SOURCE CODE MANAGEMENT FILE

Submitted by:
Kartavya Tomar
2210990484
Group 24-B

Submission of:
Task 1.1

Submitted To:
Dr. Chetna Kaushal
Assistant Professor
Department of Computer Science & Engineering
Chitkara University Institute of Engineering and Technology
Rajpura, Punjab

Source Code Management File

Subject Name: **Source Code Management (SCM)**

Subject Code: **22CS003**

Vertical Name: First Year

Submitted By:

Kartavya Tomar

2210990484

G24-B

CHITKARA UNIVERSITY

INDEX

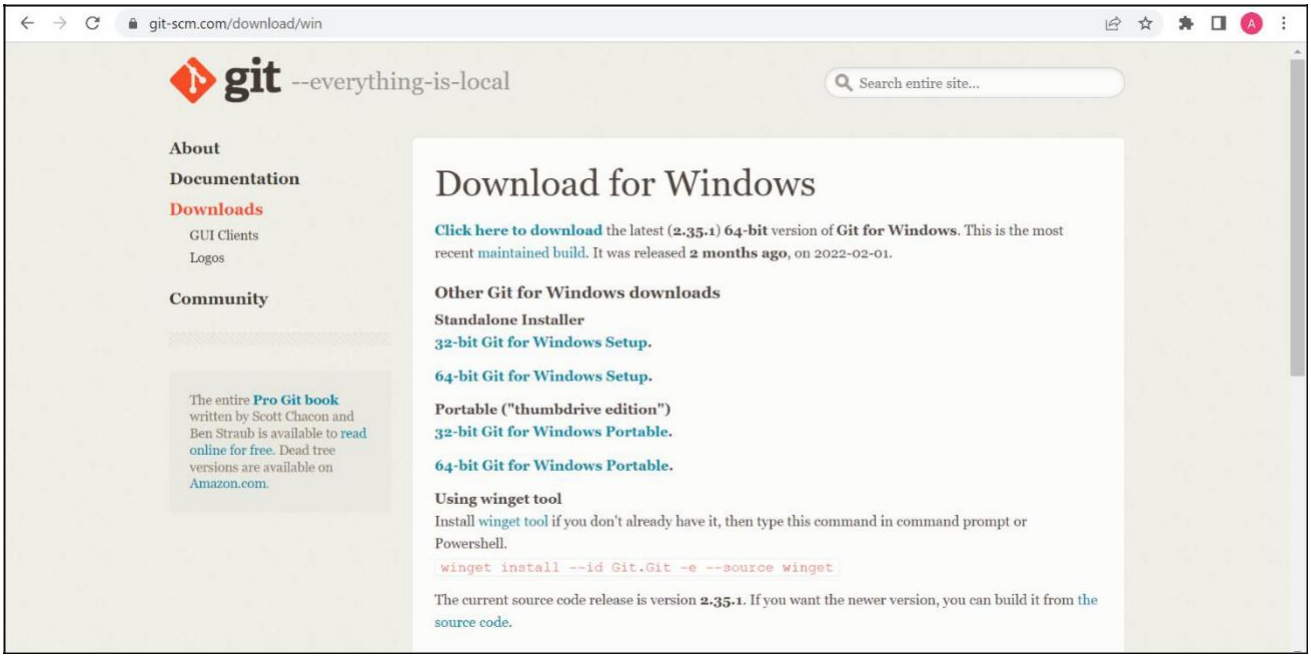
Experiment 1

Aim: Setting up of Git Client

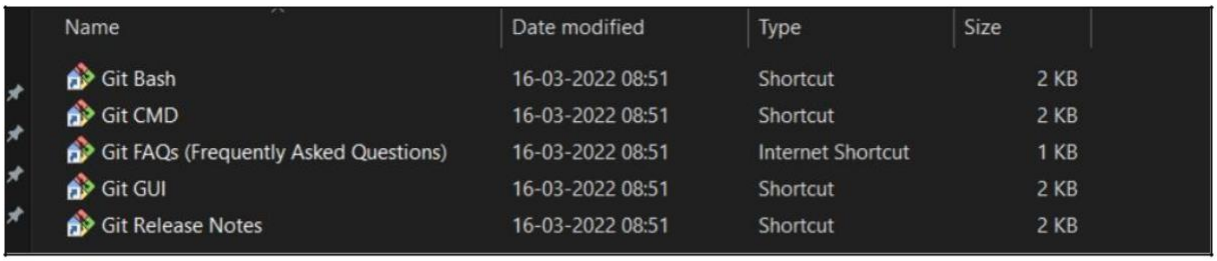
Theory:
GIT: It's a Version Control System (VCS). It is a software or we can say a server by which we are able to track all the previous changes in the code. It is basically used for pushing and pulling of code. We can use git and git-hub parallelly to work with multiple members or individually. We can make, edit, recreate, copy or download any code on git hub using git.

Procedure: We can install Git on Windows, using the most official build which is available for download on the GIT's official website or by just typing (scm git) on any search engine. We can go on https://git-scm.com/download/win and can select the platform and bit-version to download. And after clicking on your desired bit-version or iOS it will start downloading automatically.

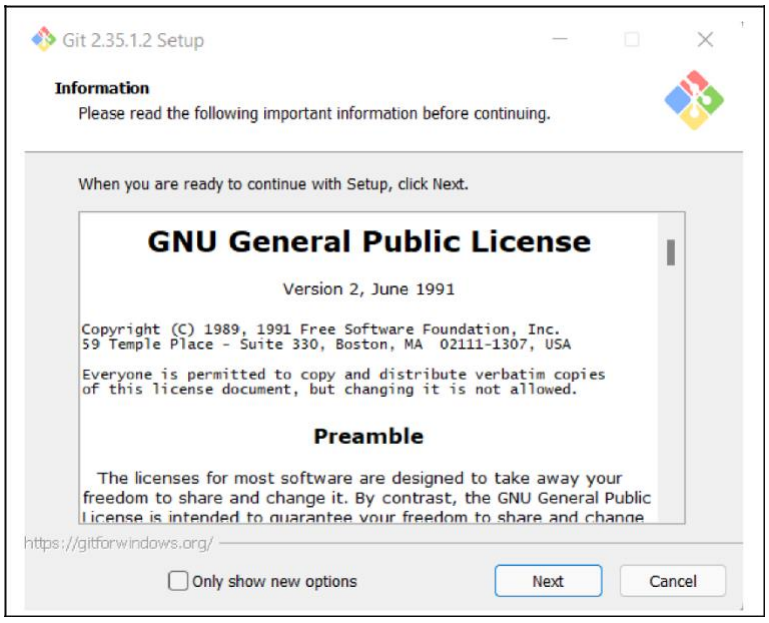
Snapshots of download:



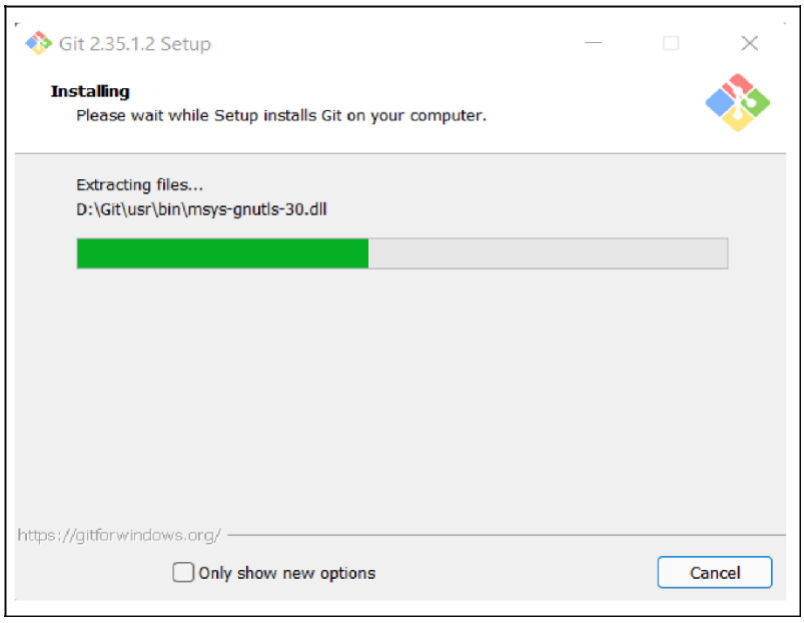
Opted for "64-bit Git for Windows Setup"



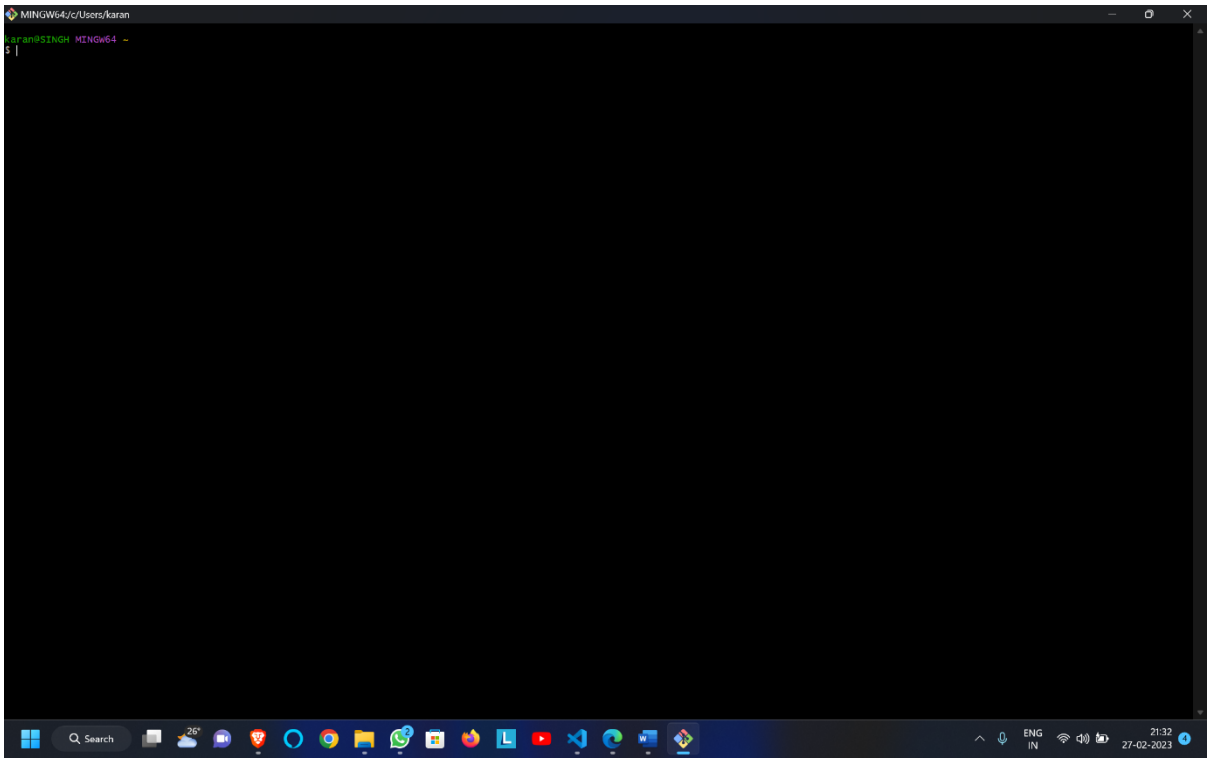
Git and its files in downloads



Git Setup



Git Installation



Git Bash launched

Experiment 2

Aim: Setting up GitHub Account

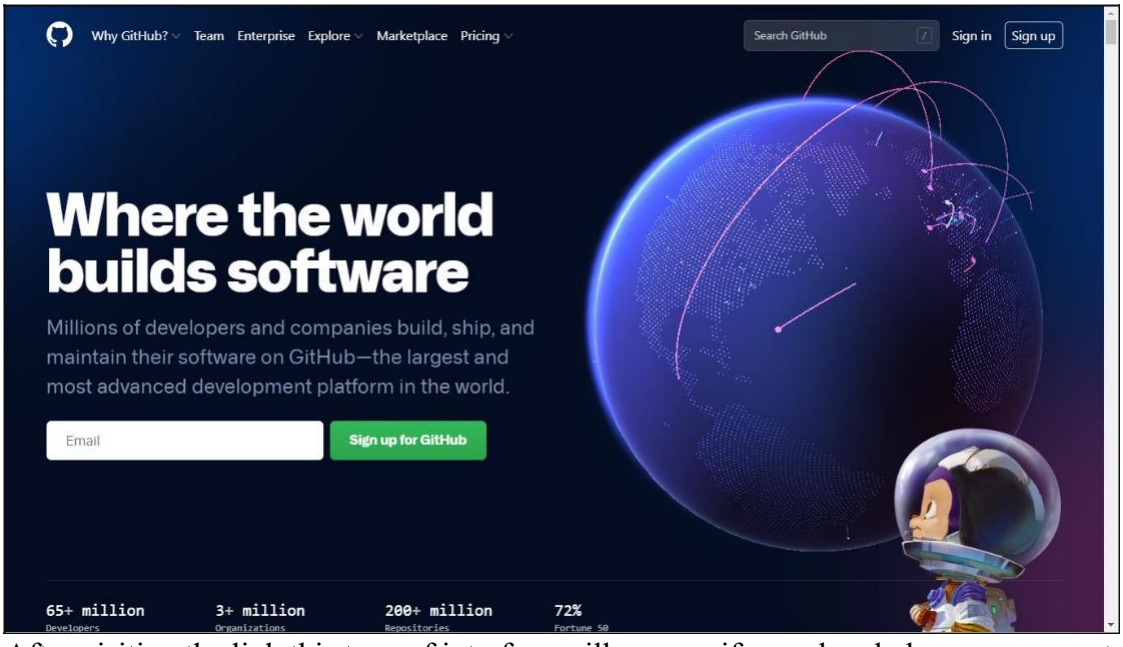
Theory:

GitHub: GitHub is a website and cloud-based service (client) that helps an individual or developers to store and manage their code. We can also track as well as control changes to our or public code.

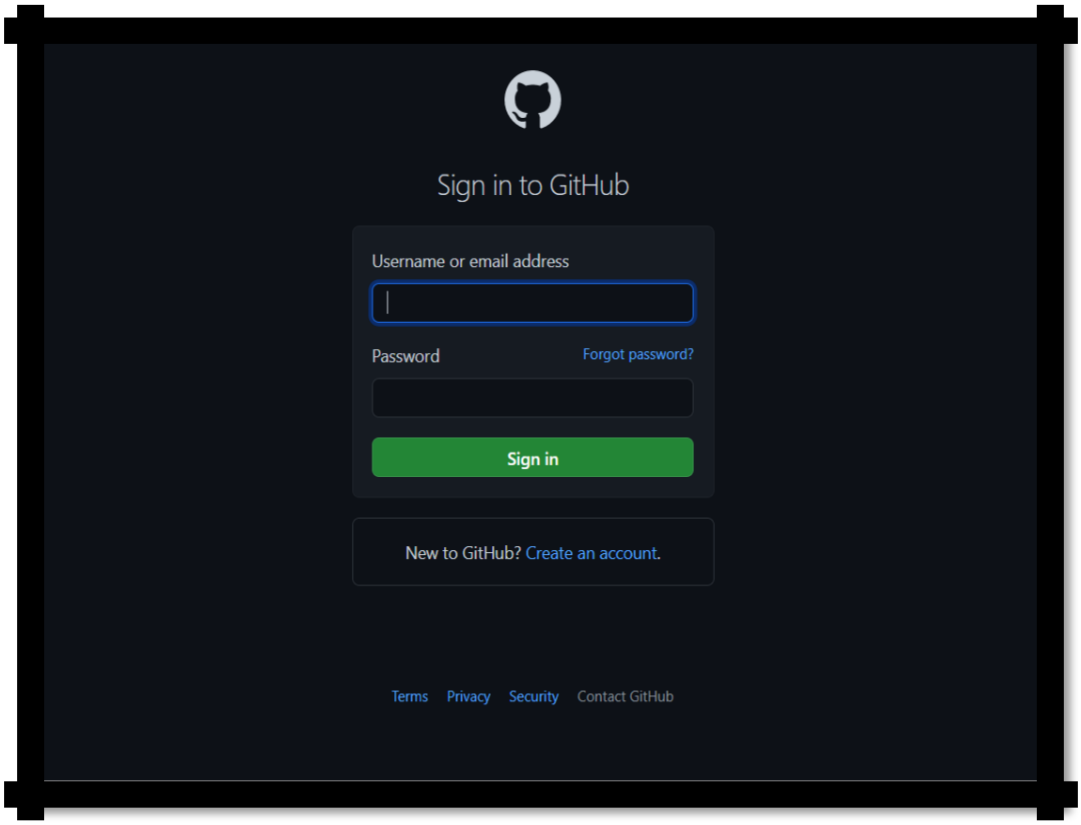
Advantages of GitHub: GitHub has a user-friendly interface and is easy to use. We can connect the git-hub and git but using some commands shown below in figure 1. Without GitHub we cannot use Git because it generally requires a host and if we are working for a project, we need to share it will our team members, which can only be done by making a repository. Additionally, anyone can sign up and host a public code repository for free, which makes GitHub especially popular with open-source projects.

Procedure:
To make an account on GitHub, we search for GitHub on our browser or visit https://github.com/signup. Then, we will enter our mail ID and create a username and password for a GitHub account.

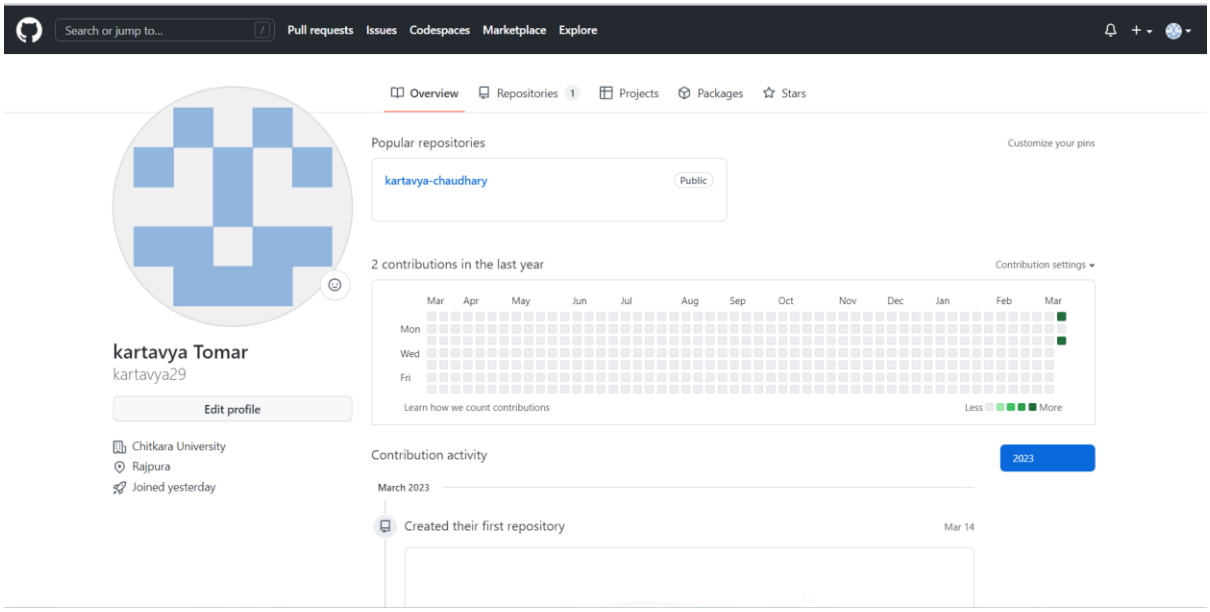
Snapshots –



After visiting the link this type of interface will appear, if you already have an account, you can sign in and if not, you can create.



