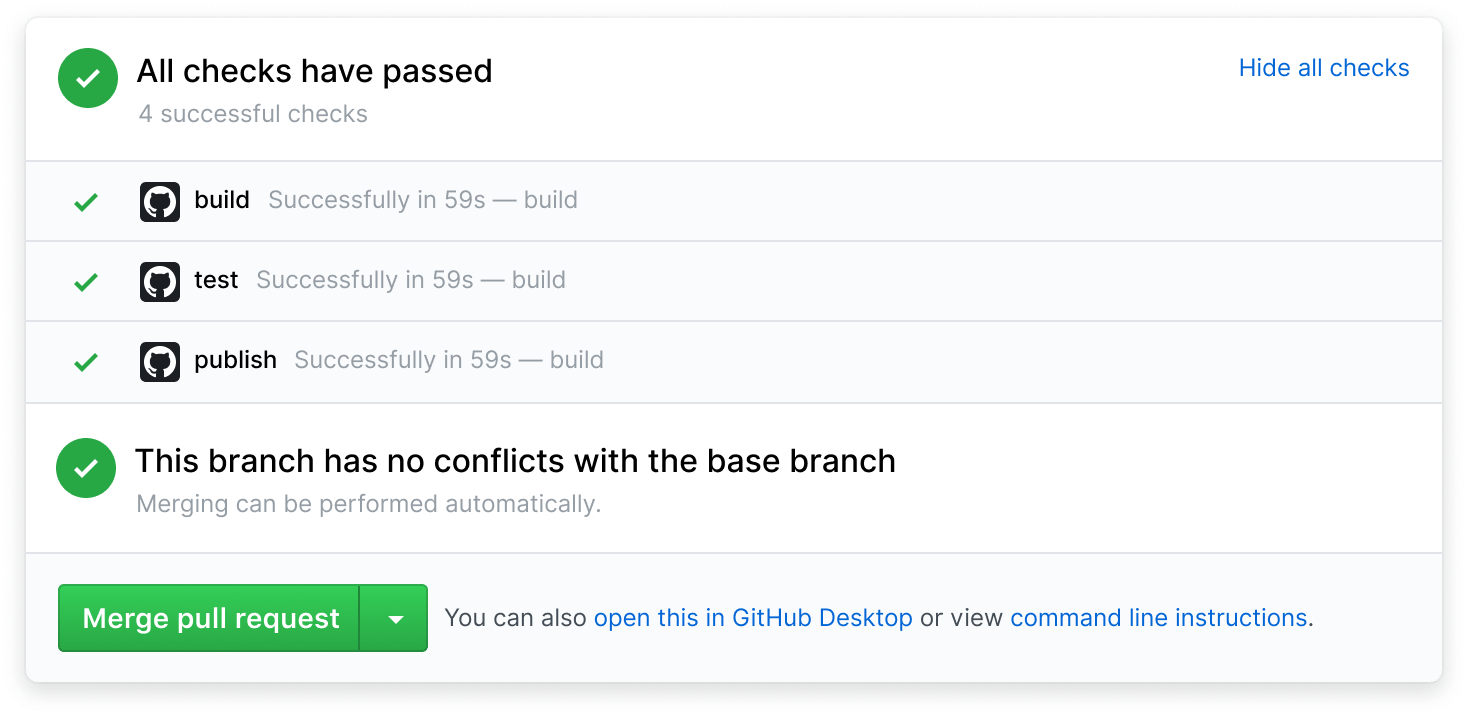
9). By selecting the merge branch option the main branch will get updated for all the team members.

10).By selecting close the pull request the pull request is not accepted and not merged with main branch.

11).The process is similar to closing and merging the pull request by you. It simply includes an external party to execute.



12).Thus, we conclude opening and closing of pull request. We also conclude merging of the pull request to the main branch.



**Experiment No. 04**

**Aim:**

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 graphs give you information on traffic, projects that depend onthe 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 abetter understanding of who's using your repository and why they're using it.

