

A Project report
on
“Task 2”
with
Source Code Management
22CS003

Submitted by:

Name: Harjot Singh
Roll No. 2210990363

Submitted To:

Dr. Prabjot Kaur Chahal

Team Member 1 Name: Hardik Singh

Roll No. 2210990358

Team Member 2 Name: Harjot Singh

Roll No. 2210990363

Team Member 3 Name: Harinder Singh

Roll No. 2210990361

CHITKARA
UNIVERSITY



CHITKARA
UNIVERSITY



Institute/School: **Chitkara University Institute of Engineering and Technology**
Name

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** Semester/Batch: **1/2022**
-

Vertical Name: - **Beta** Group No: - **G23-01**

Faculty Name :- **Dr. Prabhjot Kaur Chahal**

Signature:-

INDEX

S. NO	Experiment Name	Remarks
1.	Introduction	
2.	Problem statement	
3.	Solution	
4.	Objective	
5.	Create a distributed Repository and add members in project team	
6.	Open and close a pull request	
7.	Create a pull request on a team members repo and close pull requests generated by team members on own Repo as a maintainer	
8.	Publish and print network graphs	

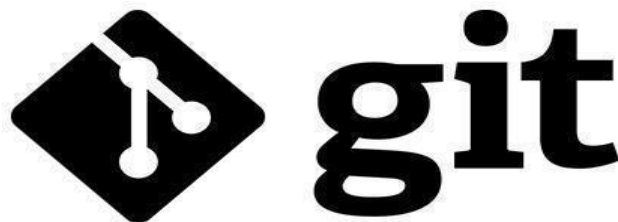
Introduction

What is GIT and why is it used?

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. It was developed in 2005 by Linus Torvalds, the creator of the Linux operating system kernel.

Git is a speedy and efficient distributed [VCS](#) tool that can handle projects of any size, from small to very large ones. Git provides cheap local branching, convenient staging areas, and multiple workflows. It is free, open-source software that lowers the cost because developers can use Git without paying money. It provides support for non-linear development. Git enables multiple developers or teams to work separately without having an impact on the work of others.

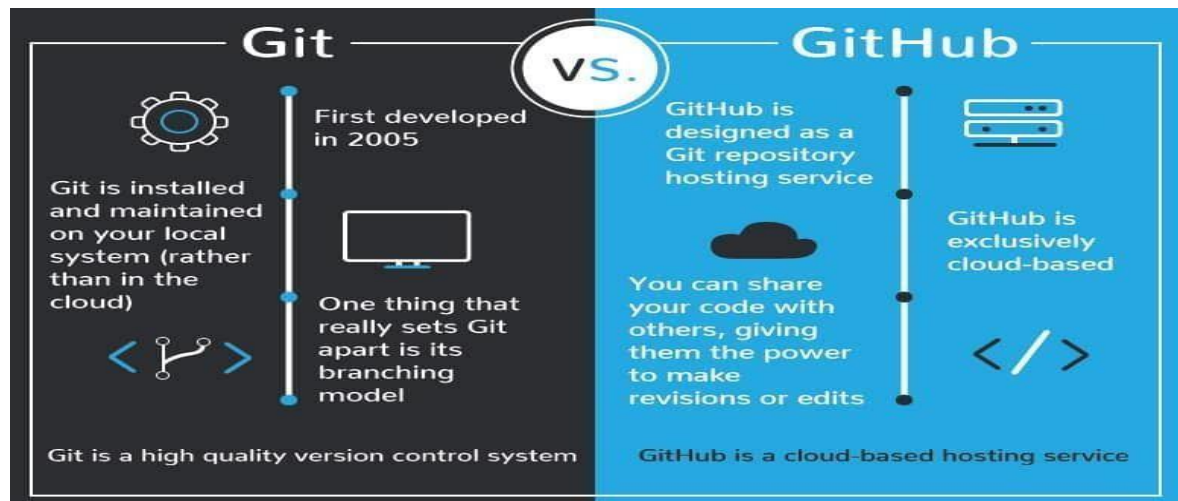
Git is an example of a distributed version control system (DVCS) (hence Distributed Version Control System).



What is GITHUB?

It is the world's largest open-source software developer community platform where the users upload their projects using the software Git.

What is the difference between GIT and 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

1. **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.

AI. Centralized Version Control System: In the Centralized Version Control

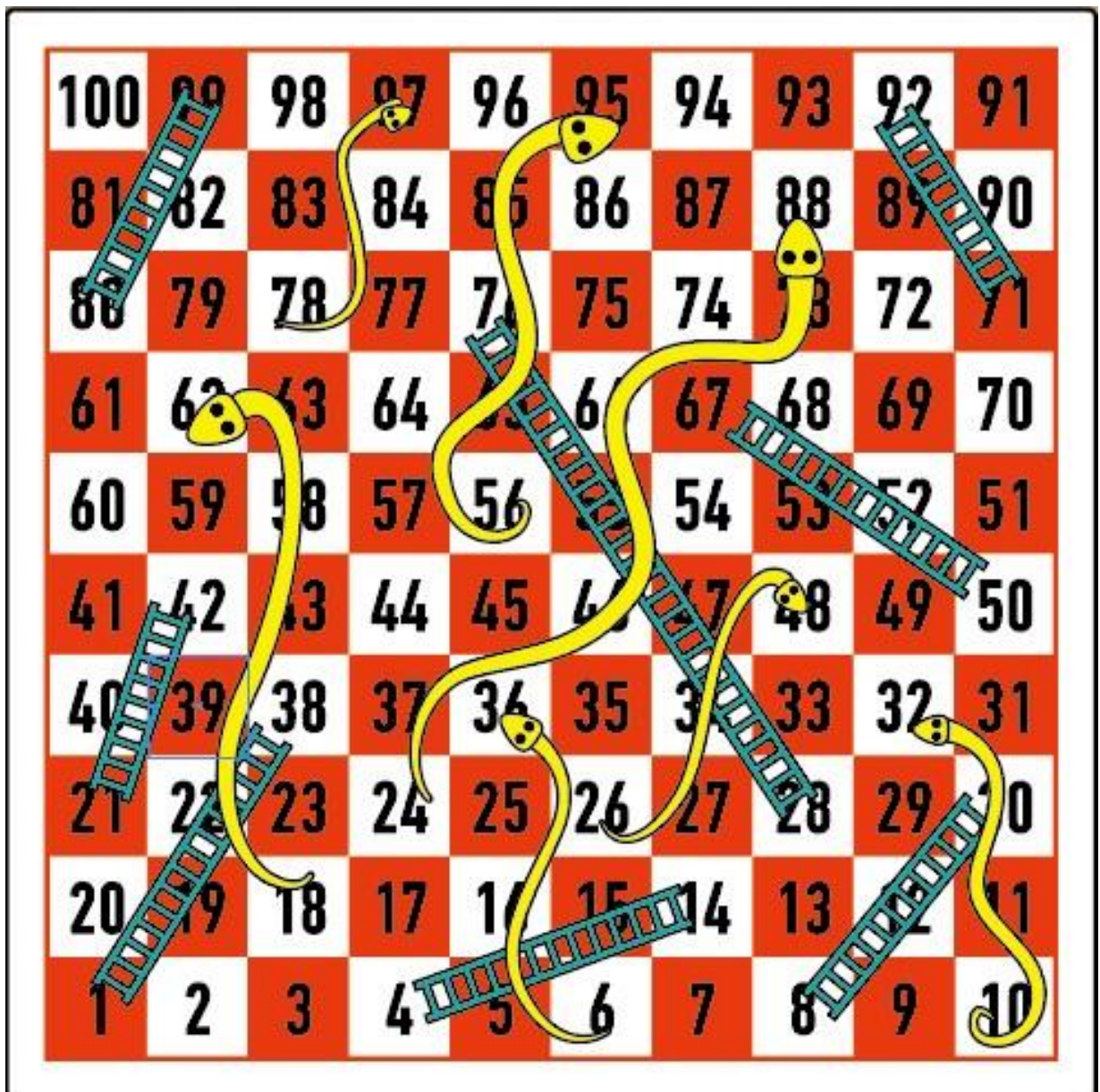
Systems, there will be a single central server that contains all the files related to the project, and many collaborators checkout files from this single server (you will only have a working copy). The problem with the Centralized

Version Control Systems is if the central server crashes, almost everything related to the project will be lost.