GitHub Login



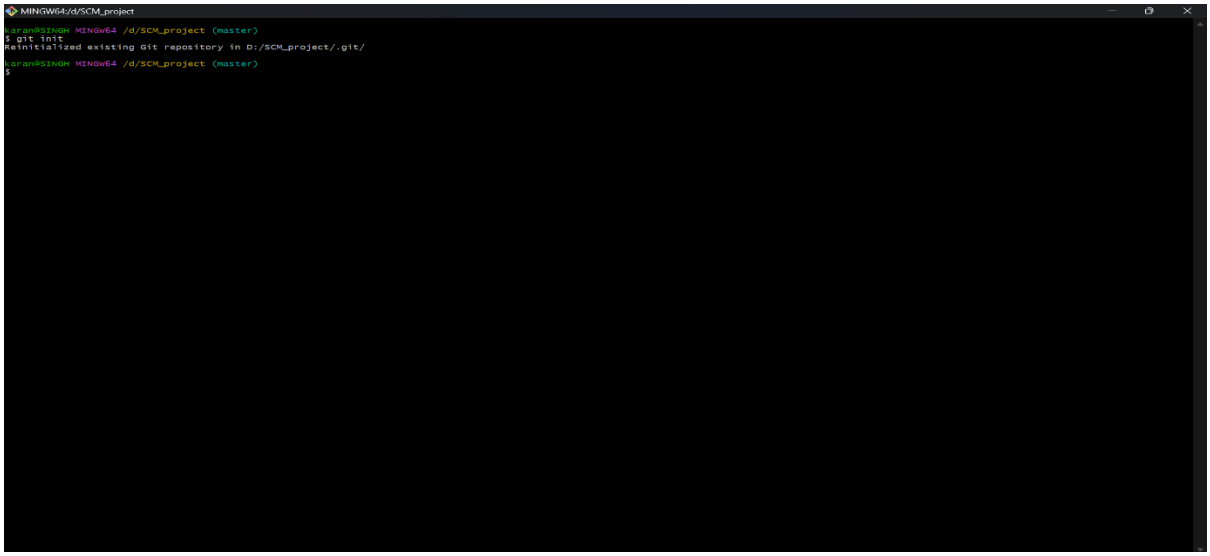
GitHub Interface

Experiment 3

Aim: Program to Generate log

Theory:
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 in it which we can see any time. Logs helps to check that what were the changes in the code or any other file and by whom. It also contains the number of insertions and deletions including at which time it was changed.

Procedure:
First of all, create a local repository using Git. For this, you have to make a folder in your device, right click and select "Git Bash Here". This opens the Git terminal. To create a new local repository, use the command "git init" and it creates a folder ". git".



When we use GIT for the first time, we have to give the user name and email so that if I am going to change in project, it will be visible to all.

For this, we use command:

"git config --global user.name Name"
"git config --global user.email email"

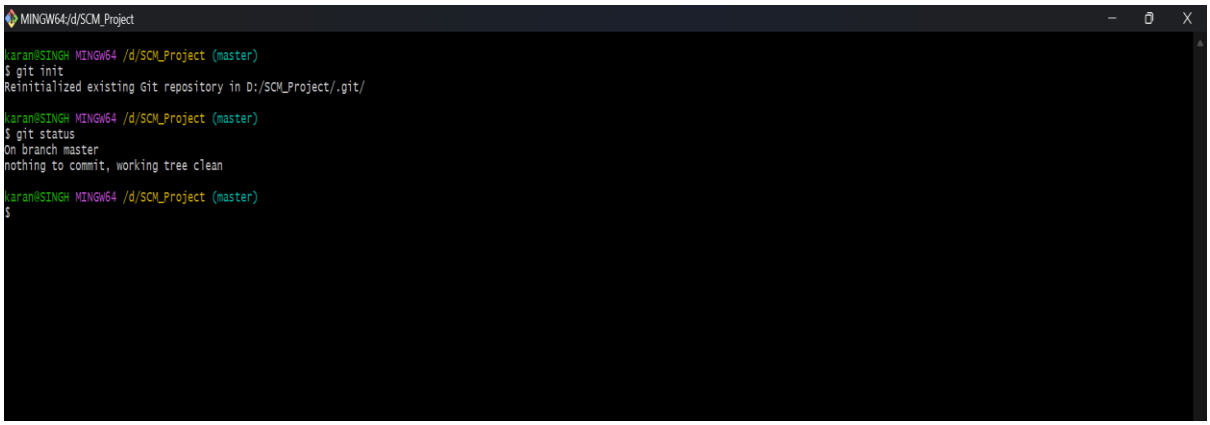
For verifying the user's name and email, we use:

"git config --global user.name"
"git config --global user.email"

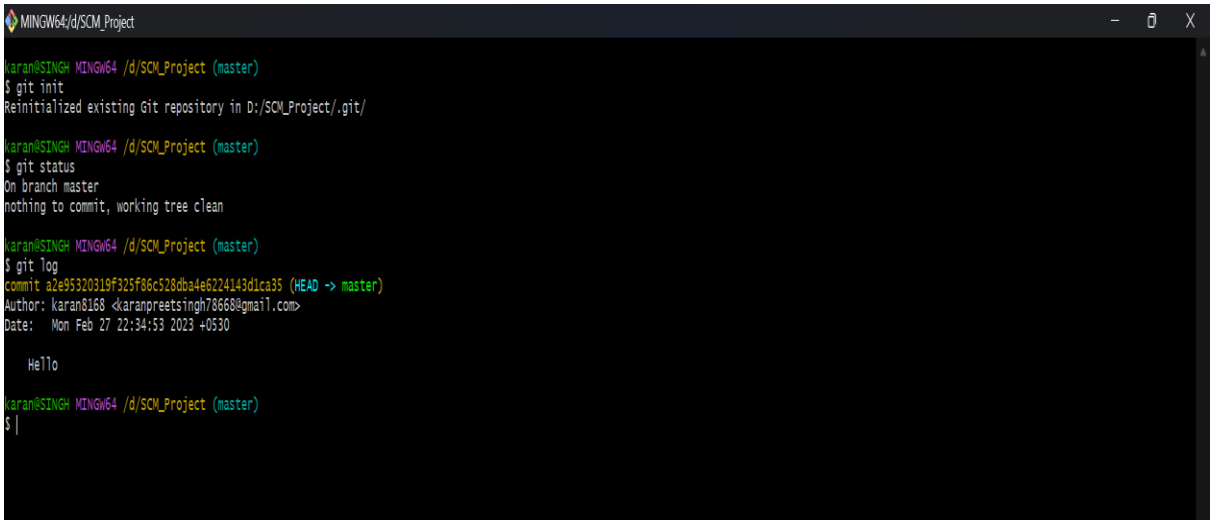
Some Important Commands

- ls → It gives the file names in the folder.
- ls -lart → Gives the hidden files also.
- git status → Displays the state of the working directory and the staged snapshot.
- touch filename → This command creates a new file in the repository.
- Clear → It clears the terminal.
- rm -rf .git → It removes the repository.
- git log → displays all of the commits in a repository's history
- git diff → It compares my working tree to staging area.

Git status:



Git log:



The git log command displays a record of the commits in a Git repository. By default, the git log command displays a commit hash, the commit message and other commit metadata.

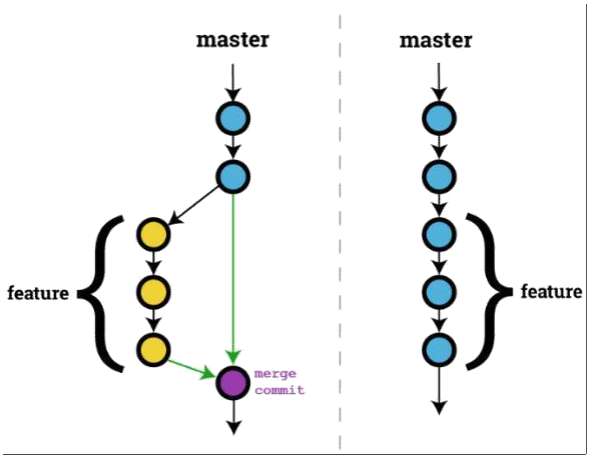
Experiment 4

Aim: Create and visualize branches

Theory:
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.

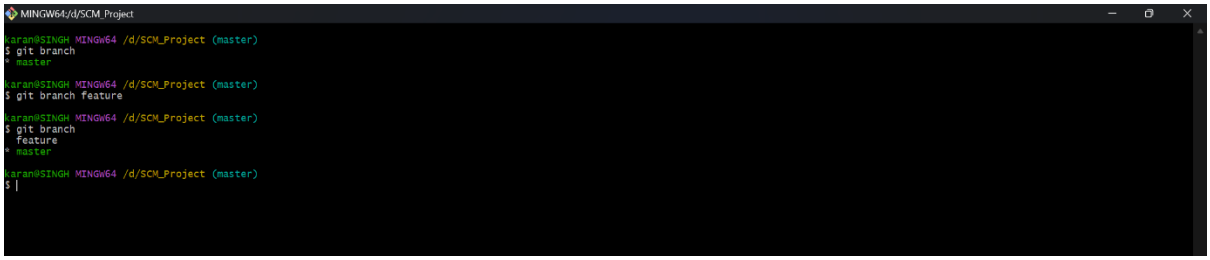
Create branches: The main branch in git is called as master branch. But we can make branches out of this main master branch. All the files present in master can be shown in branch but the file which are created in branch are not shown in master branch. We can also merge both the parent (master) and child (other branches).
Syntax:

For creating a new branch, git branch name by default is master branch.

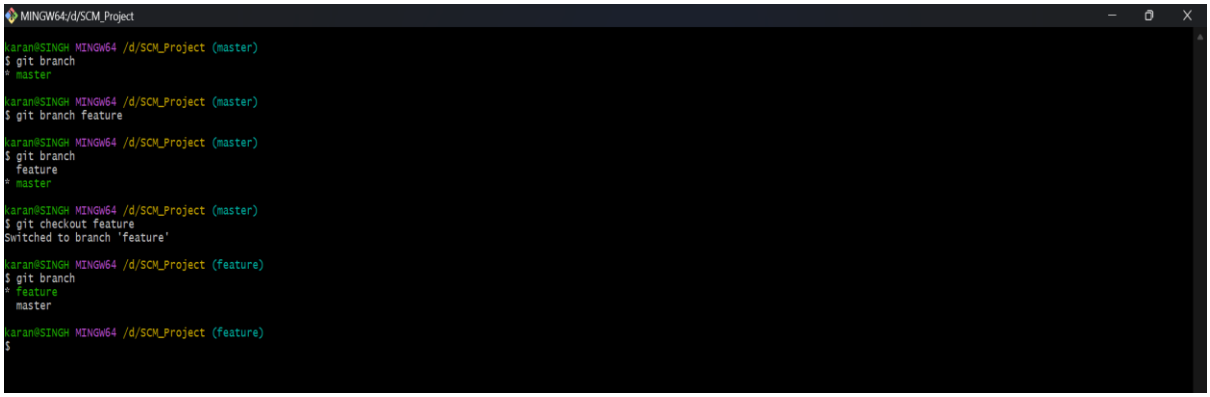


Snapshots –

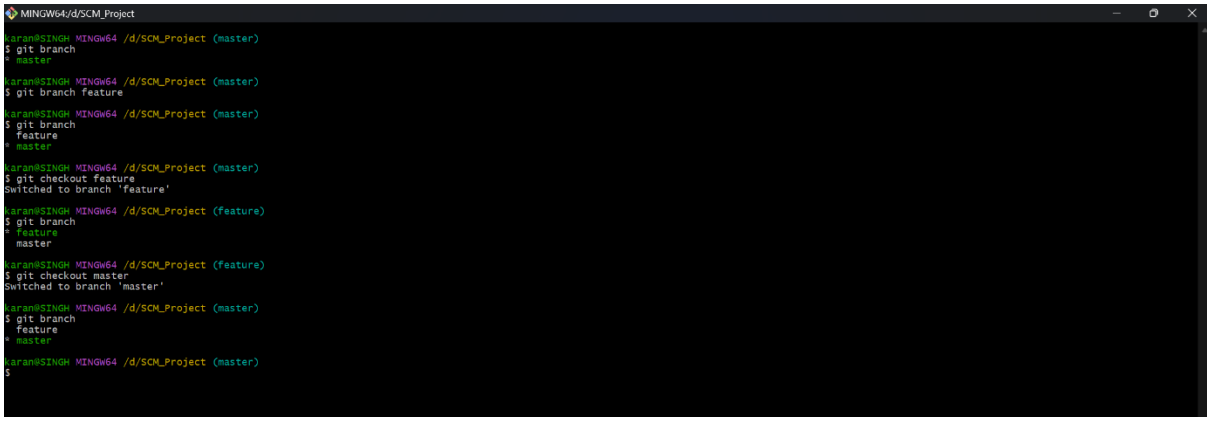
Default branch is master branch



Adding a feature branch



Switching to feature branch



Switching to master branch

Checking commits

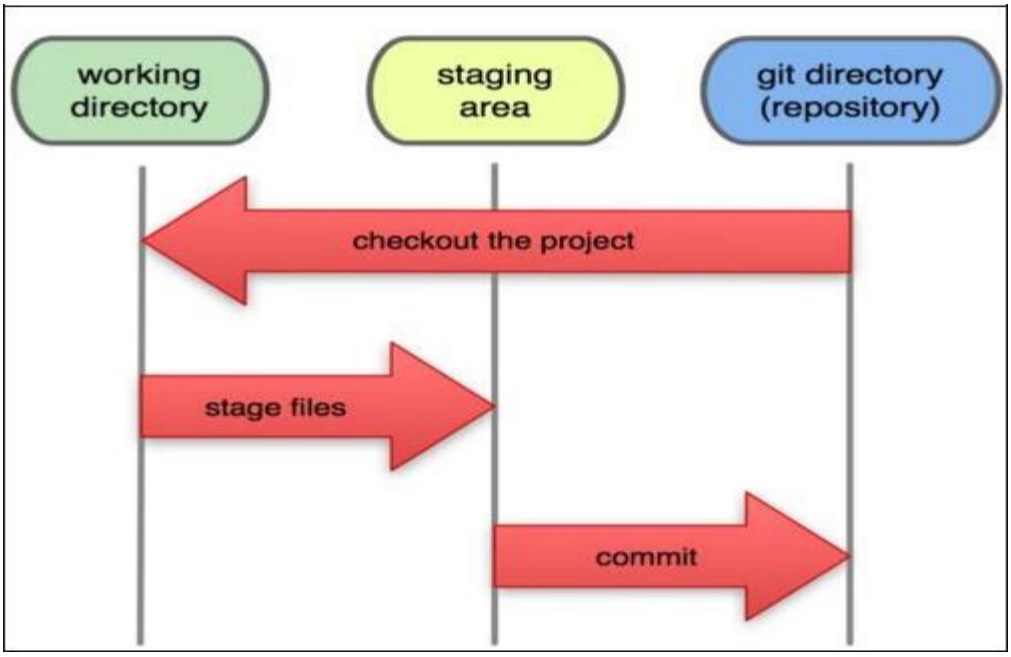
Experiment 5

Aim: Git lifecycle description

Theory:

Stages in GIT Life Cycle: Files in a Git project have various stages like Creation, Modification, Refactoring, and Deletion and so on. Irrespective of whether this project is tracked by Git or not, these phases are still prevalent. However, when a project is under Git version control system, they are present in three major Git states in addition to these basic ones. Here are the three Git states:

- Working directory
- Staging area
- Git directory

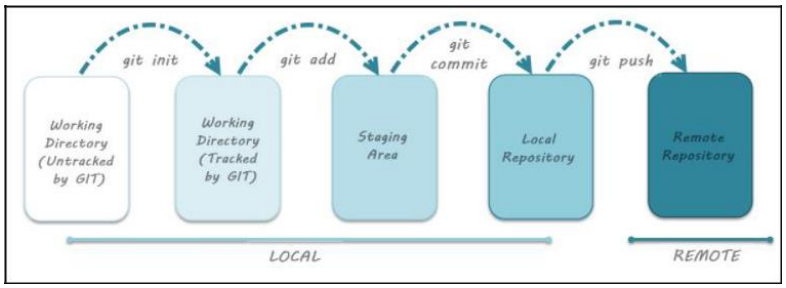


Working Directory:
Consider a project residing in your local system. This project may or may not be tracked by Git. In either case, this project directory is called your Working directory.

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: It 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.



Subject Name: **Source Code Management**

Subject Code: **22CS003**

Vertical Name: First Year

Department: **DCSE**

Submitted By:
Kartavya Tomar
2210990484
G24-B
Submission of Task 1.2

Submitted To:
Dr. Chetna Kaushal
Assistant Professor
Department of Computer Science & Engineering
Chitkara University Institute of Engineering and Technology
Rajpura, Punjab

 # Index #

Experiment 1

Aim: Add collaborators on GitHub Repo

Theory:

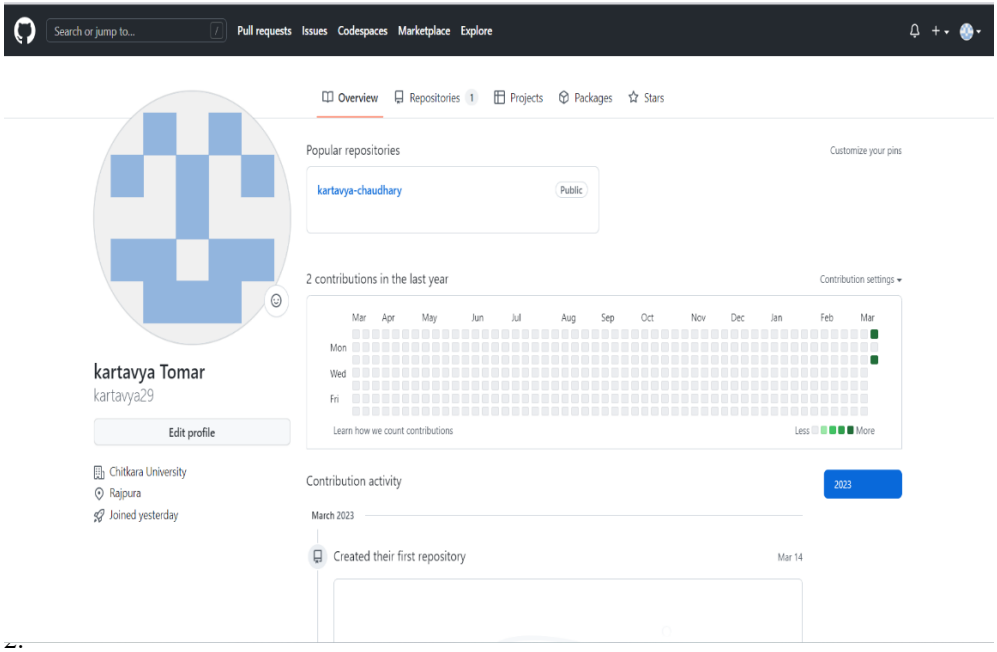
Whenever you make a repository in GitHub, not everyone has the permission to change or push codes into your repository. The users have a read-only access. In order to allow other individuals to make changes to your repository, you need to invite them to collaborate to the project.