**Some repository graphs are available only in public repositories with GitHub Free:**

**• Pulse**

**•** Contributors

**• Traffic**

**•** Commits

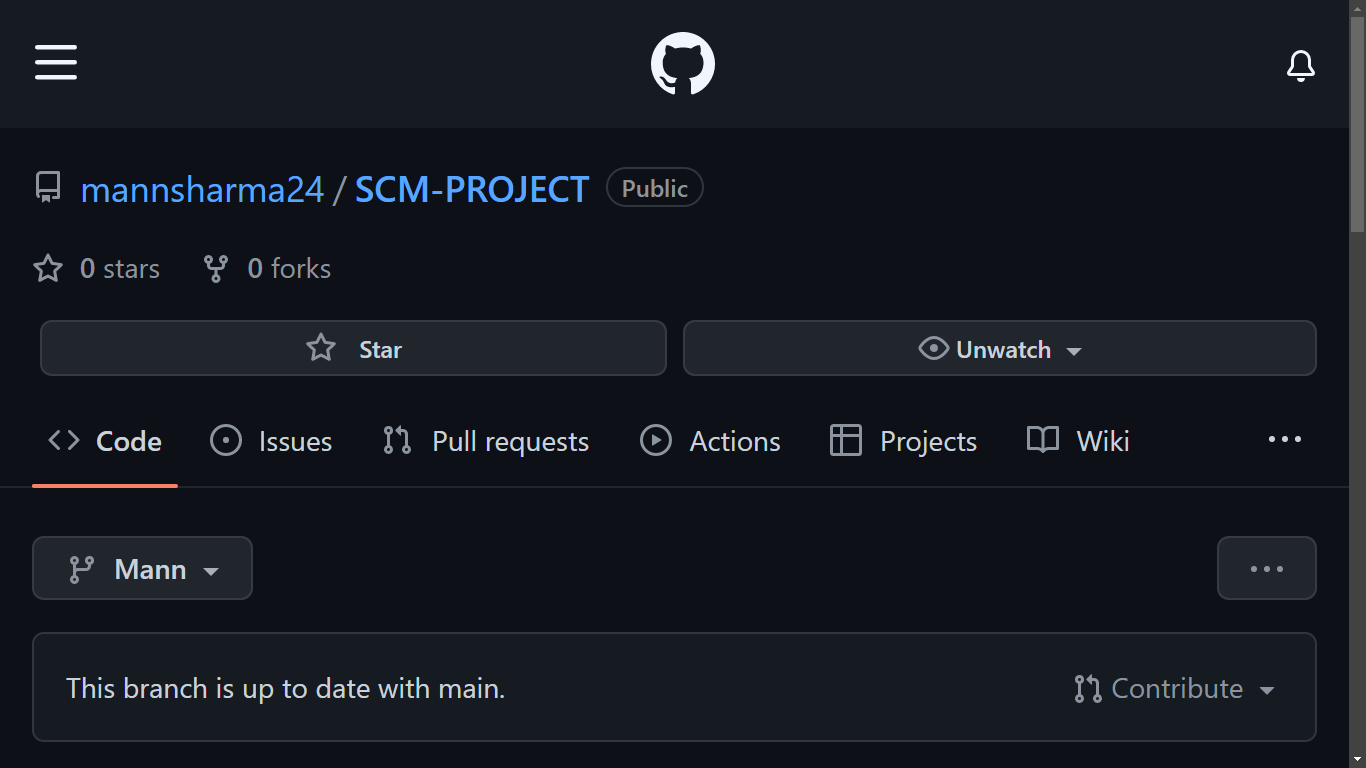
**• Code frequency**

**•** Network

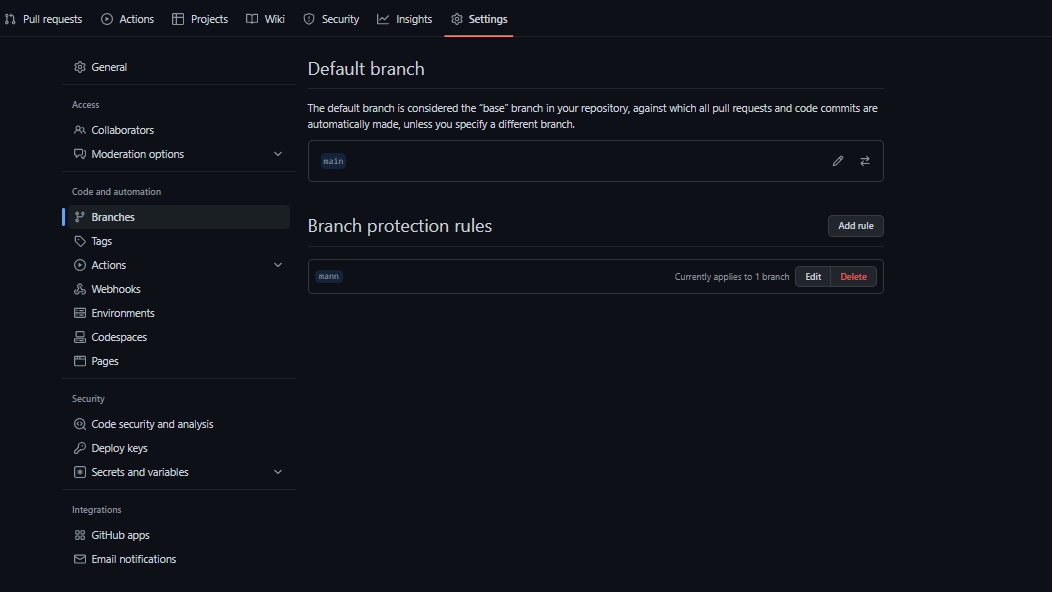
Steps to access network graphs of respective repository