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

Problem Statement

"Build a Game and deploy it on GitHub" In today's world of technology where the youngsters are getting distant from the classic games, we tried to inculcate our programming skills to create a virtual "Snakes & Ladders" game.



Solution

The Snakes & Ladders World can be entered by having an account with username and password. It is enjoyable for all irrespective of the age and gender of the player. This program comes with an inbuilt dice and a simple and elegant, yet a beautiful and colorful inbuilt game board. The account of a player is steadily protected by the security system of this program. Moreover, an account can only be accessed by using its respective credentials. Despite of the huge features of this program, the processing time of the program is very fast, enhancing no to zero time lags while playing. The Game offers inbuilt currency (coin) and trophy system using which a user can participate in tournaments and win trophies to get placed in a top league, get a name title and a star rating. Working on the project, we learnt about co-operation & team work with our peers.

```
Welcome to the game 'Snakes & Ladders'
Select a venture from the below list -
```

1. Play Game
2. Create Account
3. Leagues & Rankings
4. Rules and Instructions
5. Player Information

```
Welcome to the Account Creation Menu.
Fill up the following fields to create your Account.
```

```
Enter your Name: Player1
Enter Your Mail ID: Player1@gmail.com
Select a single character symbol for your Pawn: $
Create a Unique Username: Player1
Create a password: Player1
Reconfirm the password: Player1
```

```
Congratulations, Your Account Has Been Created.
10,000 Coins have been allotted to you.
Good luck for your journey to the topmost leagues.
```

```
Fill in the required credentials to proceed.
```

```
Username: Player1
Enter Your Password: Player1
```

```
Welcome, Player1
Select the Match you want to play:
```

```
Players | Cost | Rewards
```

2		25		30
3		21		40
4		18		50
5		15		60

```
Enter Number of Players: █
```


Objective:

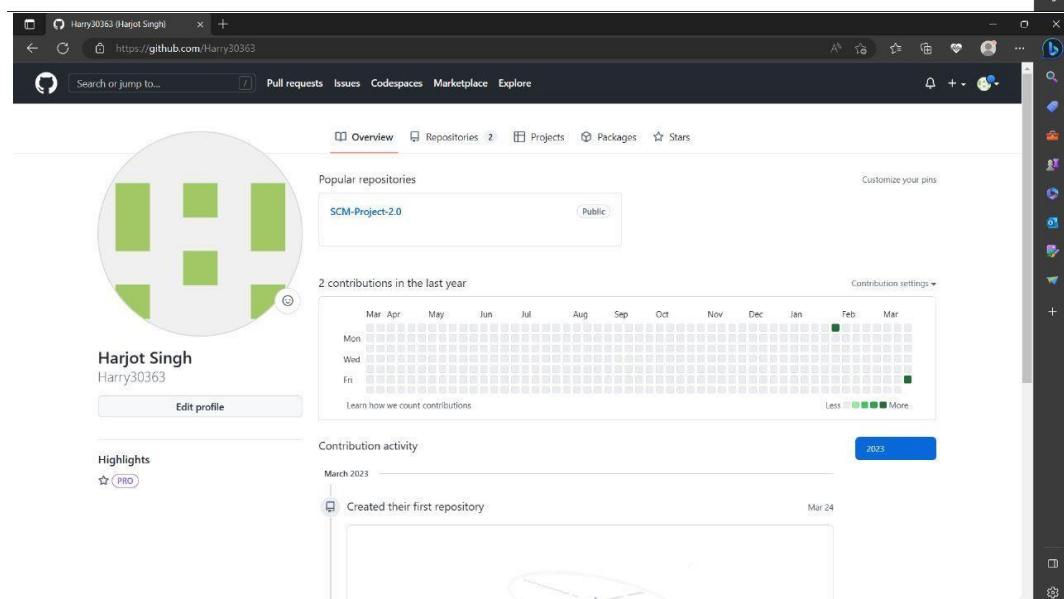
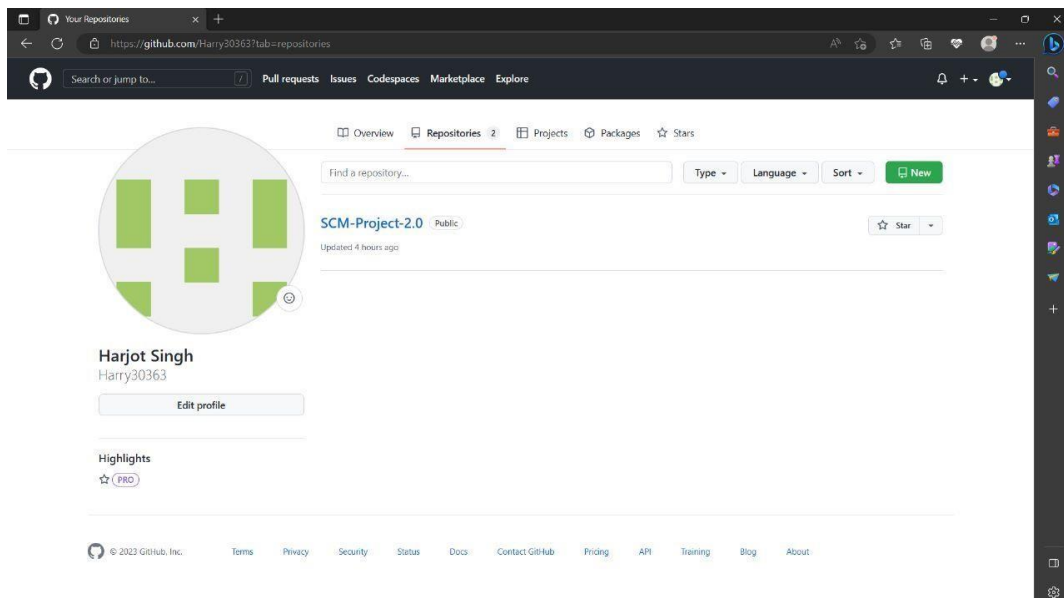
The objective of this project is to associate programming with git because:

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.
-

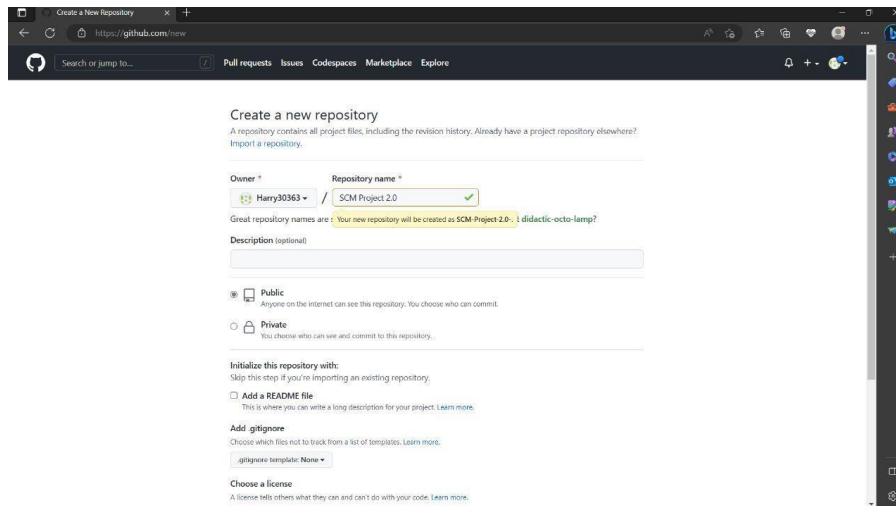
EXPERIMENT NO 1

Aim: Create a distributed Repository and add members in project team

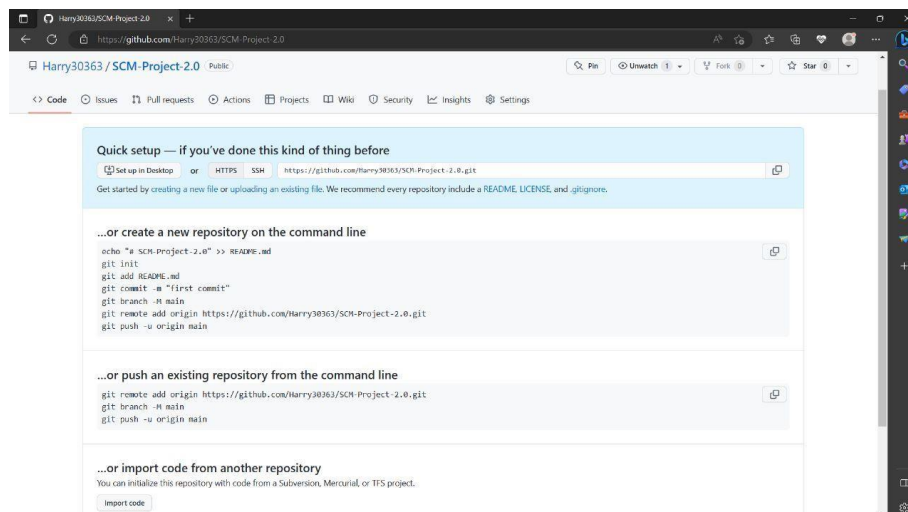
- 1) Login to your GitHub account and you will land on the homepage as shown below. Click on Repositories option in the menu bar.
- 2) Click on the 'New' button in the top right corner.
- 3) Enter the Repository name and add the description of the repository.



4) Select if you want the repository to be public.

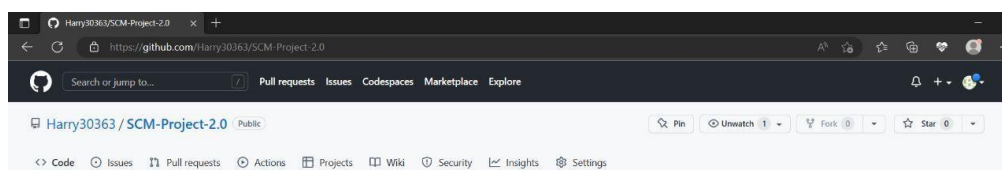


5) If you want to import code from an existing repository select the import code option.

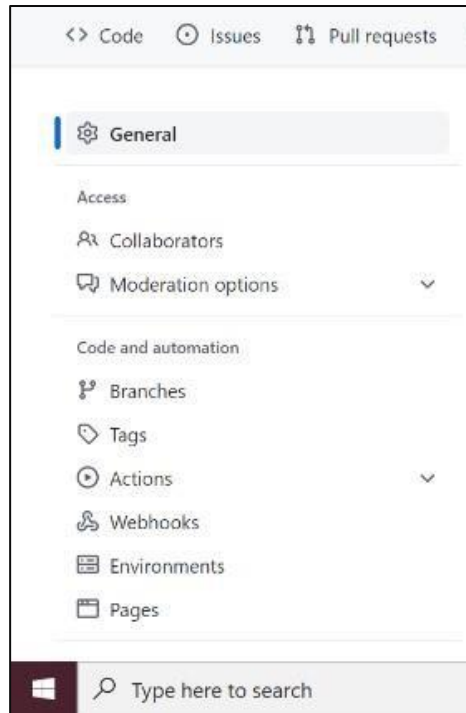


6) Now, you have created your repository successfully.

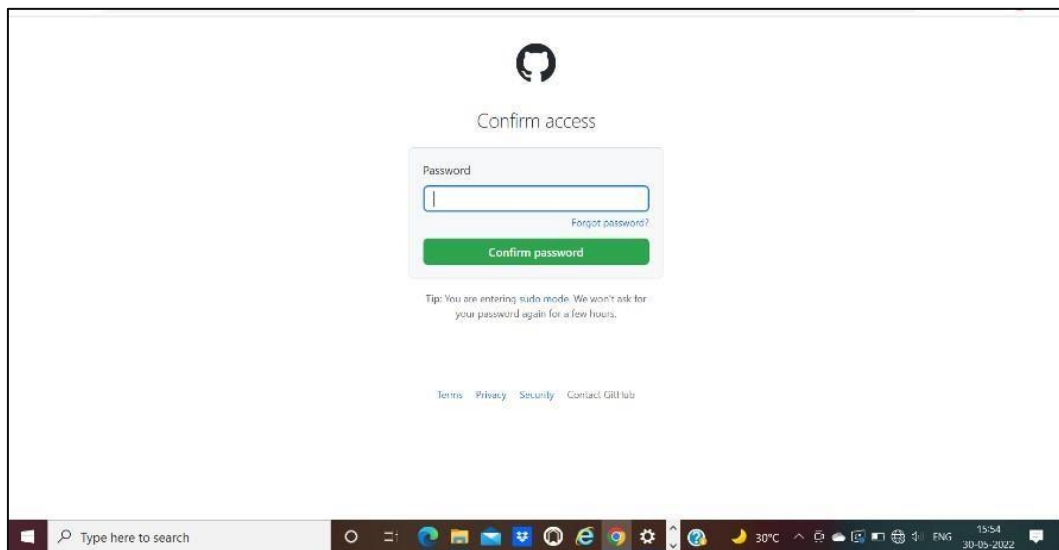
7) To add members to your repository open your repository and select settings option in the navigation bar.



8) Click on Collaborators option under the access tab.







- 9) 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.

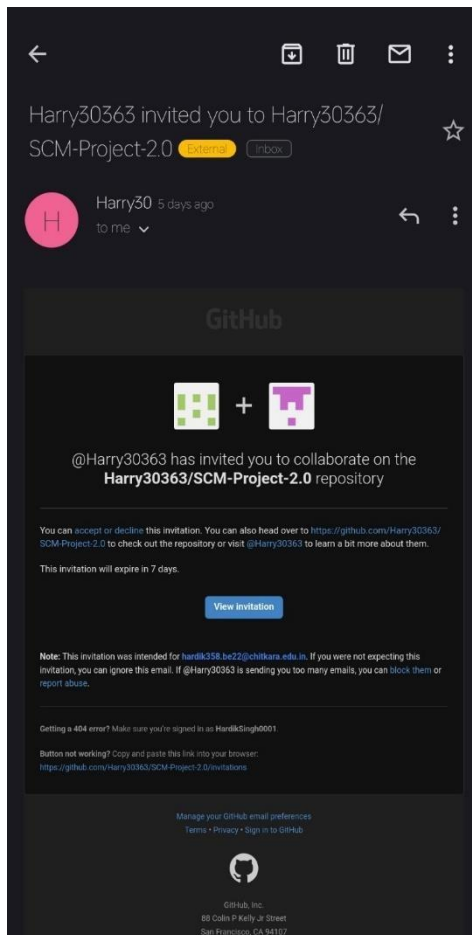
Manage access

Add people

<input type="checkbox"/> Select all	Type ▾
<input type="text" value="Find a collaborator..."/>	
<input type="checkbox"/>  HardikSingh0001 Awaiting HardikSingh0001's response	Pending Invite  Remove
<input type="checkbox"/>  harinder361 Awaiting harinder361's response	Pending Invite  Remove