GitHub also restricts the number of collaborators we can invite within a period of 24 hours. If we exceed the limit, then either we have to wait for 24-hours or we can also create an organization to collaborate with more people.

Being a collaborator, the user can create, merge and close pull requests in the repository. They can also remove them as the collaborator.

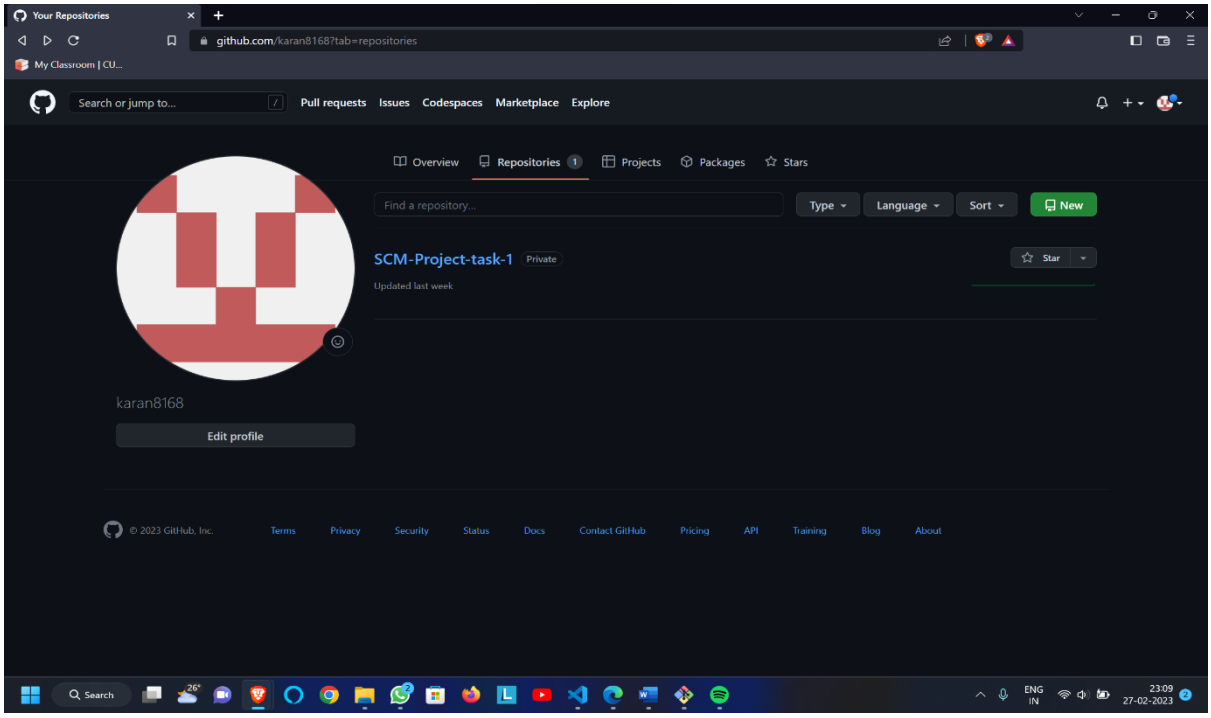
Procedure:

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

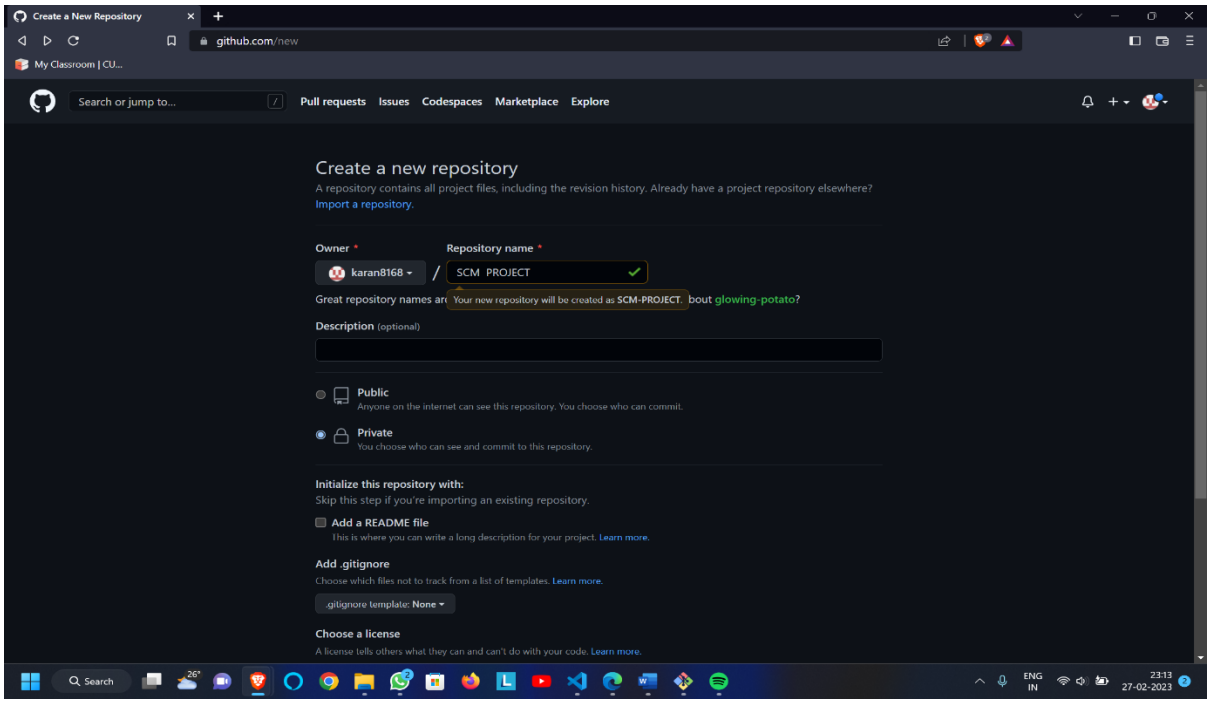


2.

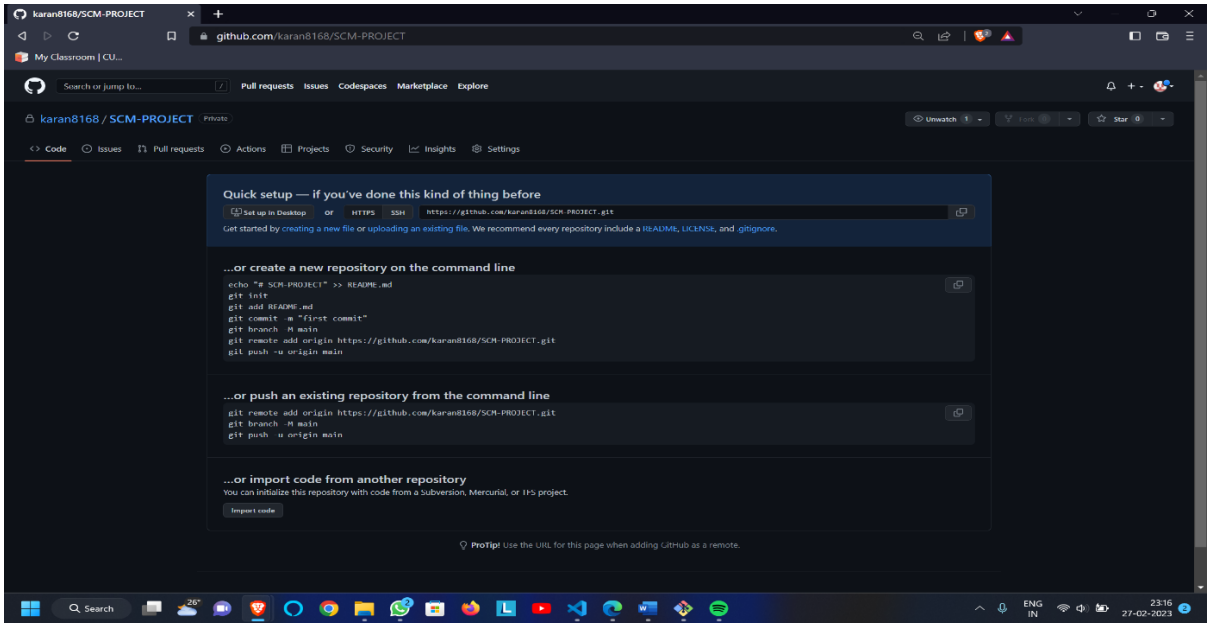
2. Click on the ‘New’ button in the top right corner.



3. Enter the Repository name and add the description of the repository.
4. Select if you want the repository to be public or private.



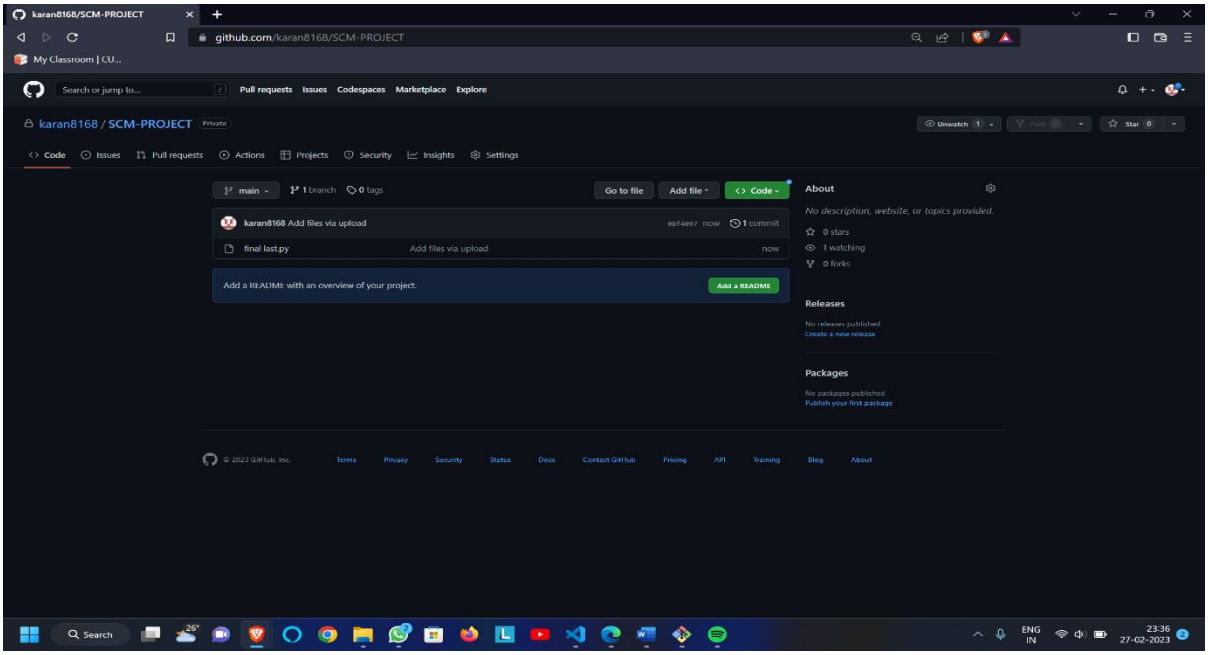
5. If you want to import code from an existing repository select the import code option.



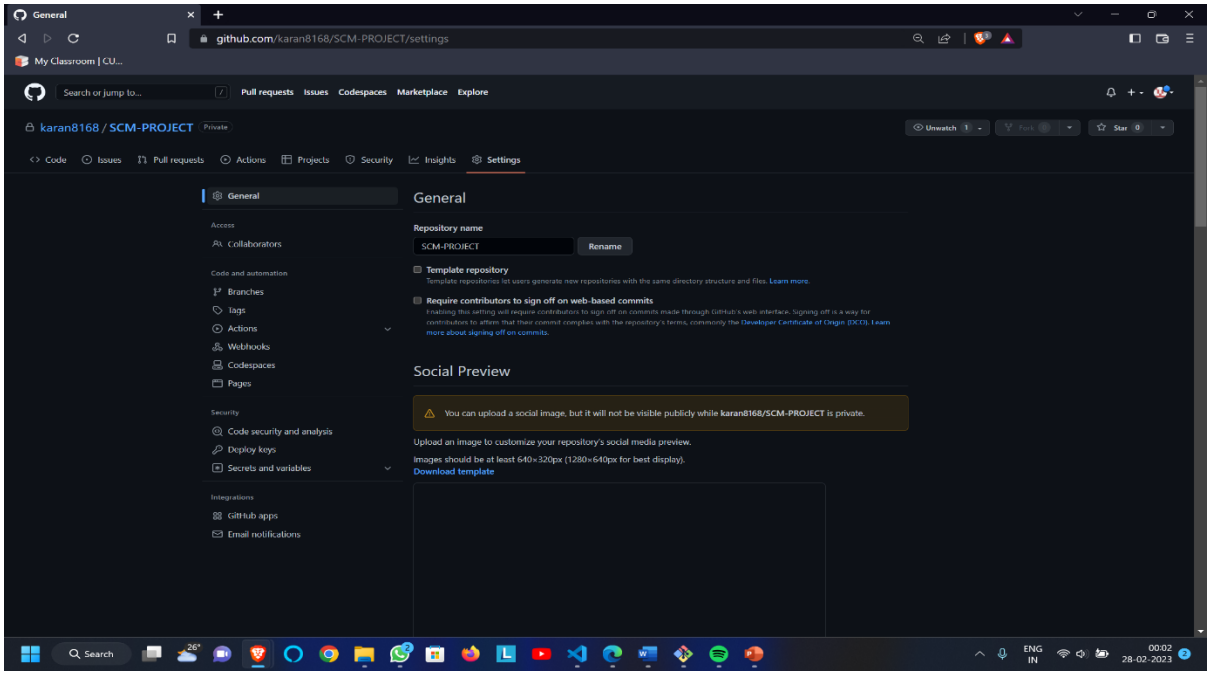
6. Now, you have created your repository successfully.

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

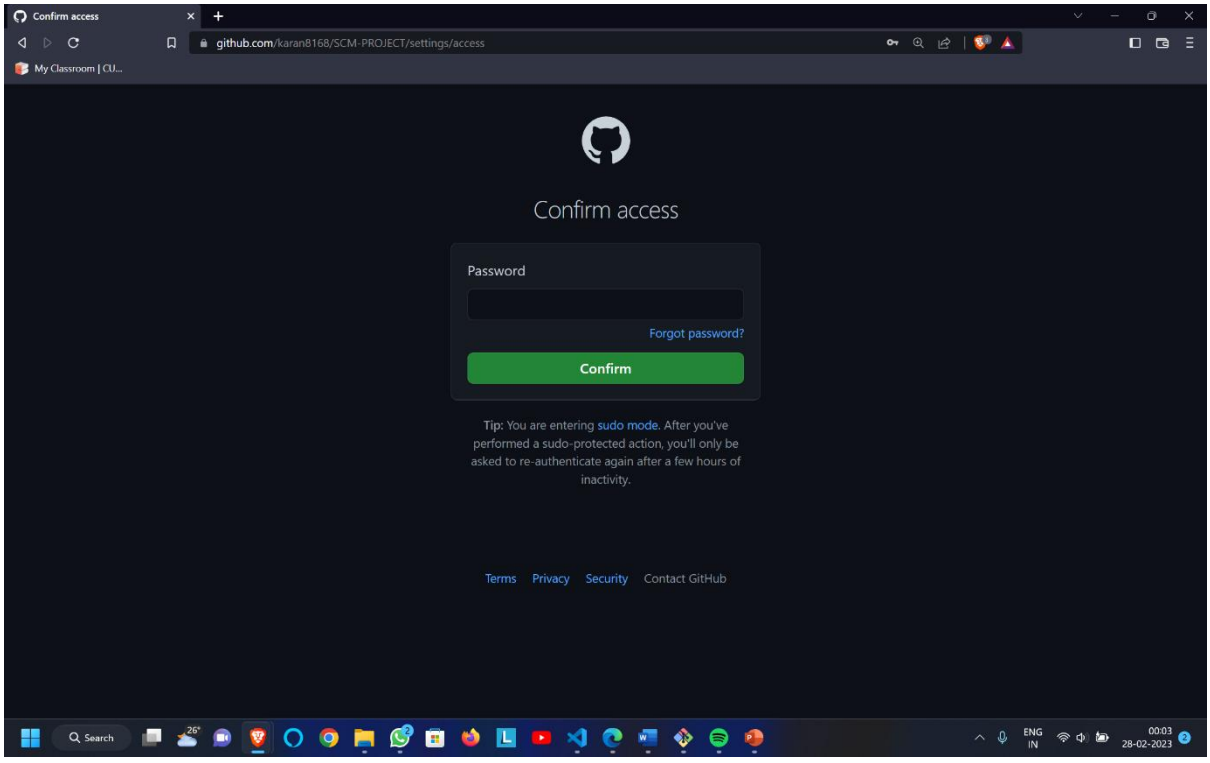
8.