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

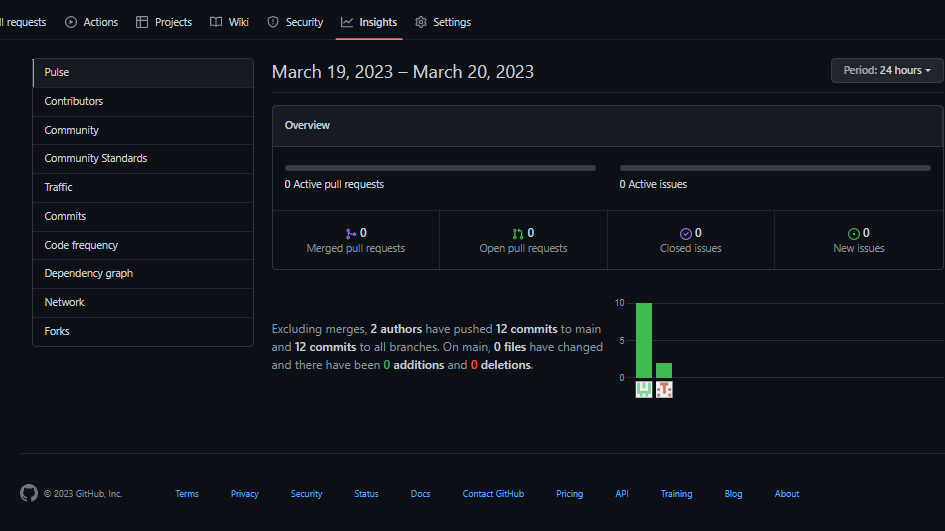
2). Under your repository name, click Insights.



3). At the left sidebar, click on Network.



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 forks that contain commits unique to the network.