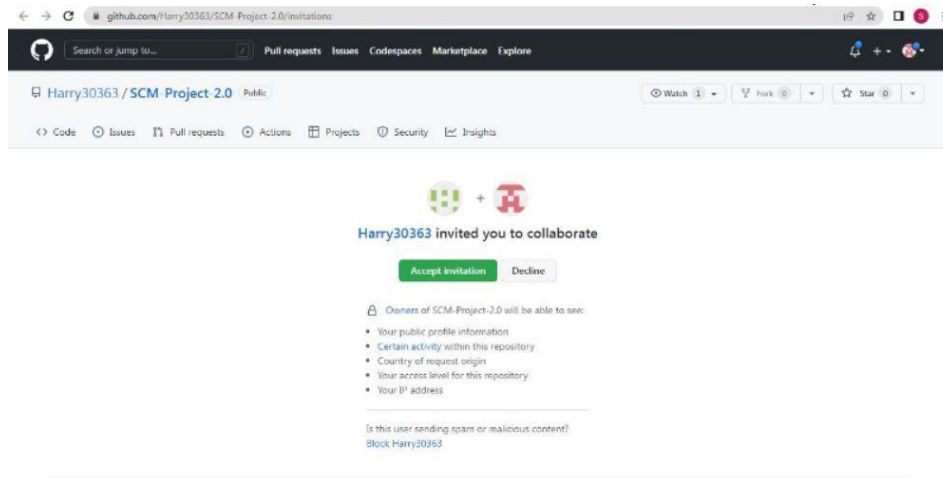
< Previous Next >

- 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.



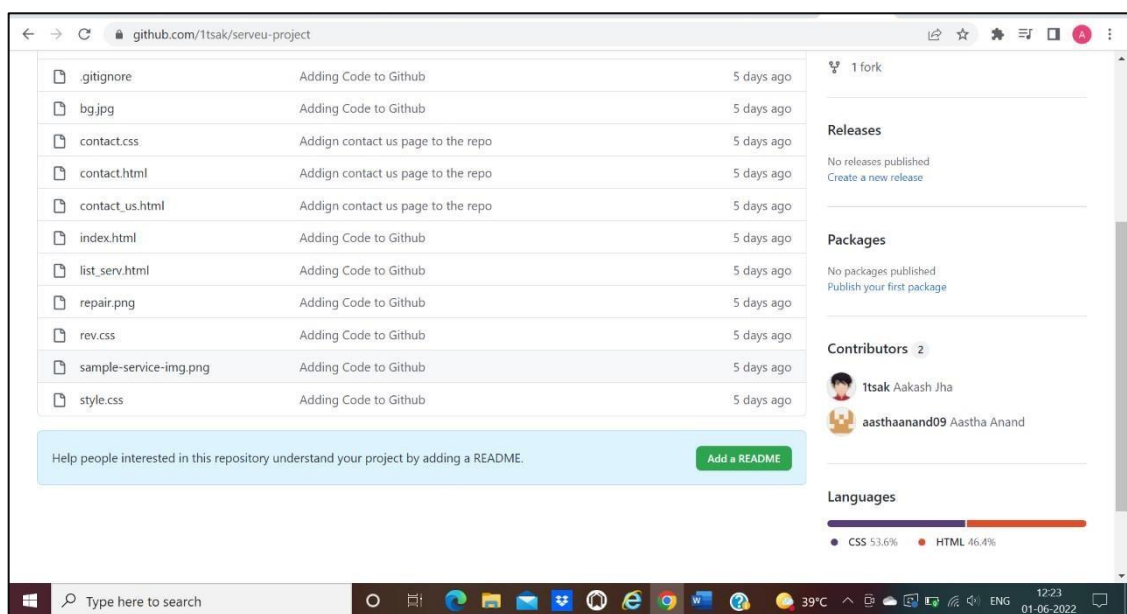
14) You will receive an invitation mail from the repository owner. Open the email and click on accept invitation.

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 2

Aim: Open and Close a Pull Request

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

```
MINGW64:/d/DESKTOP/SCM P/SCM-Project-2.0

Harjot singh@LAPTOP-HARJOT30 MINGW64 /d/DESKTOP/SCM P/SCM-Project-2.0
$ echo "# SCM-Project-2.0" >> README.md

Harjot singh@LAPTOP-HARJOT30 MINGW64 /d/DESKTOP/SCM P/SCM-Project-2.0
$ git init
Initialized empty Git repository in D:/DESKTOP/SCM P/SCM-Project-2.0/.git/

Harjot singh@LAPTOP-HARJOT30 MINGW64 /d/DESKTOP/SCM P/SCM-Project-2.0 (master)
$ git add README.md
warning: in the working copy of 'README.md', LF will be replaced by CRLF the next time Git touches it

Harjot singh@LAPTOP-HARJOT30 MINGW64 /d/DESKTOP/SCM P/SCM-Project-2.0 (master)
$ git commit -m "first commit"
[master (root-commit) dc71e63] first commit
1 file changed, 1 insertion(+)
create mode 100644 README.md

Harjot singh@LAPTOP-HARJOT30 MINGW64 /d/DESKTOP/SCM P/SCM-Project-2.0 (master)
$ git branch -M main

Harjot singh@LAPTOP-HARJOT30 MINGW64 /d/DESKTOP/SCM P/SCM-Project-2.0 (main)
$ git remote add origin https://github.com/Harry30363/SCM-Project-2.0.git

Harjot singh@LAPTOP-HARJOT30 MINGW64 /d/DESKTOP/SCM P/SCM-Project-2.0 (main)
$ git push -u origin main
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 228 bytes | 76.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/Harry30363/SCM-Project-2.0.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.

Harjot singh@LAPTOP-HARJOT30 MINGW64 /d/DESKTOP/SCM P/SCM-Project-2.0 (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    PYTHON PROJECT 1.py
    PYTHON PROJECT 2.py
    PYTHON PROJECT 5.py
    python project 3.py
    python project 4.py
```

- 2) After making new branch we add a file to the branch or make changes in the existing file.


```

MINGW64:/d/DESKTOP/SCM P/SCM-Project-2.0

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

Harjot singh@LAPTOP-HARJOT30 MINGW64 /d/DESKTOP/SCM P/SCM-Project-2.0 (main)
$ git add .

Harjot singh@LAPTOP-HARJOT30 MINGW64 /d/DESKTOP/SCM P/SCM-Project-2.0 (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   PYTHON PROJECT 1.py
        new file:   PYTHON PROJECT 2.py
        new file:   PYTHON PROJECT 5.py
        new file:   python project 3.py
        new file:   python project 4.py

Harjot singh@LAPTOP-HARJOT30 MINGW64 /d/DESKTOP/SCM P/SCM-Project-2.0 (main)
$ git commit -m "Uploaded By Harjot Singh"
[main d16d3ef] Uploaded By Harjot Singh
5 files changed, 239 insertions(+)
create mode 100644 PYTHON PROJECT 1.py
create mode 100644 PYTHON PROJECT 2.py
create mode 100644 PYTHON PROJECT 5.py
create mode 100644 python project 3.py
create mode 100644 python project 4.py

Harjot singh@LAPTOP-HARJOT30 MINGW64 /d/DESKTOP/SCM P/SCM-Project-2.0 (main)
$ git pull
Already up to date.

Harjot singh@LAPTOP-HARJOT30 MINGW64 /d/DESKTOP/SCM P/SCM-Project-2.0 (main)
$ git push
Enumerating objects: 8, done.
Counting objects: 100% (8/8), done.
Delta compression using up to 8 threads
Compressing objects: 100% (7/7), done.
Writing objects: 100% (7/7), 2.82 KiB | 1.41 MiB/s, done.
Total 7 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/Harry30363/SCM-Project-2.0.git
dc71e63..d16d3ef  main -> main

Harjot singh@LAPTOP-HARJOT30 MINGW64 /d/DESKTOP/SCM P/SCM-Project-2.0 (main)
$ |