8. Click on Collaborators option under the access tab.

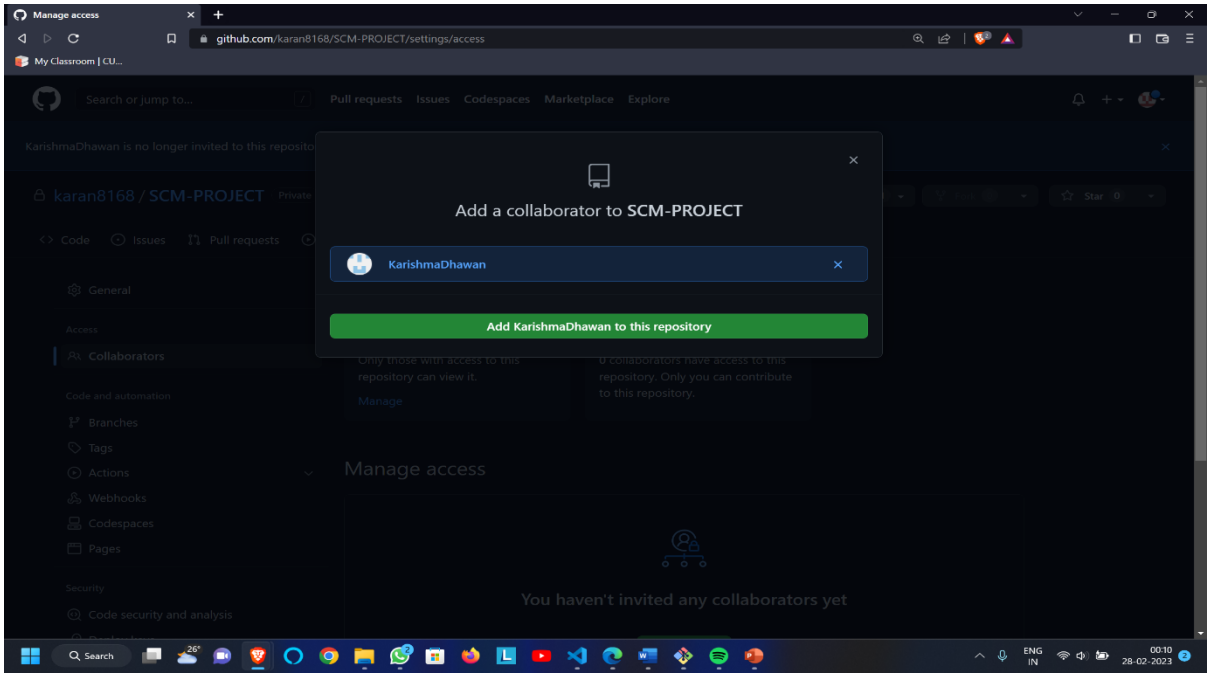


9. After clicking on collaborators, GitHub asks you to enter your password to confirm the access to the repository.



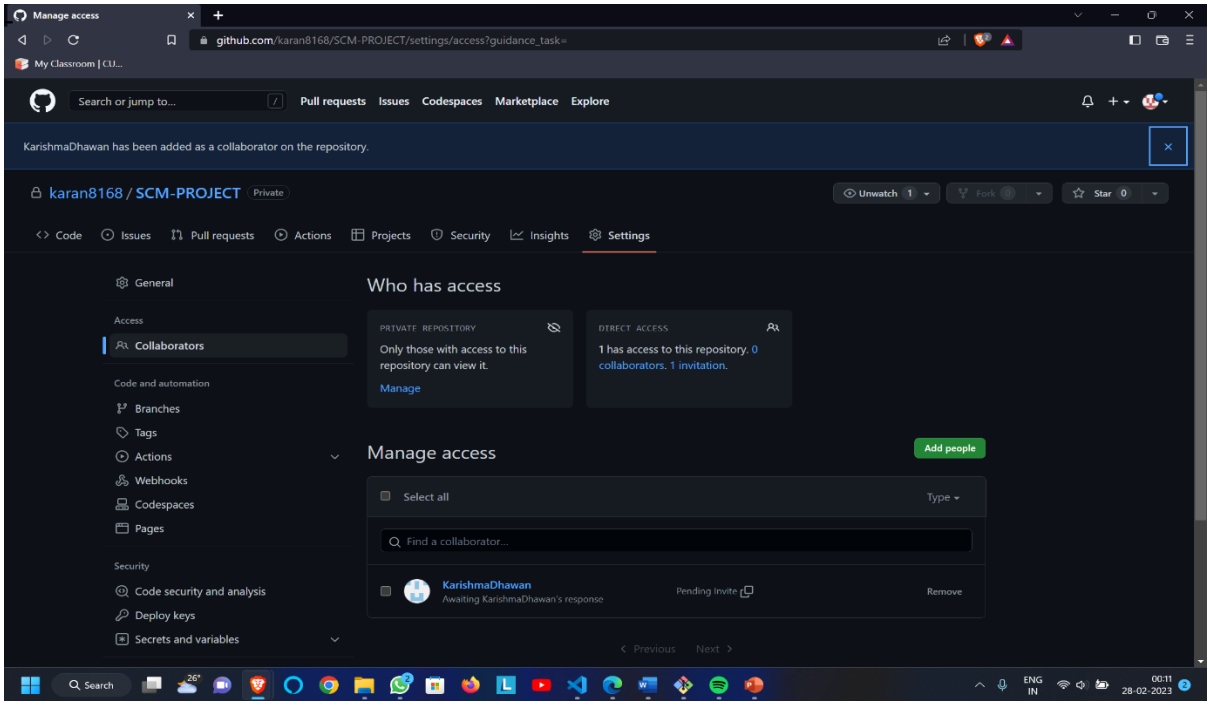
10. After entering the password, you can manage access and add/remove team members to your project.

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



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

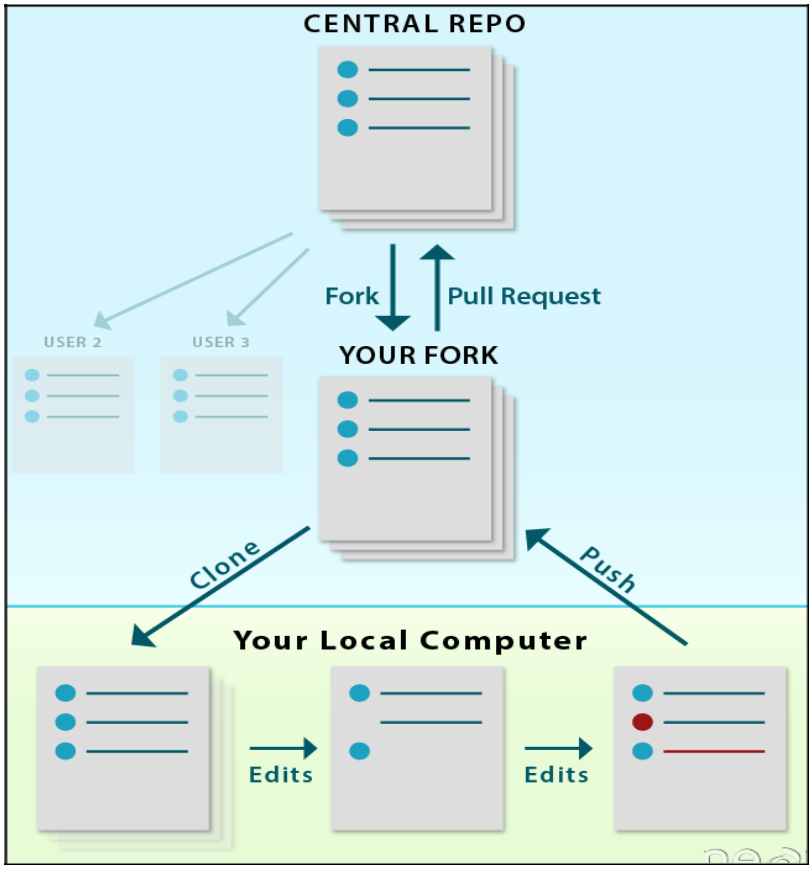
--



Experiment 7

Aim: Fork and Commit

Theory: A fork is a copy of a repository that you manage. It allows us to freely experiment with the data. After creating a fork, we can make any desired change like adding collaborators, rename files, generate GitHub pages but all these changes won't be reflected in the original repository.



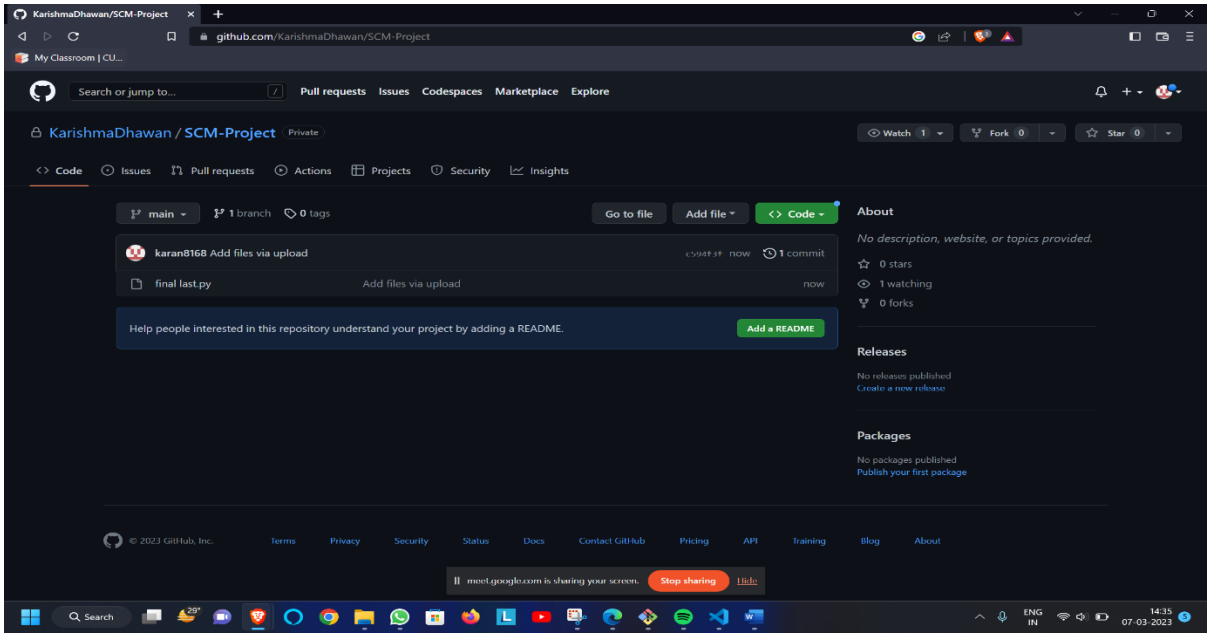
reference for picture: https://www.earthdatascience.org

To import the changes into the original repository, the user needs to send a pull request to the maintainer. If the maintainer closes the pull request only then the content can be added to the original repository.

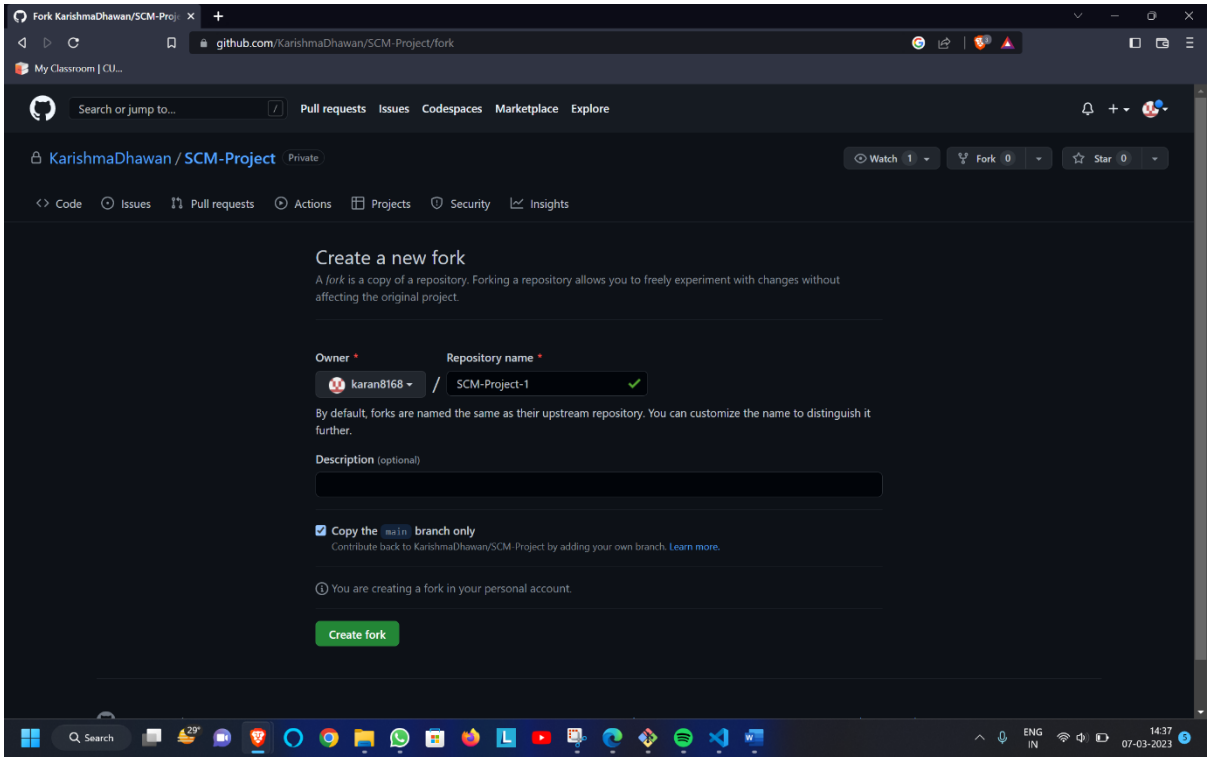
Forking is a better method than directly cloning any repository, as in cloning only the default branch is cloned whereas forking creates a clone of the complete repo and also allows us to push the changes to the main repository by using open and close pull request.

Procedure:

1. To fork a repository first thing, you need to do is, sign in to your GitHub account and then you come to the repository you want to fork, so here for demo purpose

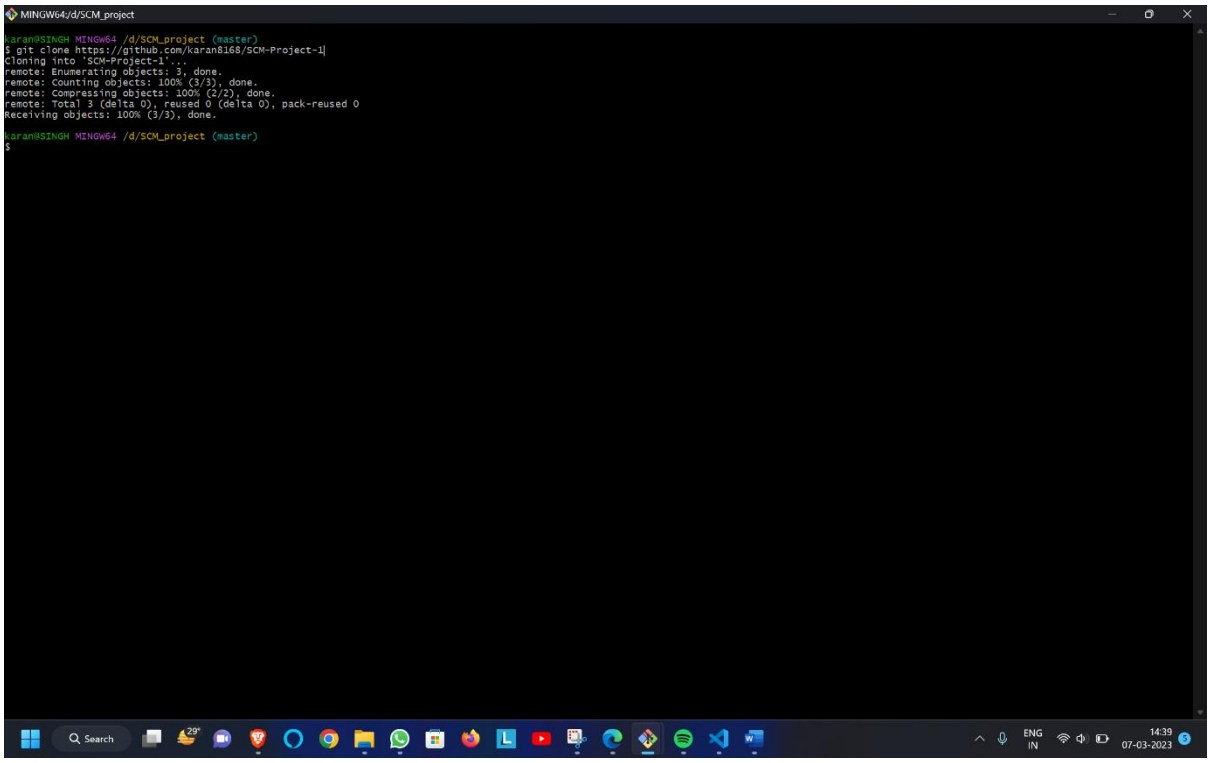


2. Click on the **Fork** button on right upside corner. Then it will ask to create a new fork, add description if you want and then click on create fork.

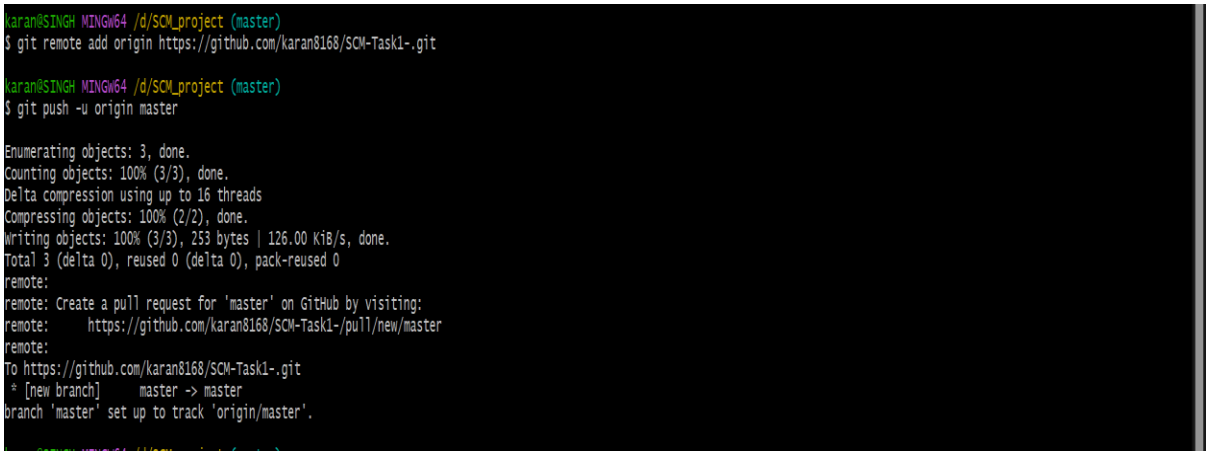


3. Now you will have a copy of the repo you have forked from other user. Now you can do any modification you want without making changes to main source code.

4. Now type git clone https://github.com/karan8168/SCM-Project-1on git.
   Git clone <URL> --> This command is used to fetch the remote repo or to clone the repo.