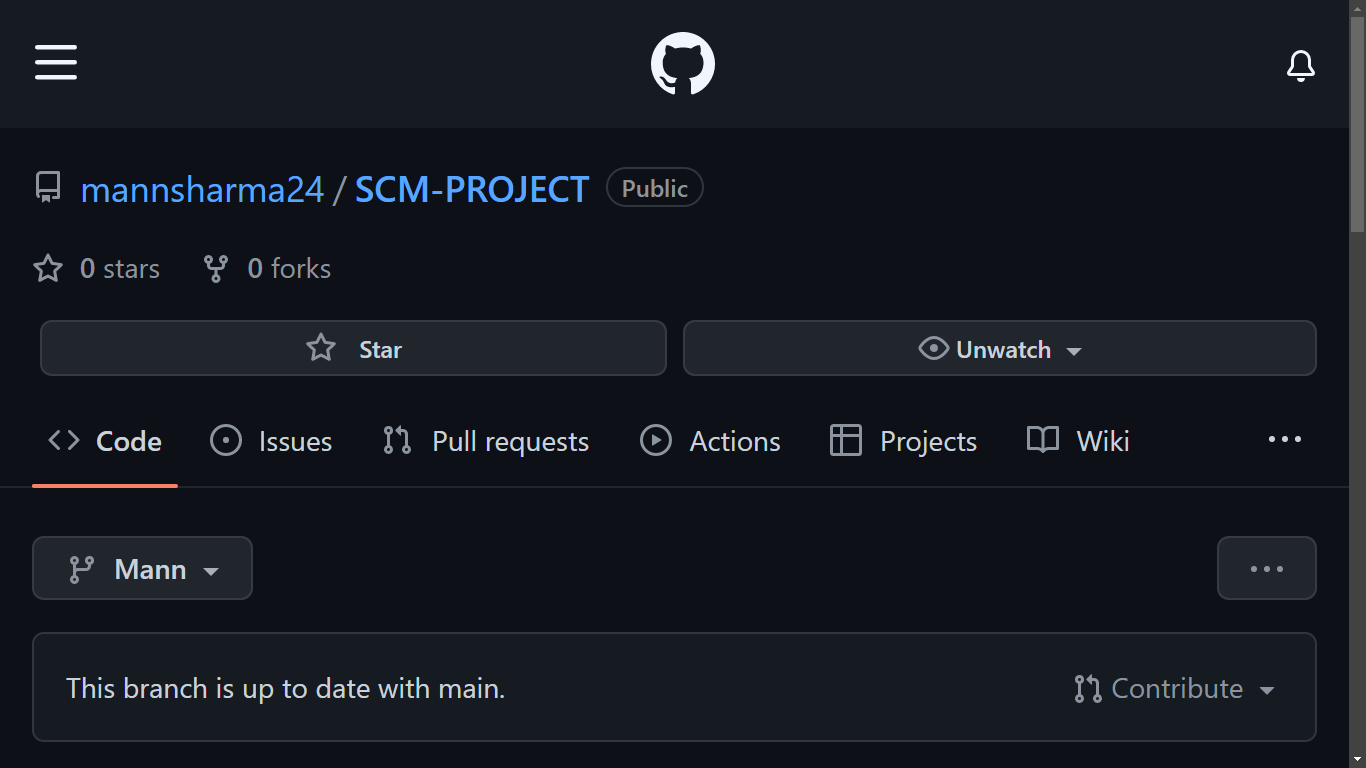


**Listing the forks of a repository**

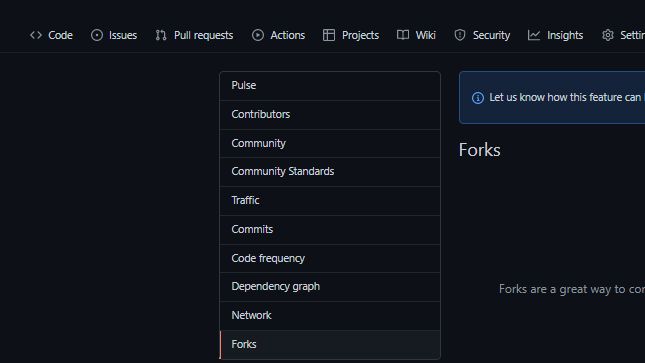
Forks are listed alphabetically by the username of the person who forked the repository Clicking the number of forks shows you the full network. From there you can click "members" to see who forked the repo.