```

- 3) Add and commit the changes to the local repository.
- 4) Use git push origin *branchname* option to push the new branch to the main repository.

```

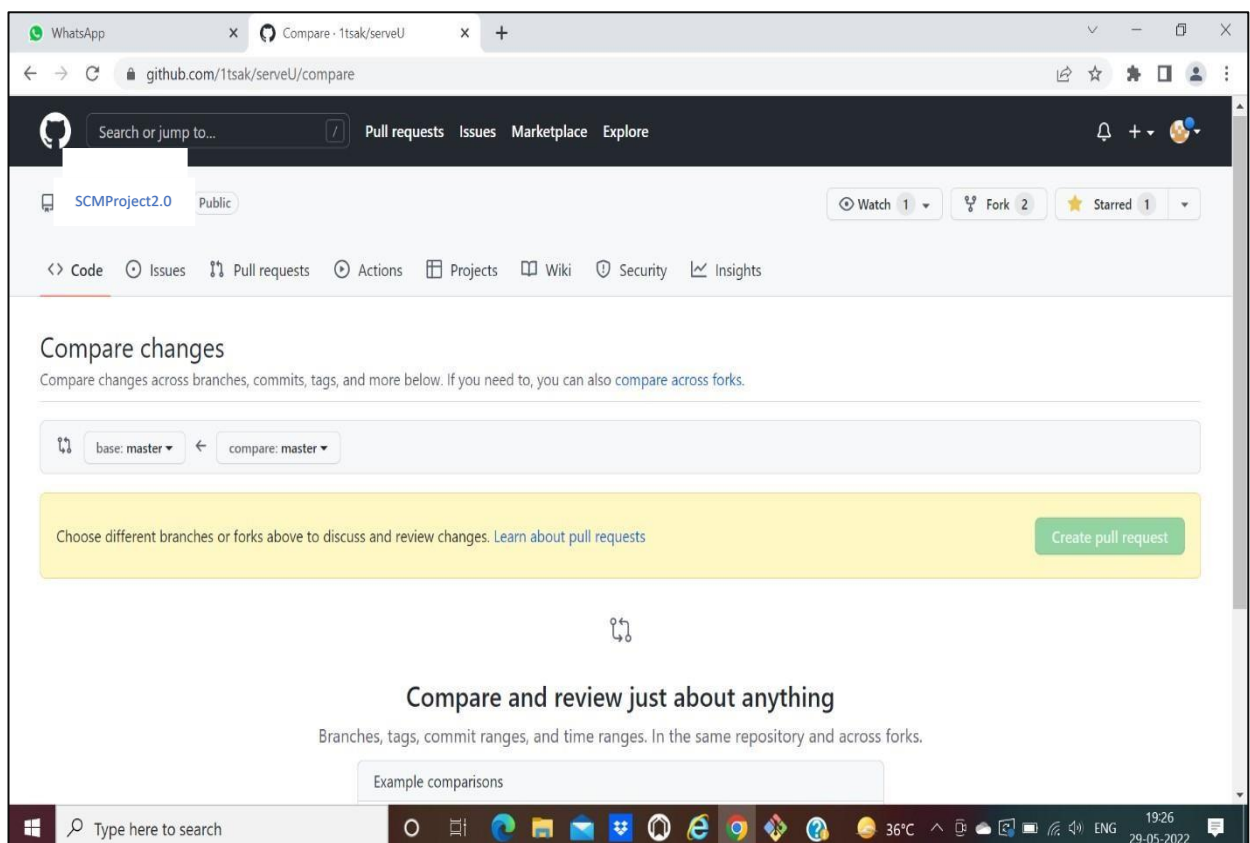
Harjot singh@LAPTOP-HARJOT30 MINGW64 /d/DESKTOP/SCM P/SCM-Project-2.0 (harjot)
$ git commit -m "Adding members name to the repo"
[harjot 70aaf07] Adding members name to the repo
1 file changed, 3 insertions(+)
create mode 100644 Names Of Team Members.txt

Harjot singh@LAPTOP-HARJOT30 MINGW64 /d/DESKTOP/SCM P/SCM-Project-2.0 (harjot)
$ git push origin harjot
Enumerating objects: 13, done.
Counting objects: 100% (13/13), done.
Delta compression using up to 8 threads
Compressing objects: 100% (11/11), done.
Writing objects: 100% (13/13), 3.29 KiB | 674.00 KiB/s, done.
Total 13 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), done.
remote:
remote: Create a pull request for 'harjot' on GitHub by visiting:
remote:   https://github.com/Harry30363/SCM-Project-2.0/pull/new/harjot
remote:
To https://github.com/Harry30363/SCM-Project-2.0.git
 * [new branch]      harjot -> harjot

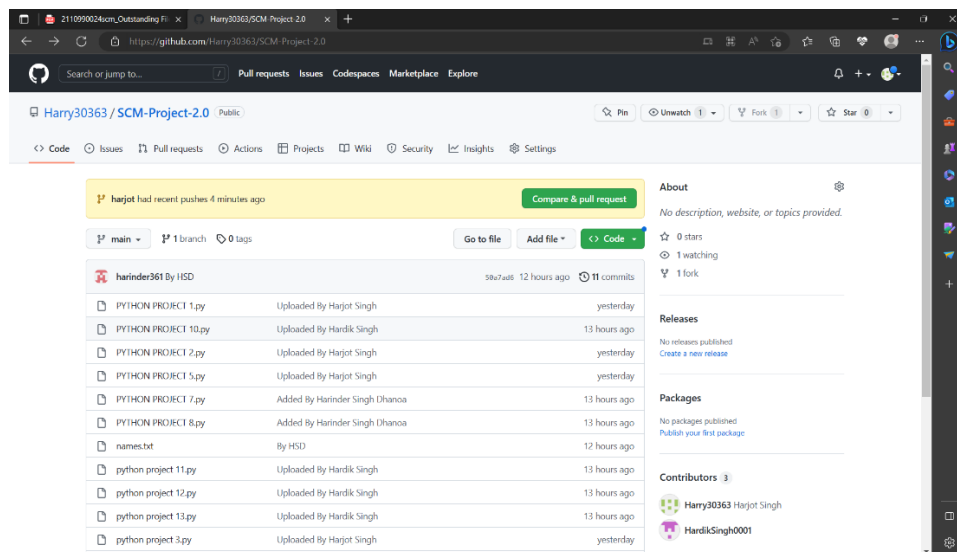
Harjot singh@LAPTOP-HARJOT30 MINGW64 /d/DESKTOP/SCM P/SCM-Project-2.0 (harjot)
$ |