5. Now Open the file make changes in it and commit it and push it to remote.

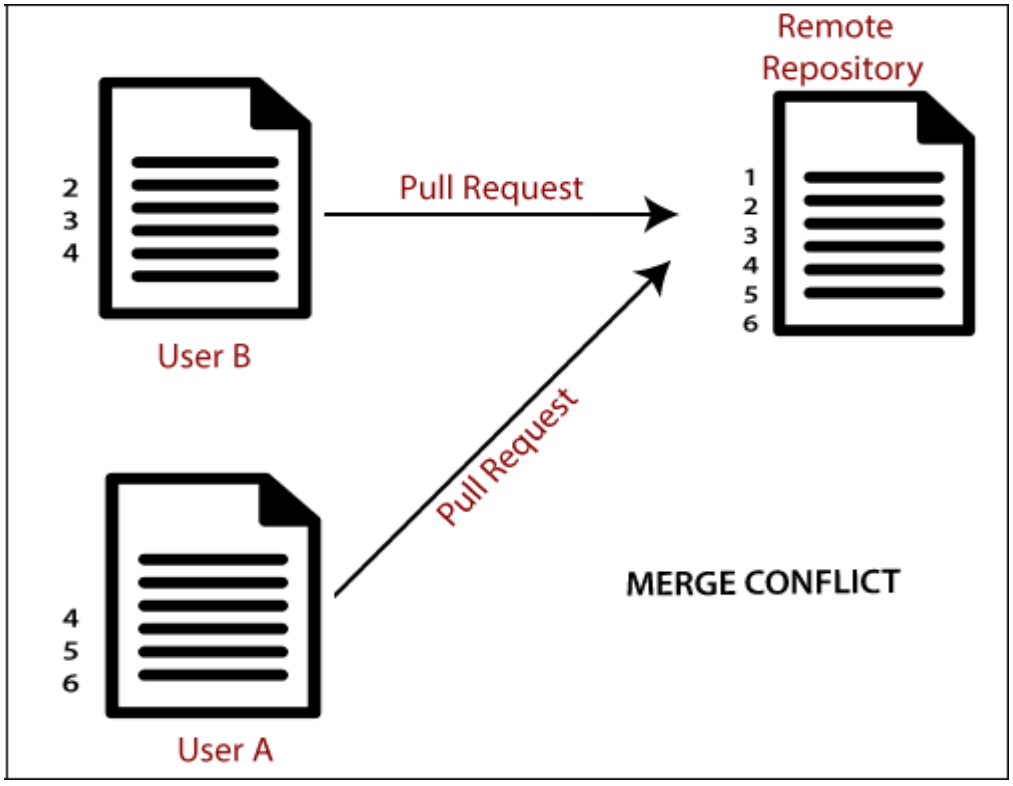


Experiment 8

Aim: Merge and Resolve conflicts created due to own activity and collaborators activity.

Theory:

Version control systems are all about managing contributions between multiple distributed authors (usually developers). Sometimes multiple developers may try to edit the same content. If Developer A tries to edit code that Developer B is editing a conflict may occur.



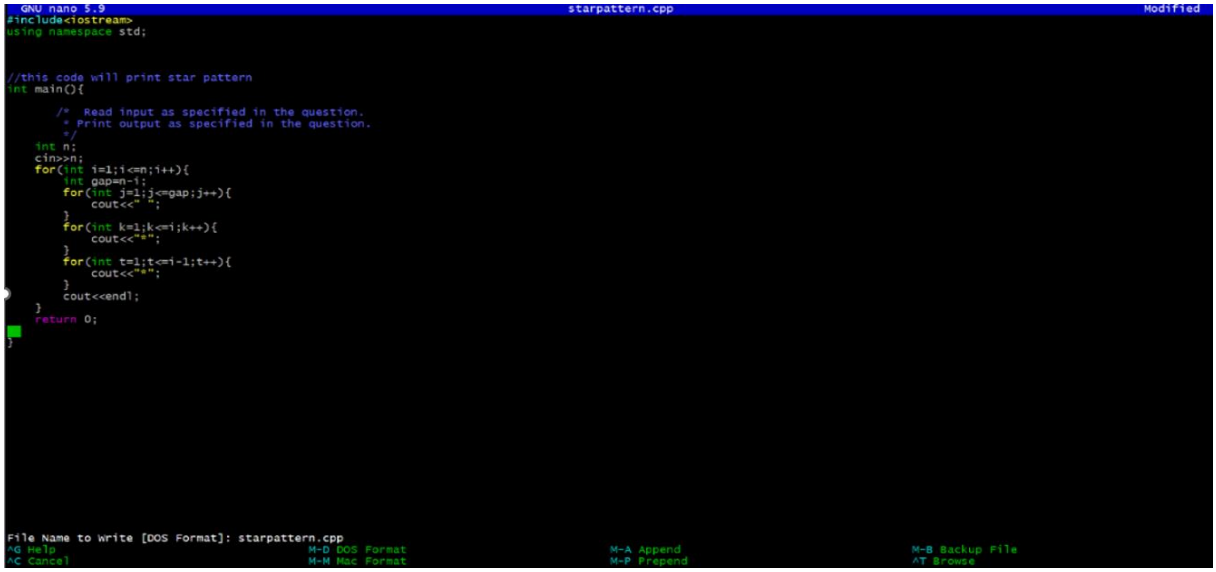
reference for picture: https://www.javatpoint.com/git-merge-and-merge-conflict

If you have a merge conflict on the command line, you cannot push your local changes to GitHub until you resolve the merge conflict locally on your computer.

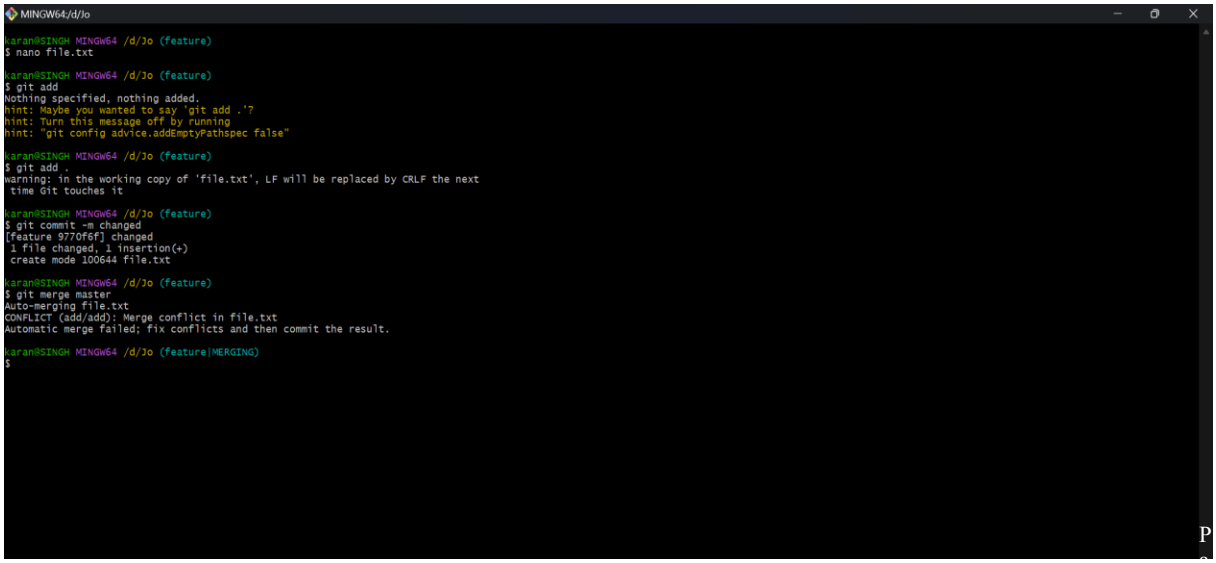
To alleviate the occurrence of conflicts developers will work in separate isolated branches. If a merge conflict still arises between the compare branch and base branch in your pull request, you can view a list of the files with conflicting changes above the Merge pull request button. The Merge pull request button is deactivated until you've resolved all conflicts between the compare branch and base branch.

Procedure:

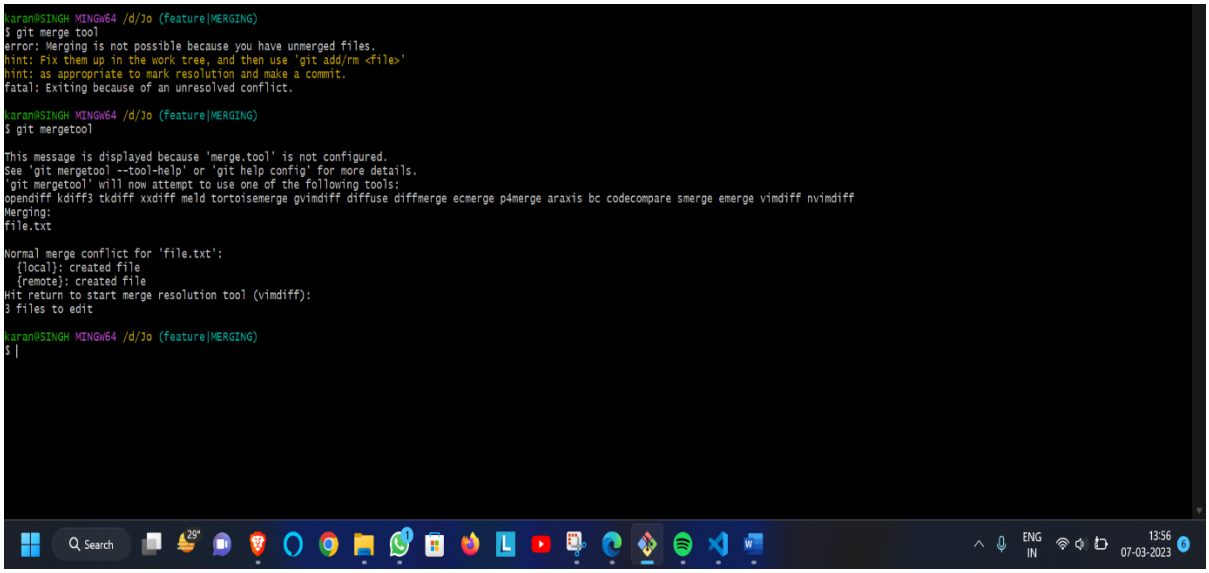
1) Do changes in master branch and commit those change. And checkout to different branch and again do changes and commit it. Now checkout to master branch and merge that branch in master.



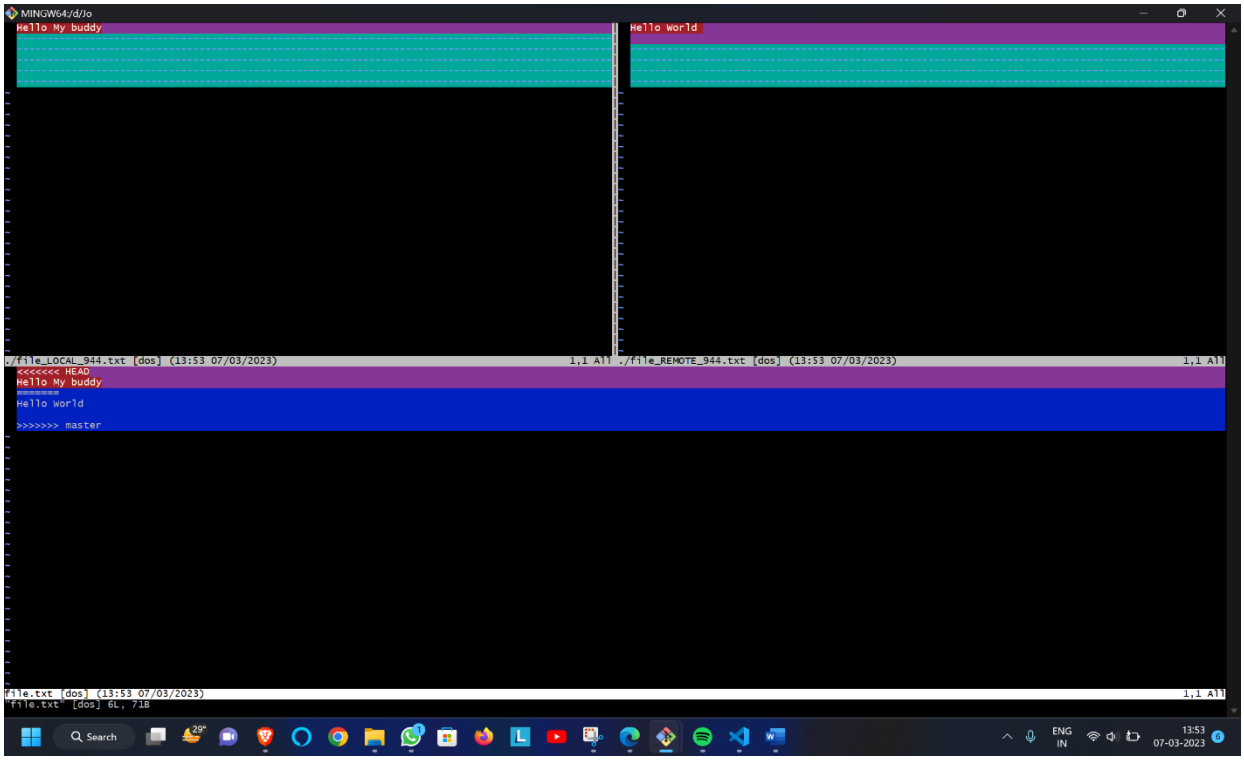
2. Now try to merge it will give Conflicts Error.



3. Use Command “git mergetool” to solve the conflict.
   git -mergetool – Run merge conflict resolution tools to resolve merge conflicts.



4. Press “I” to insert, after insertion. Press “: wq”. The merge conflict
5. is solved and feature branch is merged to master branch.



Experiment 9

Aim: Reset and Revert

Theory:

Git-revert – Revert some existing commits.

A reset is an operation that takes a specified commit and resets the "three trees" to match the state of the repository at that specified commit. A reset can be invoked in three different modes which correspond to the three trees. In reset, rest of the commits wash out after the mentioned commit. This is a limitation of reset command that we cannot have any random access.

A revert is an operation that takes a specified commit and creates a new commit which inverses the specified commit. git revert can only be run at a commit level scope and has no file level functionality.

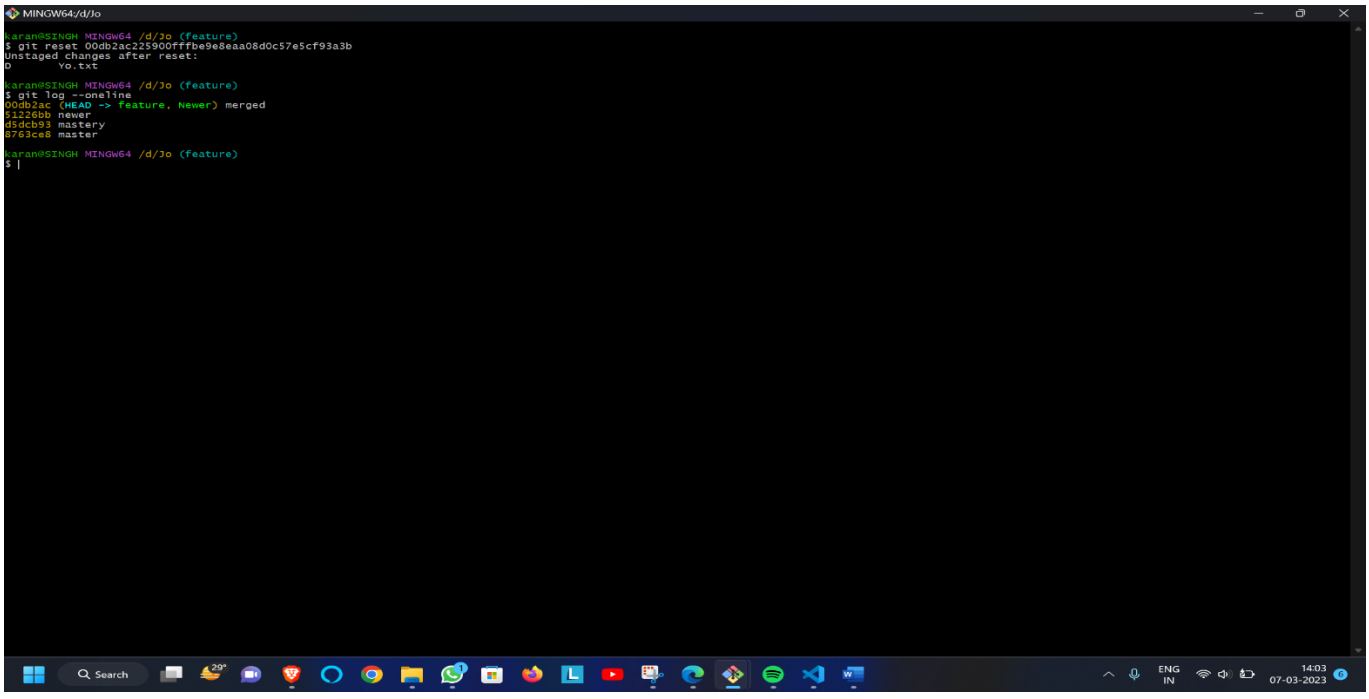
These two features justify the Version- controlled feature of the git as we can rollback to any version at any time.

PROCEDURE:

Firstly, prepare a log of multiple commits to make the reset and revert command function.

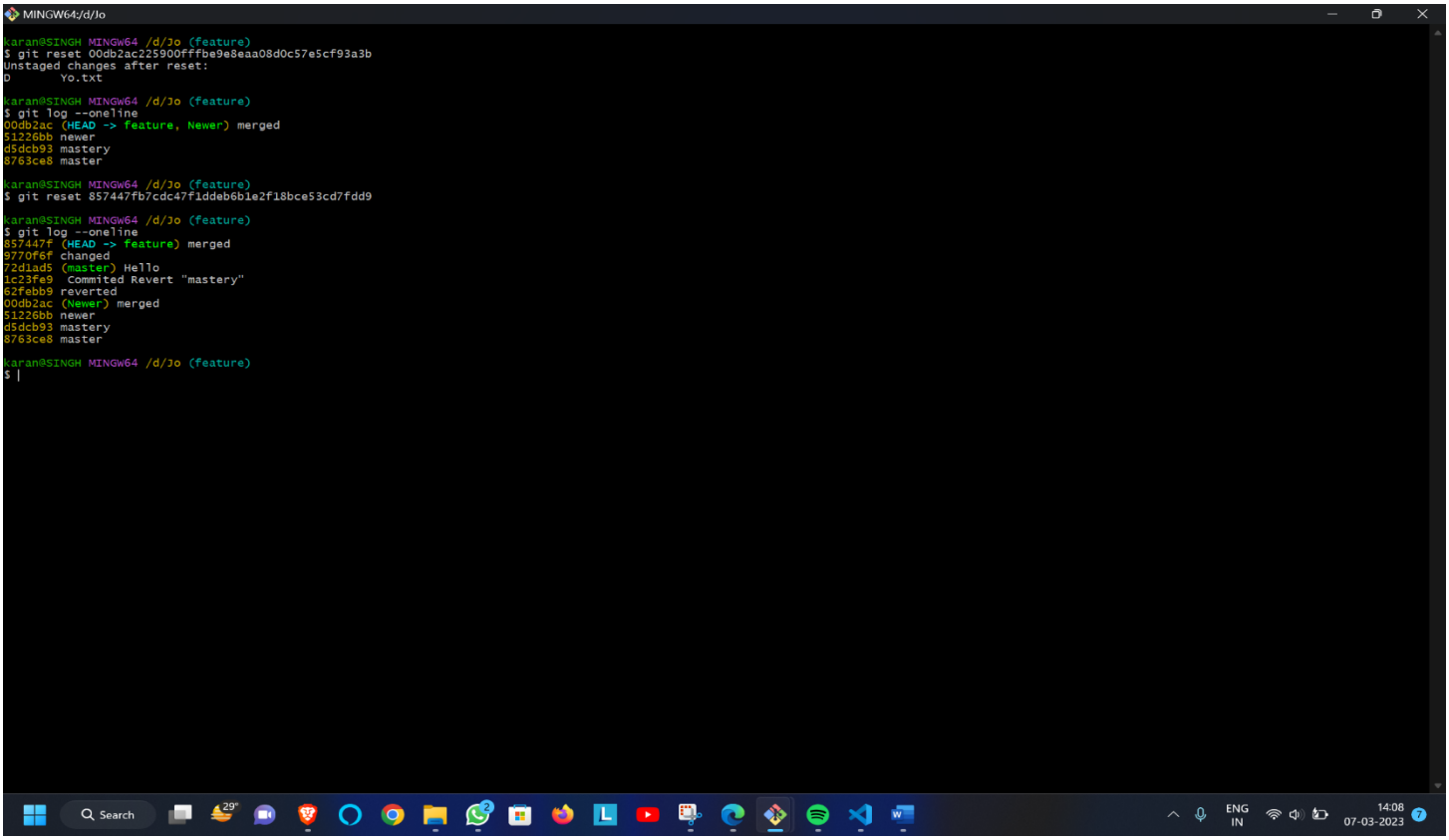
Reset: reset is the command we use when we want to move the repository back to a previous commit, discarding any changes made after that commit.