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

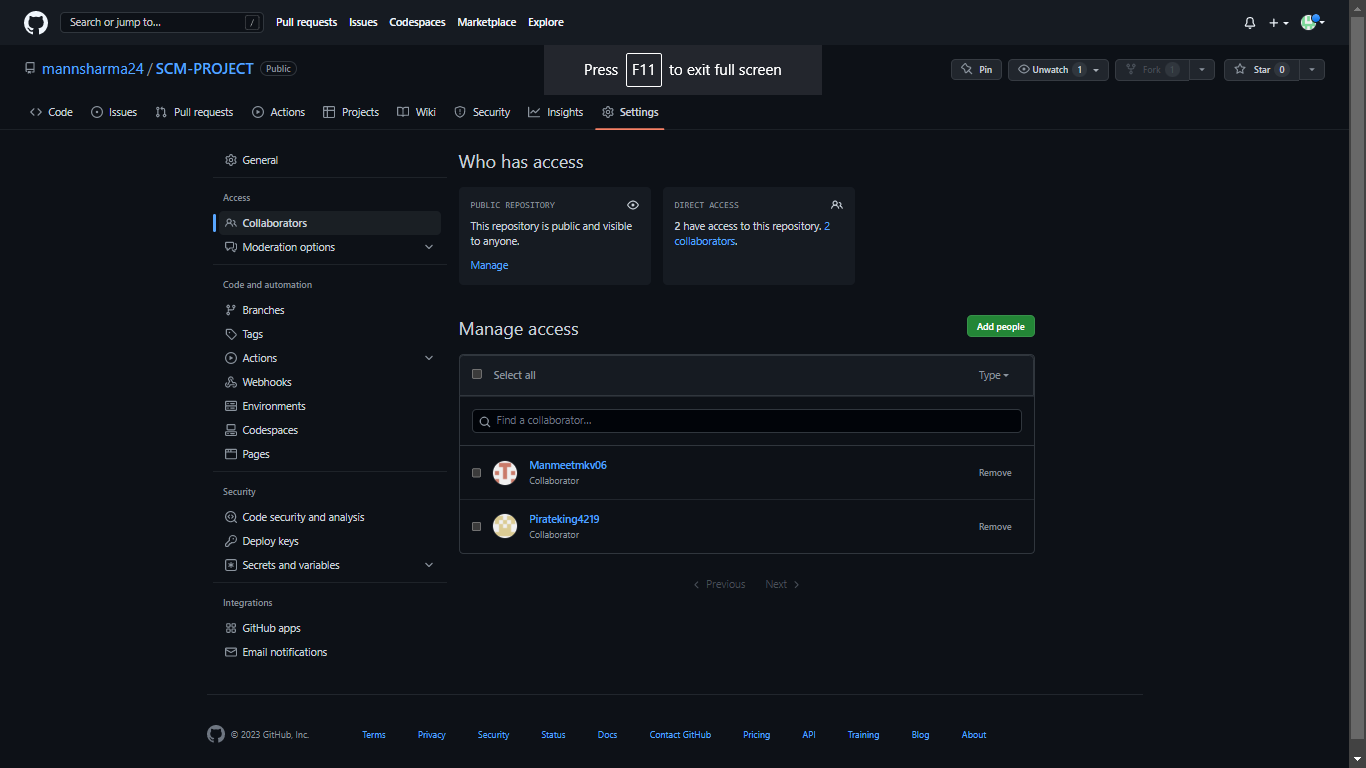
2). Under your repository name, click Insights.

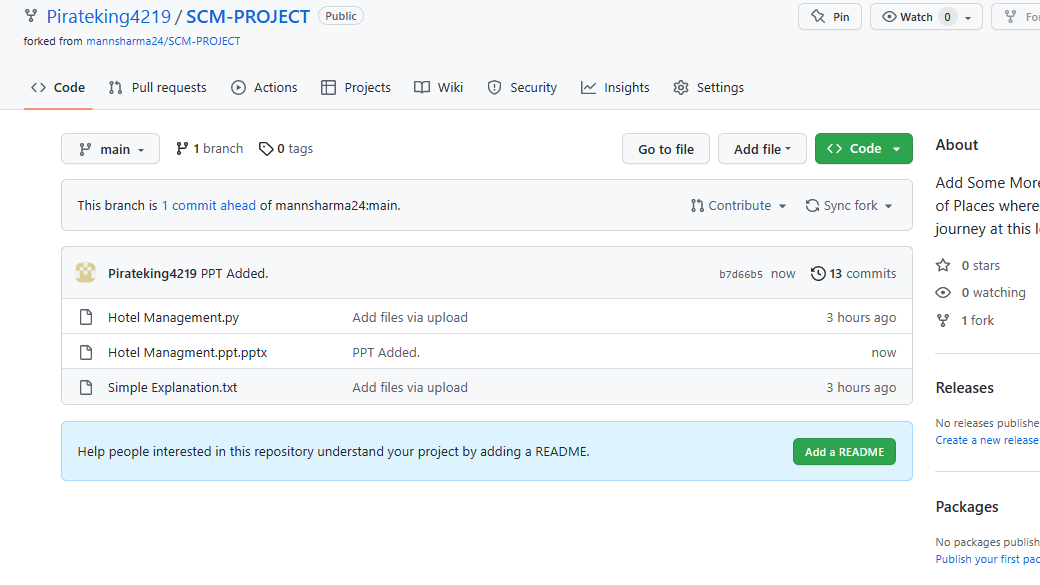


3). In the left sidebar, click Forks



4). Here you can see all the forks





Viewing the dependencies of a repository :-

You can use the dependency graph to explore the code 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.