```

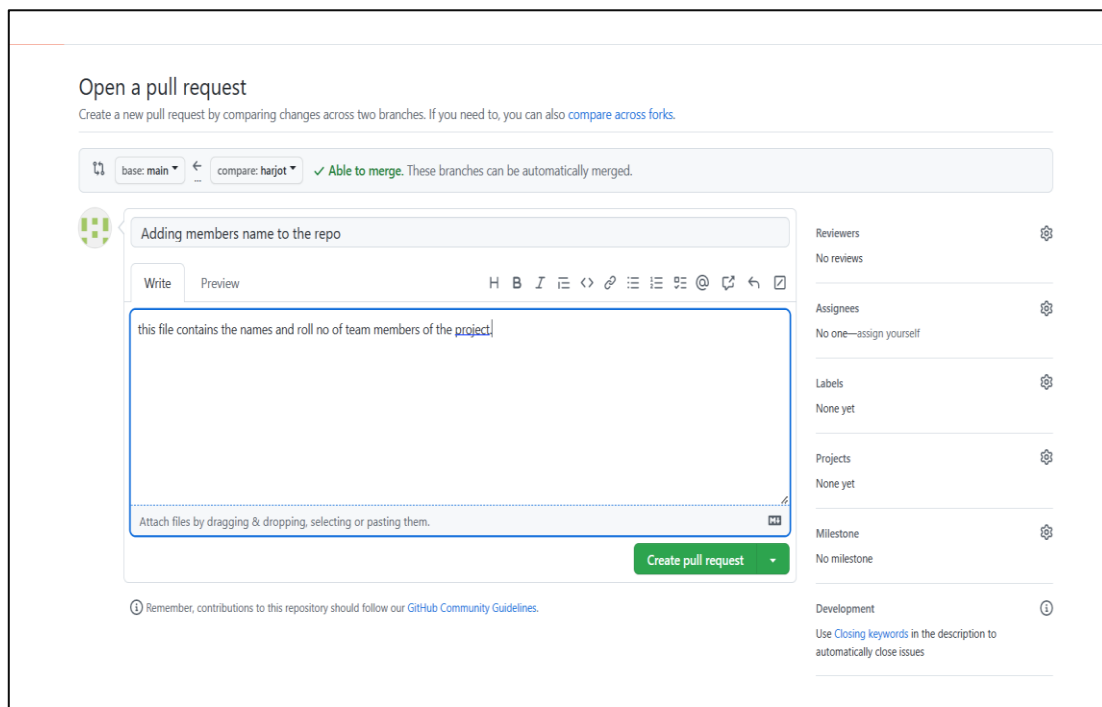
- 5) After pushing new branch GitHub will either automatically ask you to create a pull request or you can create your own pull request.



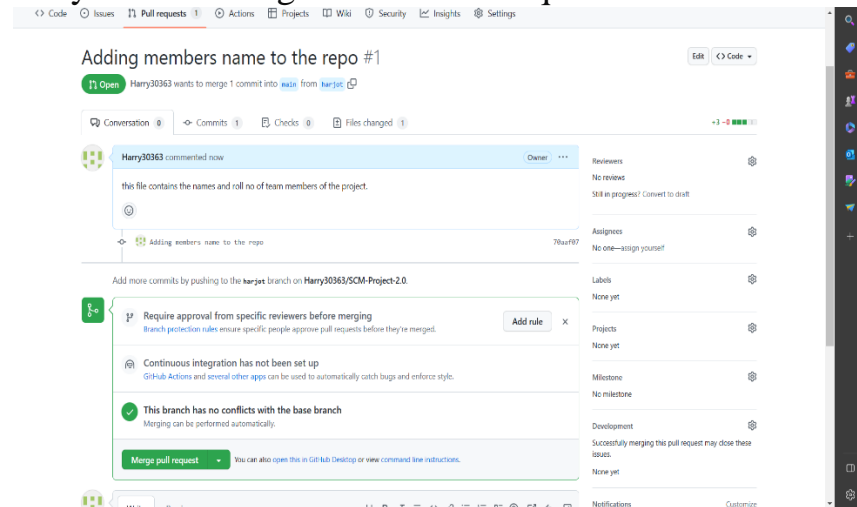
6) To create your own pull request, click on pull request option.



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



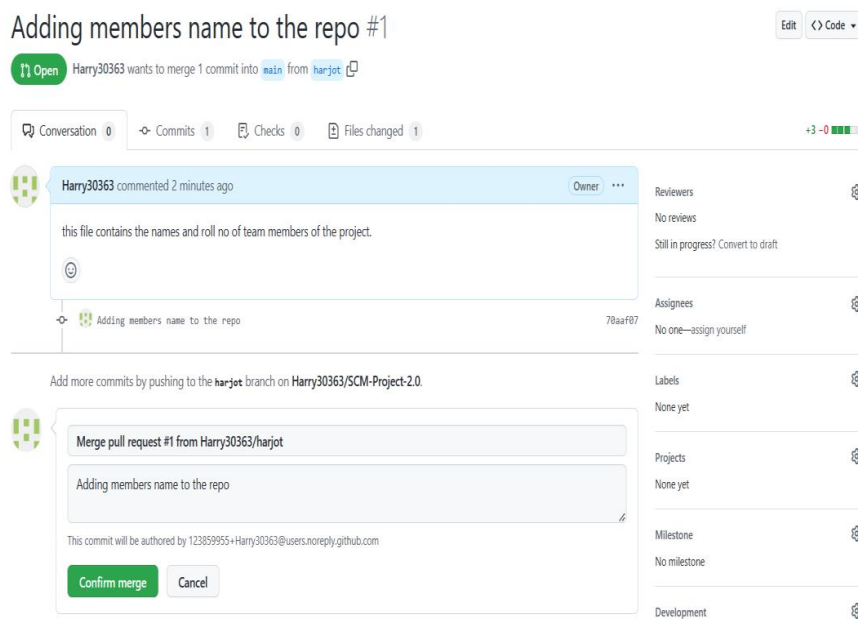
- 8) After opening a pull request all the team members will be sent the request if they want to merge or close the request.



- 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 add comment/ reason why you closed the pull request.

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



EXPERIMENT NO 3

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.

```
MINGW64/d/DESKTOP/SCM P/SCM-Project-2.0
Harjot singh@LAPTOP-HARJOT30 MINGW64 /d/DESKTOP/SCM P/SCM-Project-2.0 (main)
$ ls
'PYTHON PROJECT 1.py'* 'PYTHON PROJECT 5.py' 'python project 3.py'
'PYTHON PROJECT 2.py' README.md 'python project 4.py'

Harjot singh@LAPTOP-HARJOT30 MINGW64 /d/DESKTOP/SCM P/SCM-Project-2.0 (main)
$ git branch harjot

Harjot singh@LAPTOP-HARJOT30 MINGW64 /d/DESKTOP/SCM P/SCM-Project-2.0 (main)
$ git checkout harjot
Switched to branch 'harjot'

Harjot singh@LAPTOP-HARJOT30 MINGW64 /d/DESKTOP/SCM P/SCM-Project-2.0 (harjot)
$ git status
On branch harjot
nothing to commit, working tree clean

Harjot singh@LAPTOP-HARJOT30 MINGW64 /d/DESKTOP/SCM P/SCM-Project-2.0 (harjot)
$ git status
On branch harjot
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    Names Of Team Members.txt

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

Harjot singh@LAPTOP-HARJOT30 MINGW64 /d/DESKTOP/SCM P/SCM-Project-2.0 (harjot)
$
Harjot singh@LAPTOP-HARJOT30 MINGW64 /d/DESKTOP/SCM P/SCM-Project-2.0 (harjot)
$ git add .
Harjot singh@LAPTOP-HARJOT30 MINGW64 /d/DESKTOP/SCM P/SCM-Project-2.0 (harjot)
$ git status
```