1) Create few files, stage them and commit.
2) Check the git log
3) Pick any commit where you want the repository to rollback. Copy its checksum and paste it in the $ **git reset checksum** command.



The head is now pointing the commit whose checksum we have provided that means the commits that followed vanished.

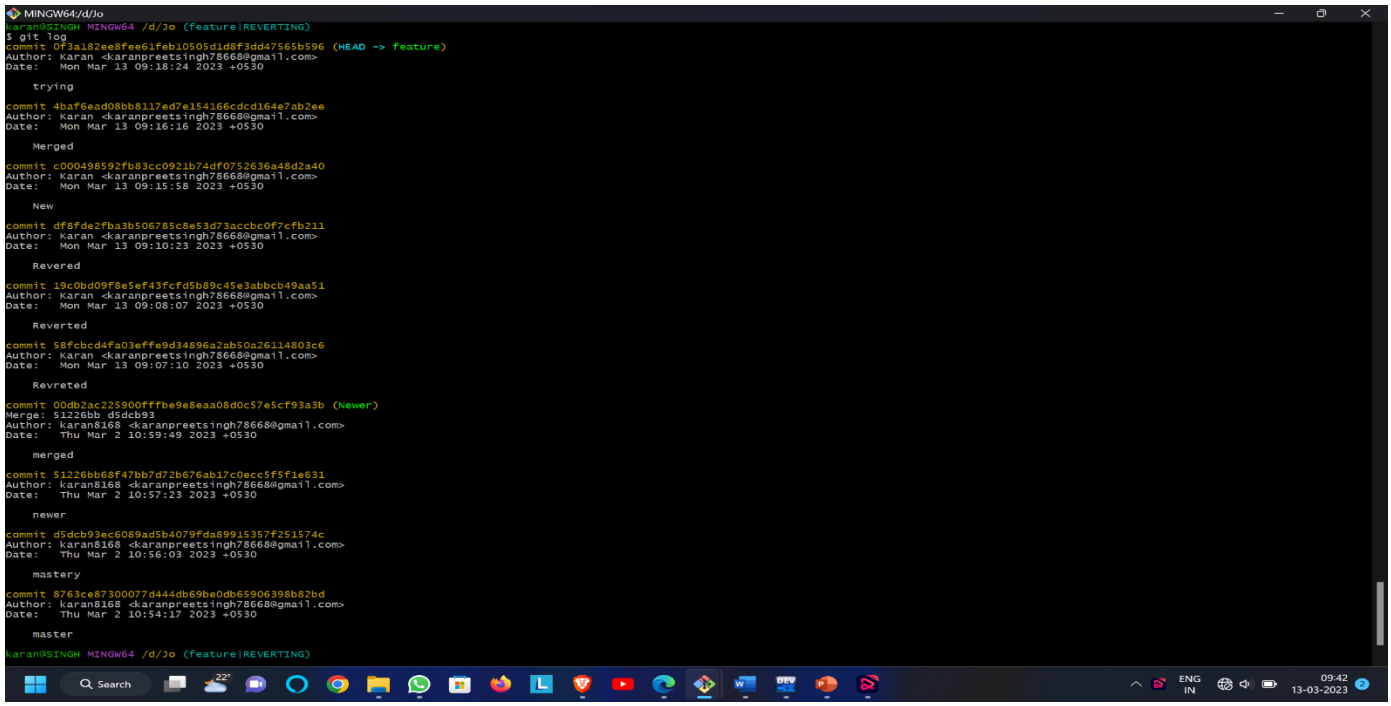
4) In you want undo this change, you copy the checksum of the commit you want back and run the same command again.



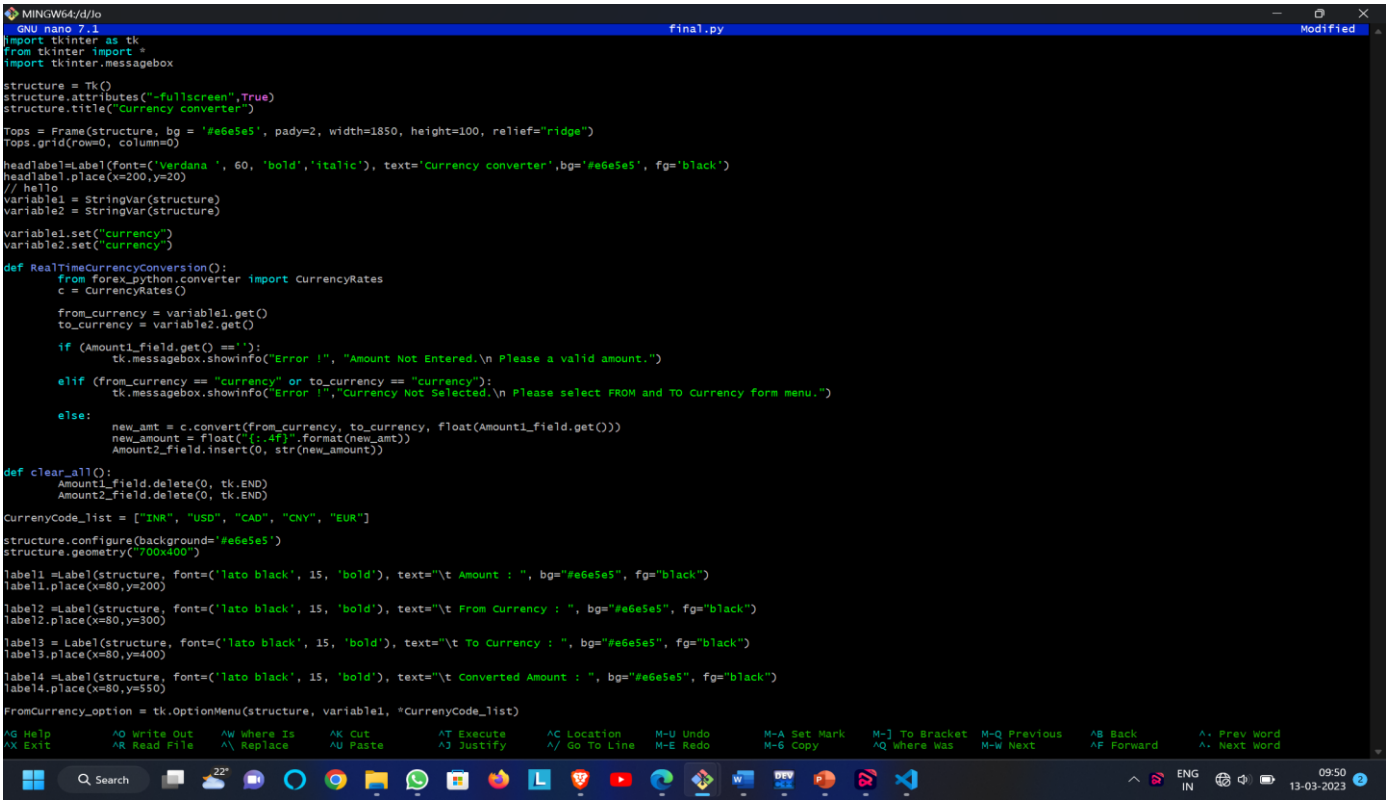
Revert:

Follow these steps to revert any change:

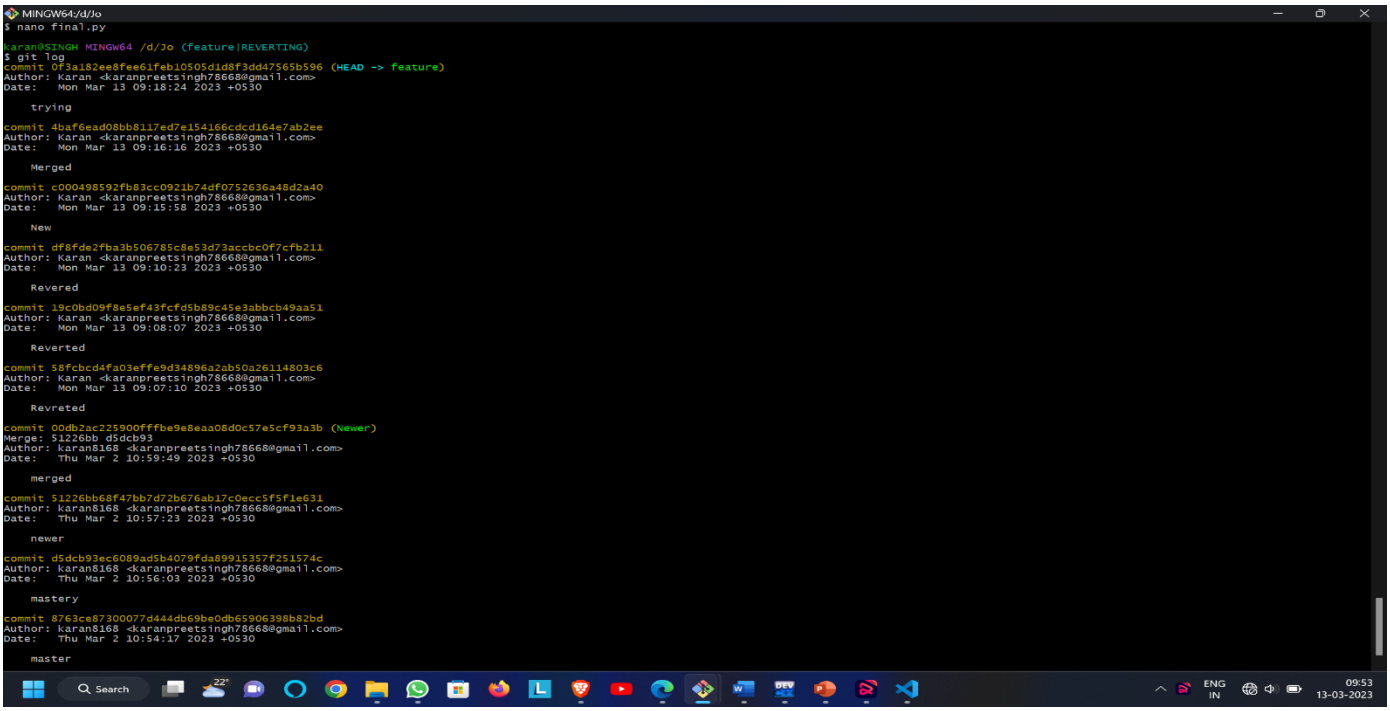
1) Pick the change where you want the project to revert back.
   Copy its checksum and paste it in the revert command.



2) A window will appear. Press ‘I” and write the statement you want to be displayed for reverting the change.



3) After completing press ‘esc’ and write: wq in the terminal
4) Check the git log and you will find another commit is added without affecting the rest commits.



5) The change associated to the reverted commit has disappeared.

A Project Report

on

“Task 2”

with

Source Code Management

(22CS003)

Submitted by:

Kartavya Tomar
Roll No. 2210990484
Submission of Task 2

Submitted To:

Dr. Chetna Kaushal
Assistant Professor
Department of Computer Science & Engineering
Chitkara University Institute of Engineering and Technology Rajpura, Punjab

Team Member 1 Name: Karanpreet Singh	Roll No. 2210990484
Team Member 2 Name: Karishma Dhawan	Roll No. 2210990483
Team Member 3 Name: Kartavya Tomar	Roll No. 2210990484

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: -	**2022**
Vertical Name: -	**First Year**
Group No: -	G24-B

INDEX

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 anysize, 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

I. **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

At present there are various programs to convert one currency into another. This may result into a code that may be absolute and does not adapt to the changing environments. In order to deal with such a issue a special program is needed that adapts to the changing environment and provide with the absolute results.

Solution

Use of various libraries like forex python to calculate one currency to another is used. This helps to manage currency conversion quite accurately. Not only does this enable accurate conversion but it also enables various buisness to make fast and reliable calculations in a short amount of time

Objective

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

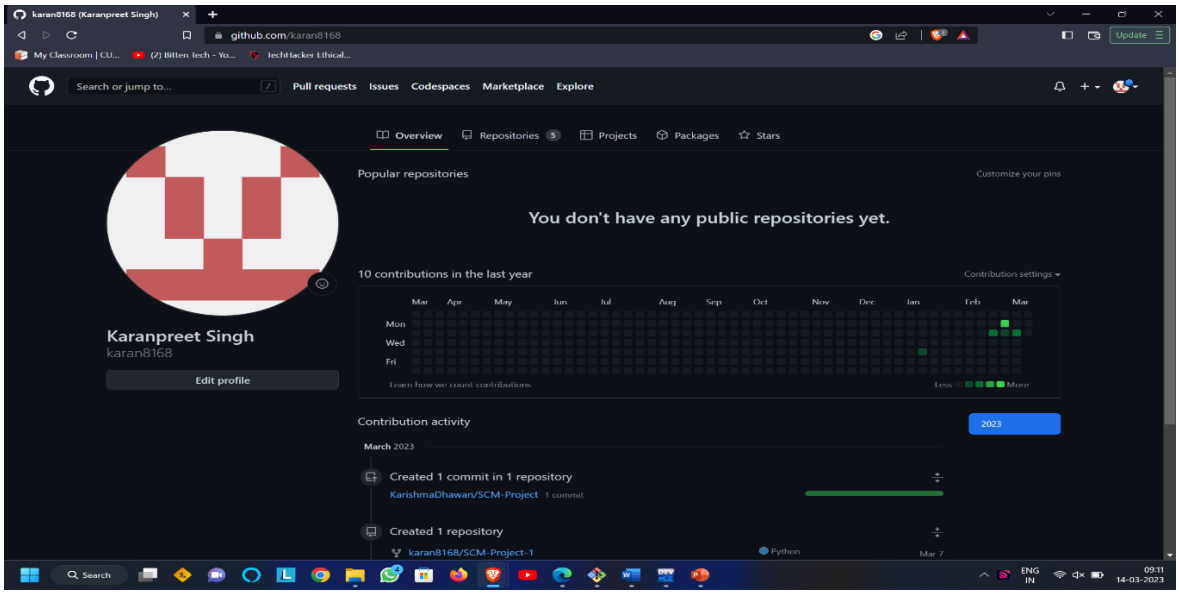
1. This is required because the collaboration makes the team work easy.
2. The code becomes manageable and we can build a clean repository.
3. Tracking and resolving of the errors is quite feasible in this process.
4. Moreover, we can make our locally available projects, globally available.

CHITKARA UNIVERSITY

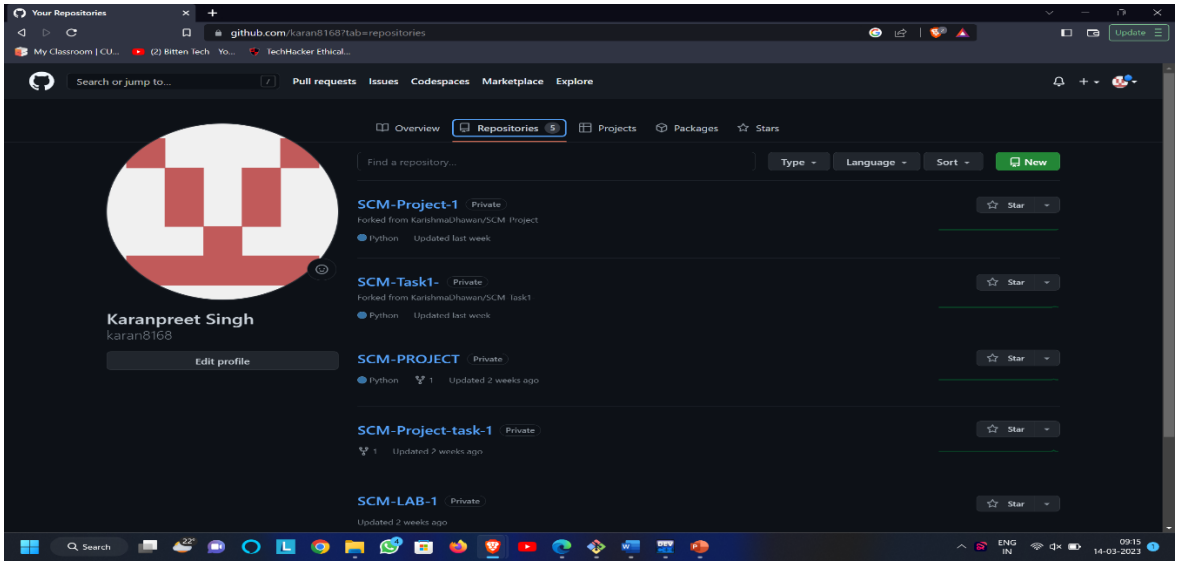
Experiment No. 01

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.

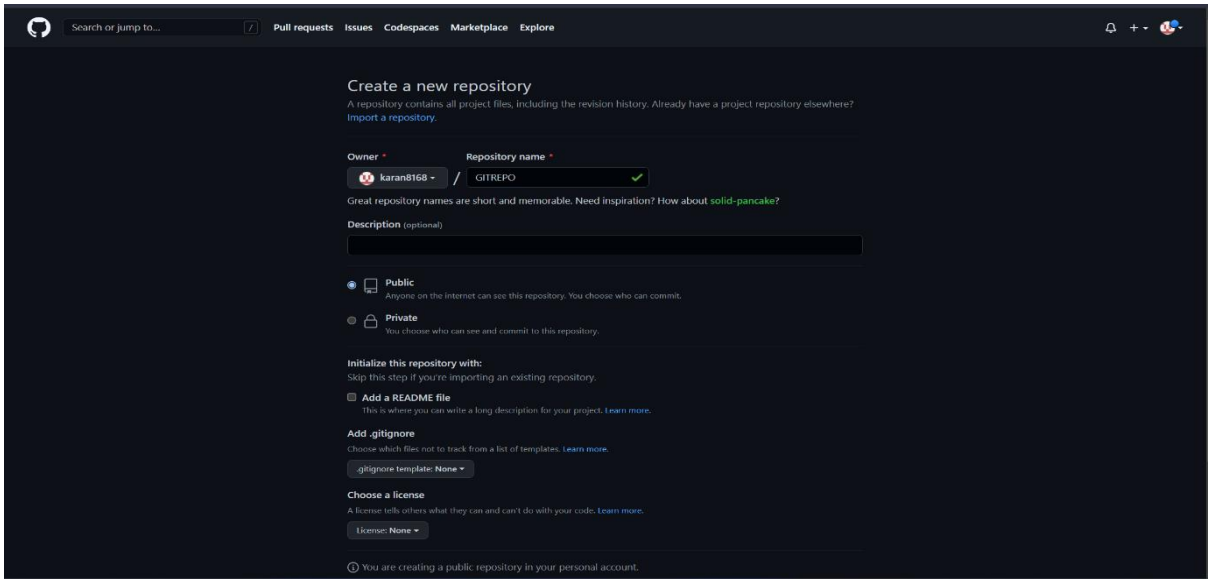


2) Click on the 'New' button in the top right corner.

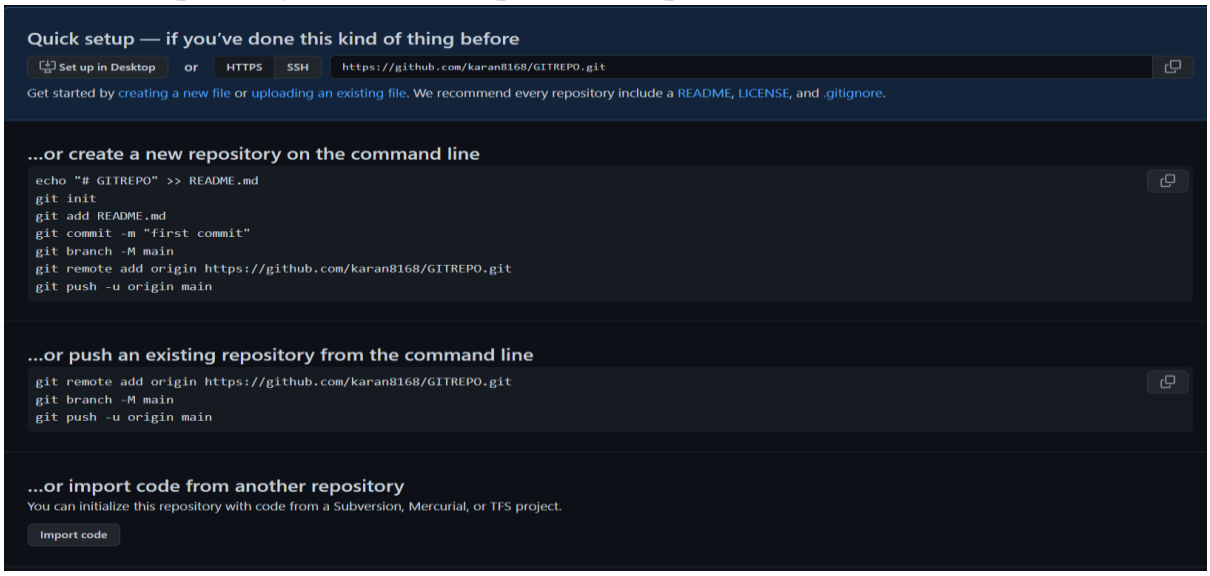


3) Enter the Repository name and add the description of the repository.

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

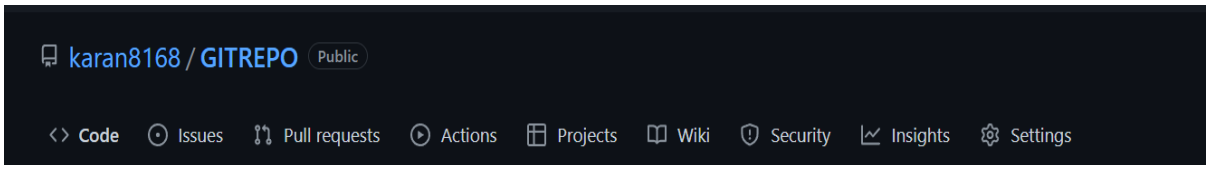


5) If you want to import code from an existing repository select the import code option.

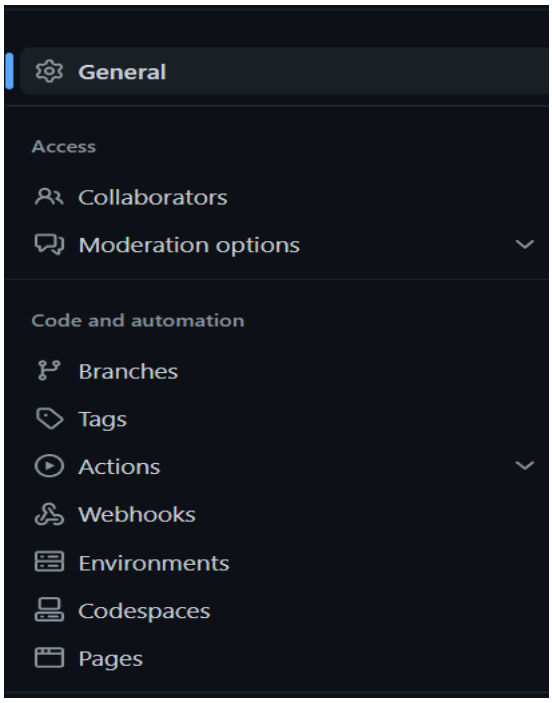


6) Now, you have created your repository successfully.

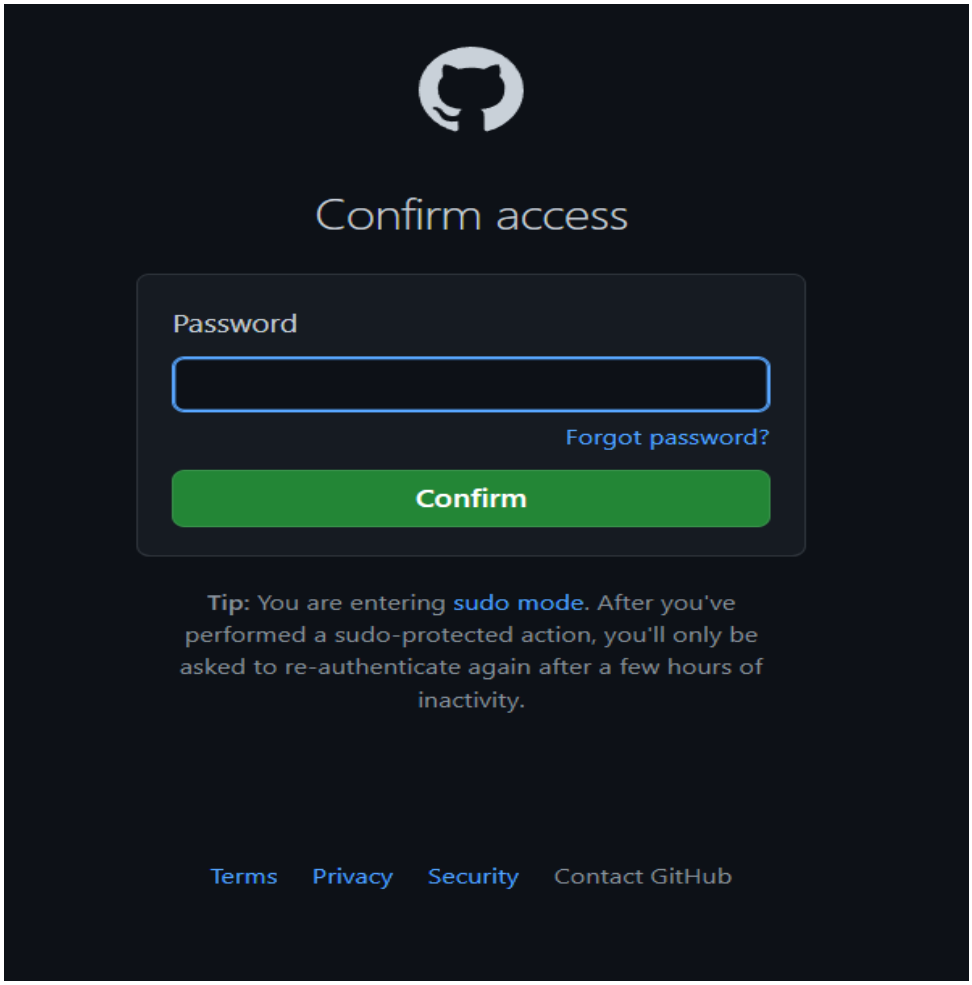
7) To add members to your repository, open your repository and select settings option in the navigation bar.



8) Click on Collaborators option under the access tab.

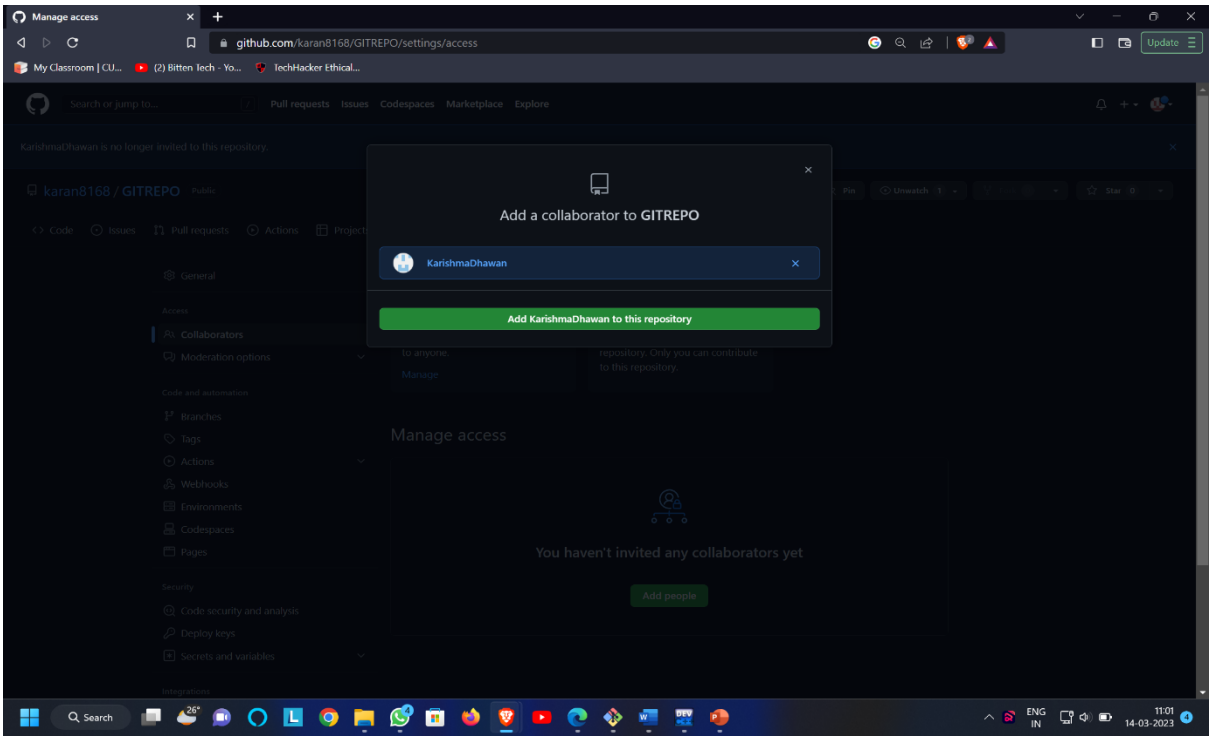


9) After clicking on collaborators GitHub asks you to enter your password to confirm the access to the repository.

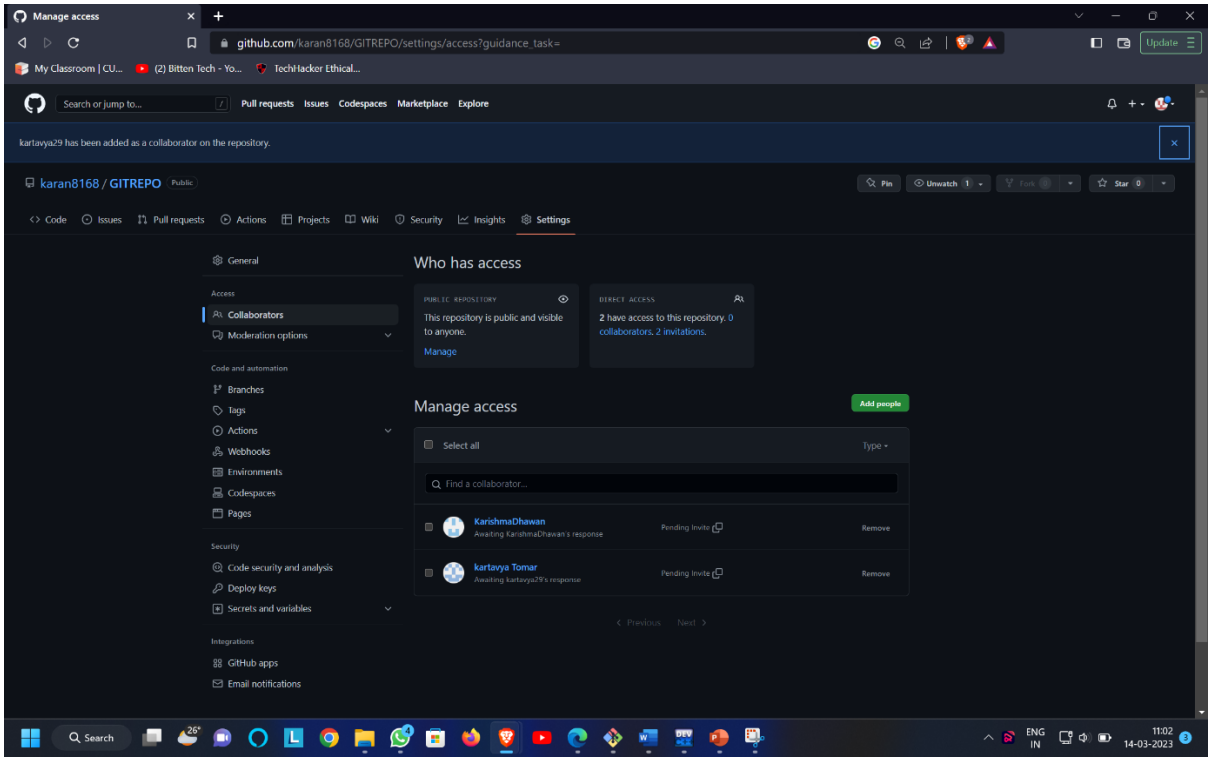


10) After entering the password, you can manage access and add/remove team members to your project.

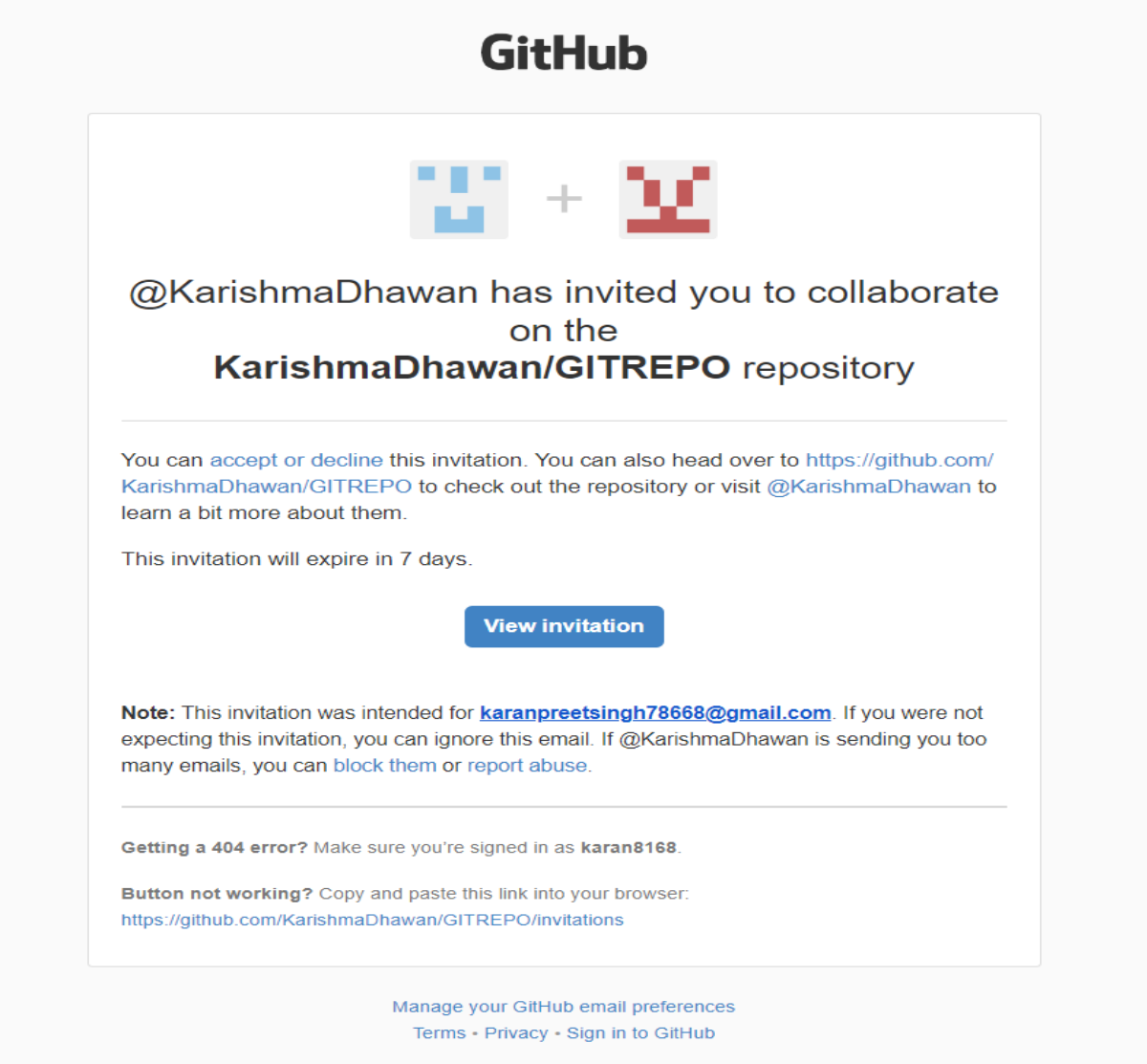
11) To add members, click on the add people option and search the id of your respective team member.