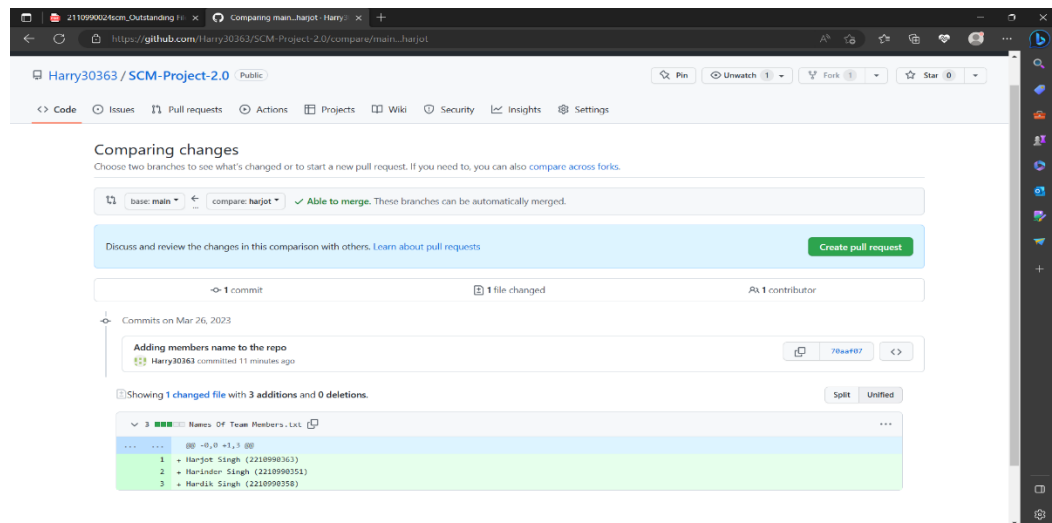
2. Push the modified branch using git push origin *branchname*.

```
Harjot singh@LAPTOP-HARJOT30 MINGW64 /d/DESKTOP/SCM P/SCM-Project-2.0 (harjot)
$ git commit -m "Adding members name to the repo"
[harjot 70aaf07] Adding members name to the repo
1 file changed, 3 insertions(+)
create mode 100644 Names Of Team Members.txt

Harjot singh@LAPTOP-HARJOT30 MINGW64 /d/DESKTOP/SCM P/SCM-Project-2.0 (harjot)
$ git push origin harjot
Enumerating objects: 13, done.
Counting objects: 100% (13/13), done.
Delta compression using up to 8 threads
Compressing objects: 100% (11/11), done.
Writing objects: 100% (13/13), 3.29 KiB | 674.00 KiB/s, done.
Total 13 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), done.
remote:
remote: Create a pull request for 'harjot' on GitHub by visiting:
remote:   https://github.com/Harry30363/SCM-Project-2.0/pull/new/harjot
remote:
To https://github.com/Harry30363/SCM-Project-2.0.git
 * [new branch]      harjot -> harjot

Harjot singh@LAPTOP-HARJOT30 MINGW64 /d/DESKTOP/SCM P/SCM-Project-2.0 (harjot)
$ |
```

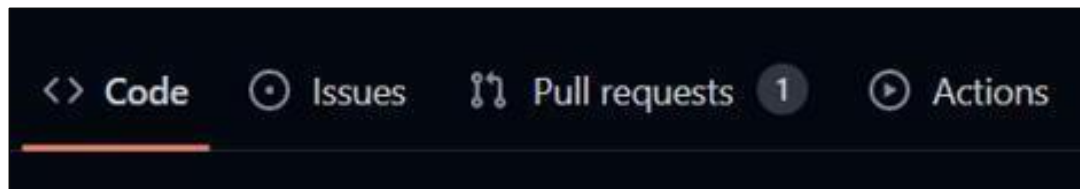
3. Open a pull request by following the procedure from the above experiment.



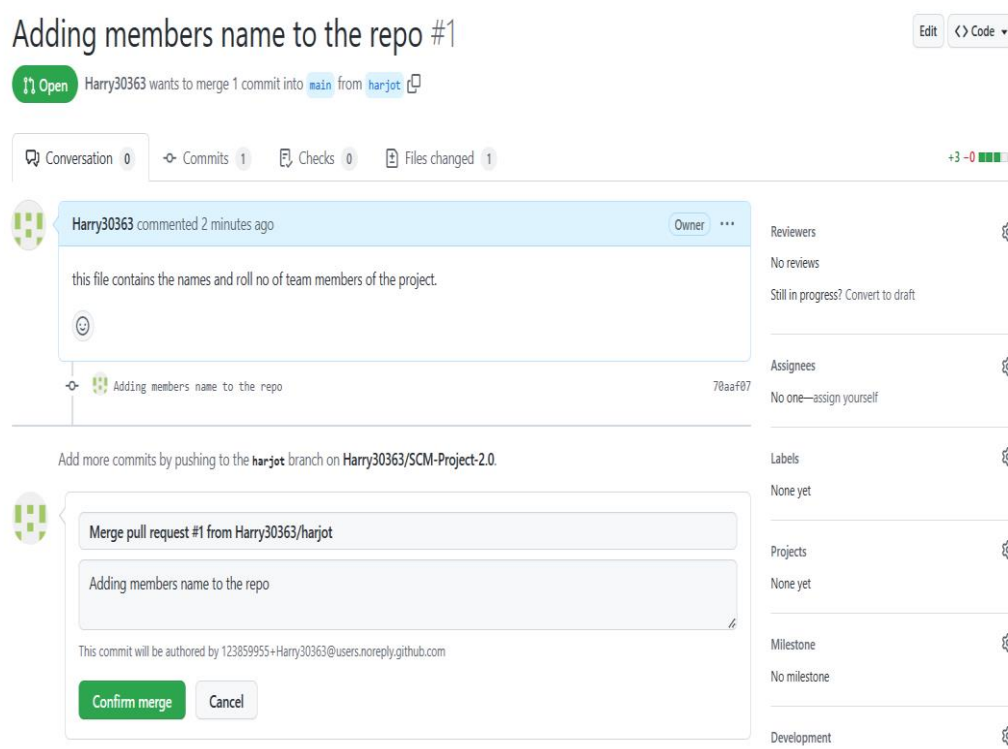
4. The pull request will be created and will be visible to all the team members.

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

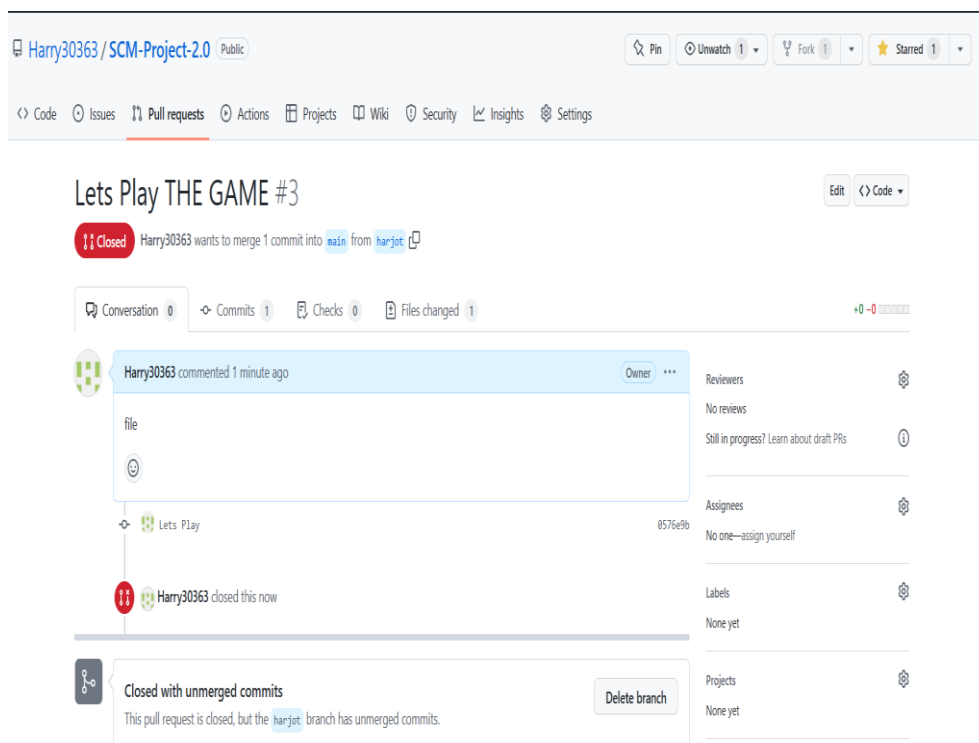
6. They will notice a new notification in the pull request menu.



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 4

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.

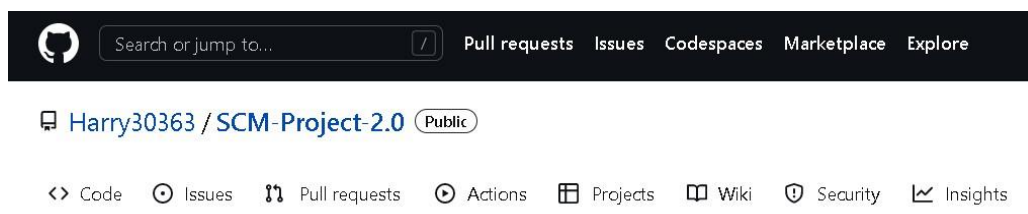
A repository's graphs give you information on traffic, projects that depend on the repository, contributors and commits to the repository, and a repository's forks and network. If you maintain a repository, you can use this data to get a better understanding of who's using your repository and why they're 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

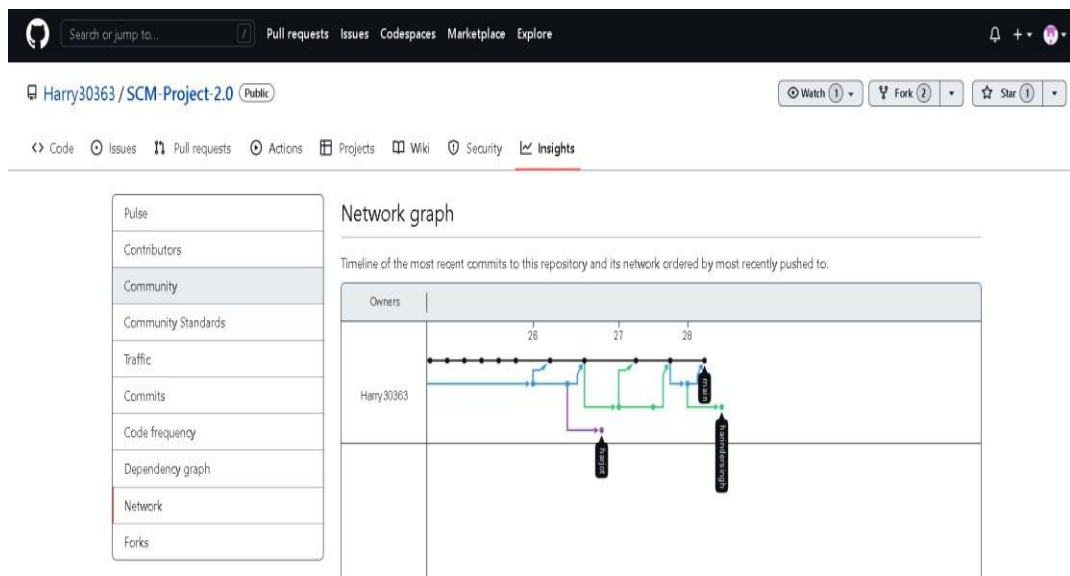
1. On GitHub.com, navigate to the main page of the repository.
2. Under your repository name, click **Insights**.



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

Pulse
Contributors
Community
Community Standards
Traffic
Commits
Code frequency
Dependency graph
Network
Forks

- 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.

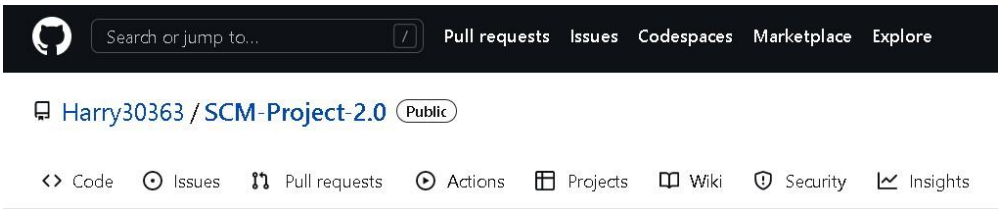


Listing the forks of a repository

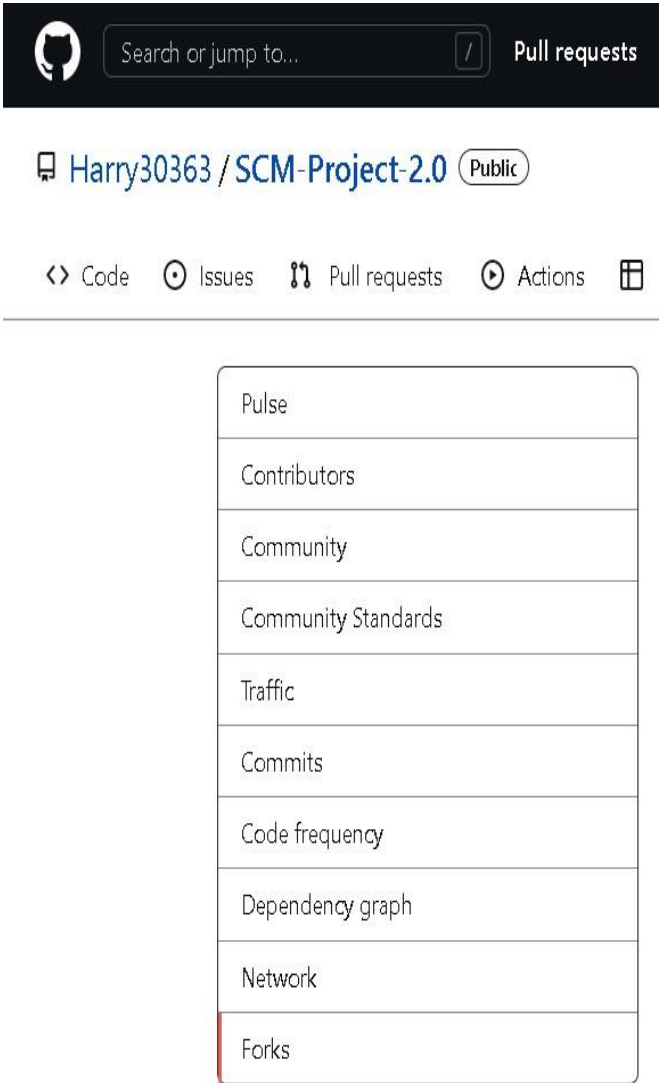
Forks are listed alphabetically by the username of the person who forked the repository

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

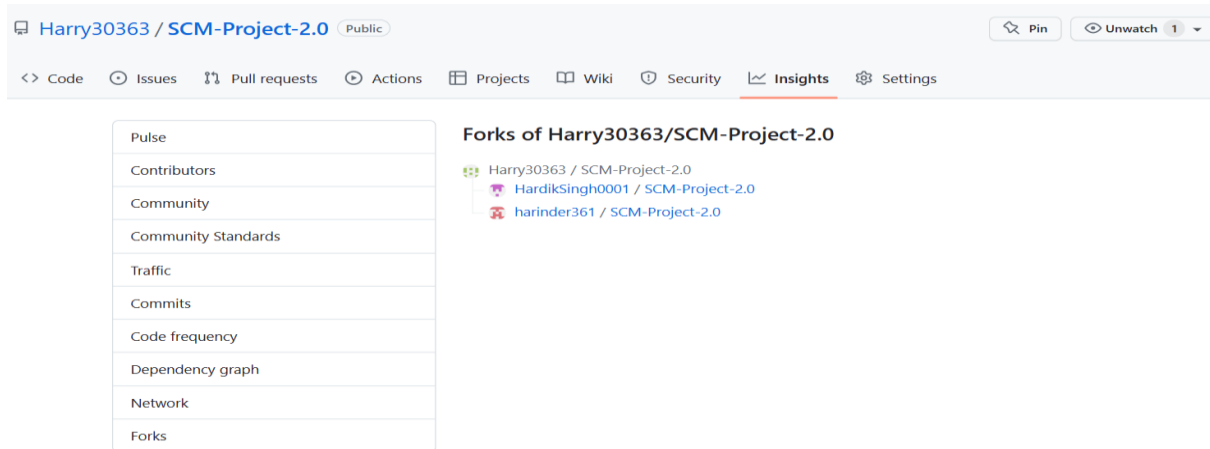
- 1. On GitHub.com, navigate to the main page of the repository.
- 2. Under your repository name, click **Insights**.



- 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.