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

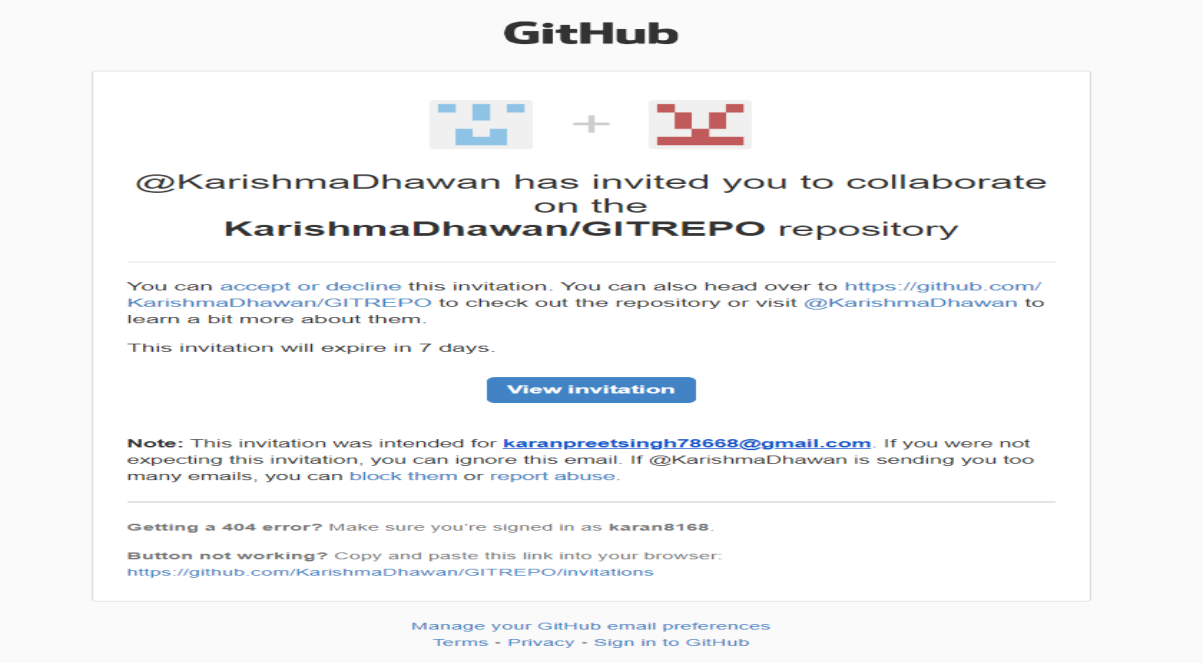


13) To accept the invitation from your team member, open your mail registered with GitHub.



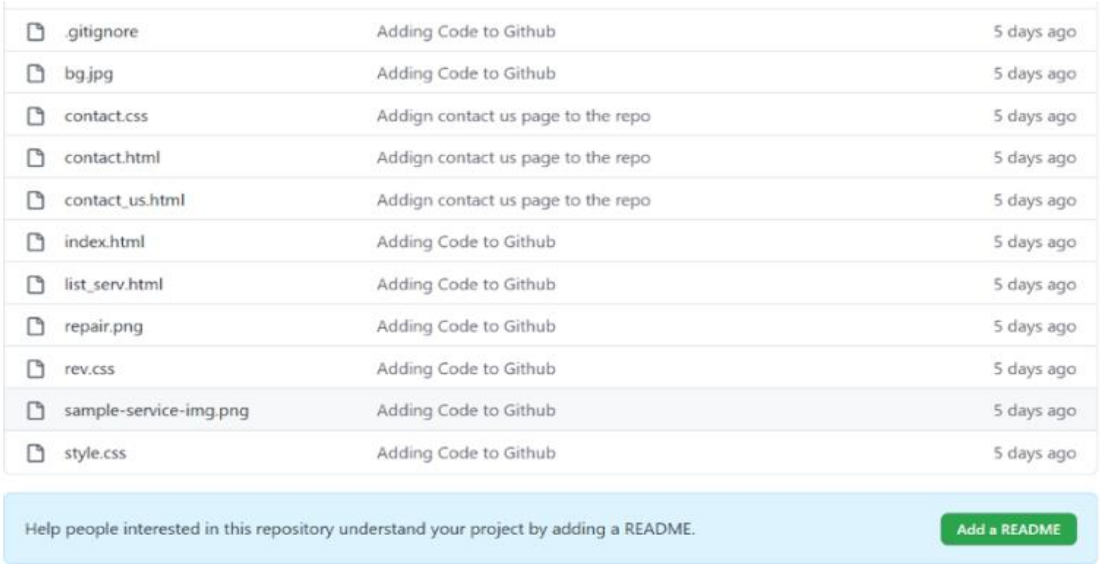
14) You will receive an invitation mail from the repository owner. Open the email and click on accept invitation.

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

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

```
himan@DESKTOP-ESG6013 MINGW64 ~/OneDrive/Desktop/scm (master)
$ git clone https://github.com/karan16kkg/Jay-Ho.git
Cloning into 'Jay-Ho'...
remote: Enumerating objects: 6, done.
remote: Counting objects: 100% (6/6), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 6 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (6/6), done.

himan@DESKTOP-ESG6013 MINGW64 ~/OneDrive/Desktop/scm (master)
$
```

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

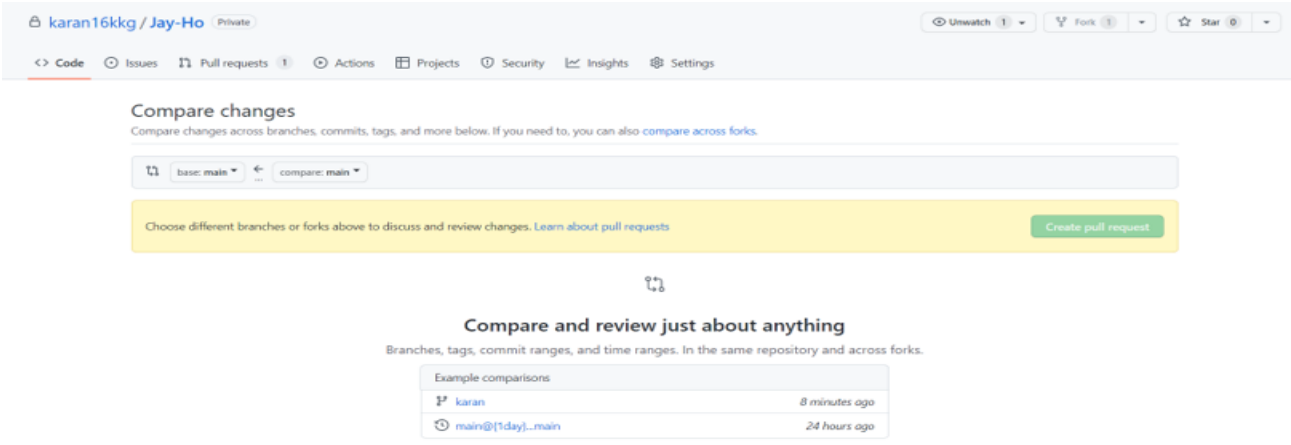
```
himan@DESKTOP-ESG6013 MINGW64 ~/OneDrive/Desktop/scm (master)
$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        Jay-Ho/

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

himan@DESKTOP-ESG6013 MINGW64 ~/OneDrive/Desktop/scm (master)
$ git add --all
warning: adding embedded git repository: Jay-Ho
hint: You've added another git repository inside your current repository.
hint: Clones of the outer repository will not contain the contents of
hint: the embedded repository and will not know how to obtain it.
hint: If you meant to add a submodule, use:
hint:
hint:   git submodule add <url> Jay-Ho
hint:
hint: If you added this path by mistake, you can remove it from the
hint: index with:
hint:
hint:   git rm --cached Jay-Ho
hint:
hint: See "git help submodule" for more information.

himan@DESKTOP-ESG6013 MINGW64 ~/OneDrive/Desktop/scm (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   Jay-Ho


himan@DESKTOP-ESG6013 MINGW64 ~/OneDrive/Desktop/scm (master)
$ git commit -m "commit1"
[master 1193c92] commit1
 1 file changed, 1 insertion(+)
 create mode 160000 Jay-Ho

himan@DESKTOP-ESG6013 MINGW64 ~/OneDrive/Desktop/scm (master)
$ git status
On branch master
nothing to commit, working tree clean

himan@DESKTOP-ESG6013 MINGW64 ~/OneDrive/Desktop/scm (master)
$
```

3) Add and commit the changes to the local repository.

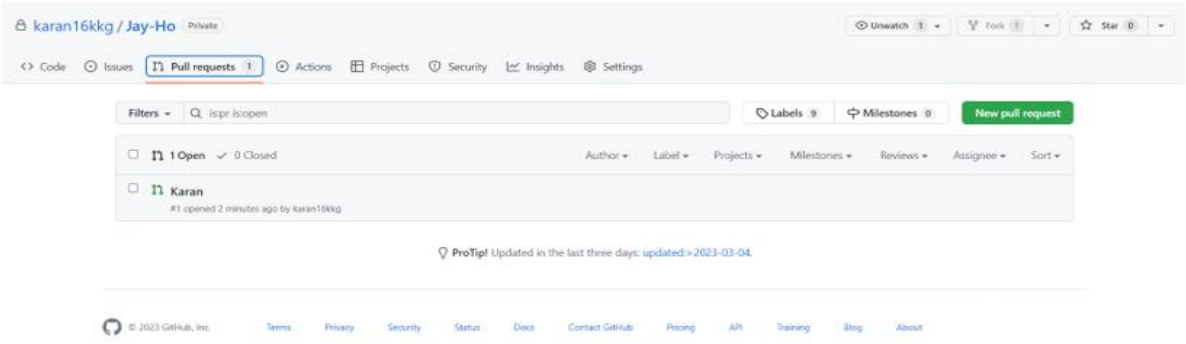
4) Use git push origin *Branch name* option to push the new branch to the main repository.

```
himan@DESKTOP-ESG6013 MINGW64 ~/OneDrive/Desktop/scm/Jay-Ho (karan16kkg)
$ cd karan16kkg / Jay-Ho
bash: cd: too many arguments

himan@DESKTOP-ESG6013 MINGW64 ~/OneDrive/Desktop/scm/Jay-Ho (karan16kkg)
$ ls
Jay-Ho/  Python  Python.py

himan@DESKTOP-ESG6013 MINGW64 ~/OneDrive/Desktop/scm/Jay-Ho (karan16kkg)
$ git branch karan

himan@DESKTOP-ESG6013 MINGW64 ~/OneDrive/Desktop/scm/Jay-Ho (karan16kkg)
$ git checkout karan
Switched to branch 'karan'
A       Jay-Ho

himan@DESKTOP-ESG6013 MINGW64 ~/OneDrive/Desktop/scm/Jay-Ho (karan)
$ git status
On branch karan
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   Jay-Ho

himan@DESKTOP-ESG6013 MINGW64 ~/OneDrive/Desktop/scm/Jay-Ho (karan)
$ git add .

himan@DESKTOP-ESG6013 MINGW64 ~/OneDrive/Desktop/scm/Jay-Ho (karan)
$ git commit -m "Addign contact us page to the repo"
[karan 3dee412] Addign contact us page to the repo
 1 file changed, 1 insertion(+)
 create mode 160000 Jay-Ho

himan@DESKTOP-ESG6013 MINGW64 ~/OneDrive/Desktop/scm/Jay-Ho (karan)
$ git push origin karan
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 4 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (5/5), 1.53 KiB | 523.00 KiB/s, done.
Total 5 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'karan' on GitHub by visiting:
remote:      https://github.com/karan16kkg/Jay-Ho/pull/new/karan
remote:
To https://github.com/karan16kkg/Jay-Ho.git
 * [new branch]      karan -> karan

himan@DESKTOP-ESG6013 MINGW64 ~/OneDrive/Desktop/scm/Jay-Ho (karan)
$
```

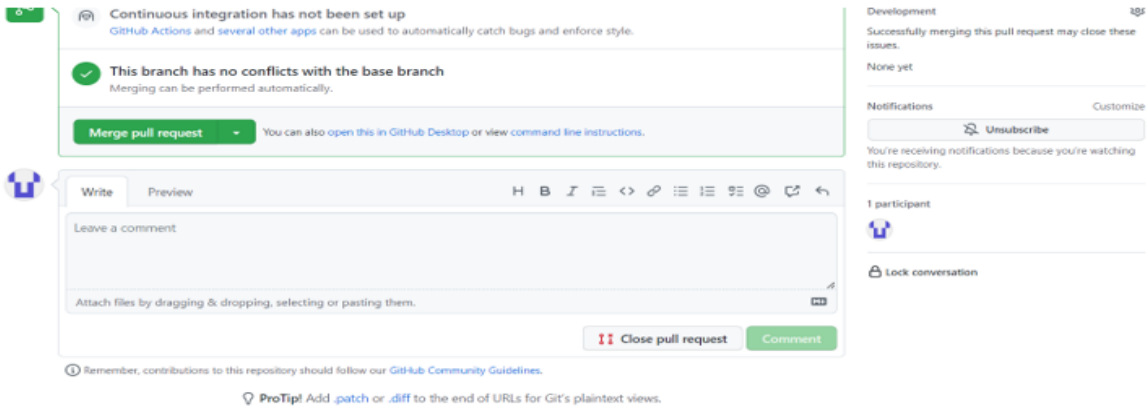
5) After pushing new branch GitHub will either automatically ask you to create a pull request or you can create your own pull request.



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.



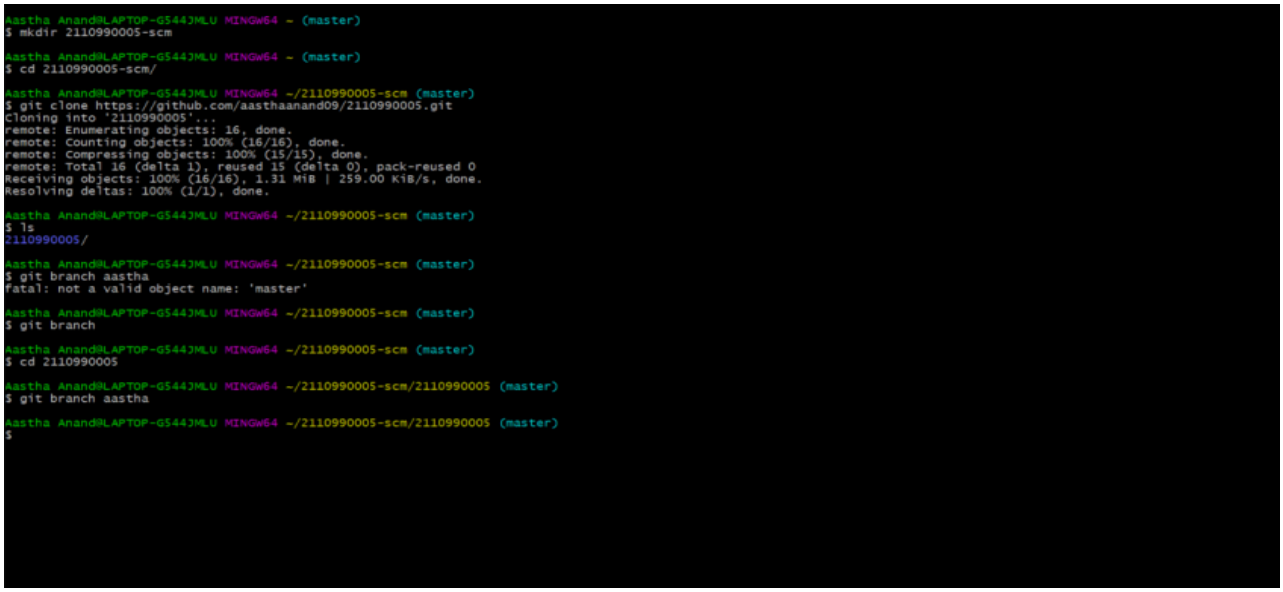
8) Af

Experiment No. 03

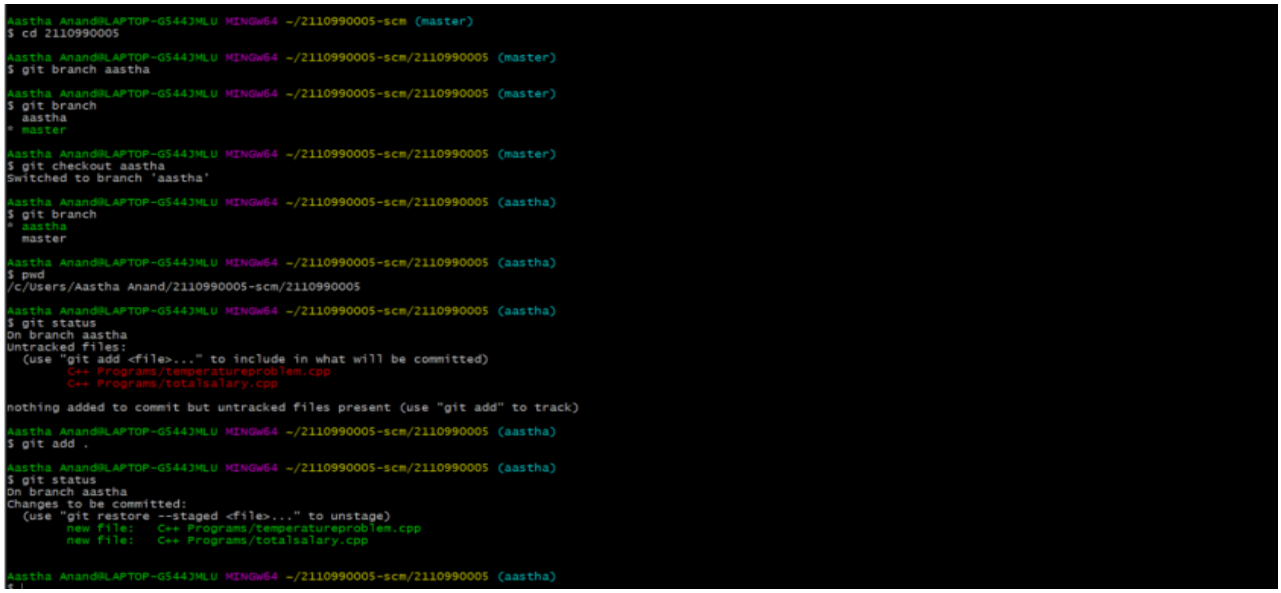
Aim: Create a pull request on a team member's repo and close pull requests generated by team members on own Repo as a maintainer

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:

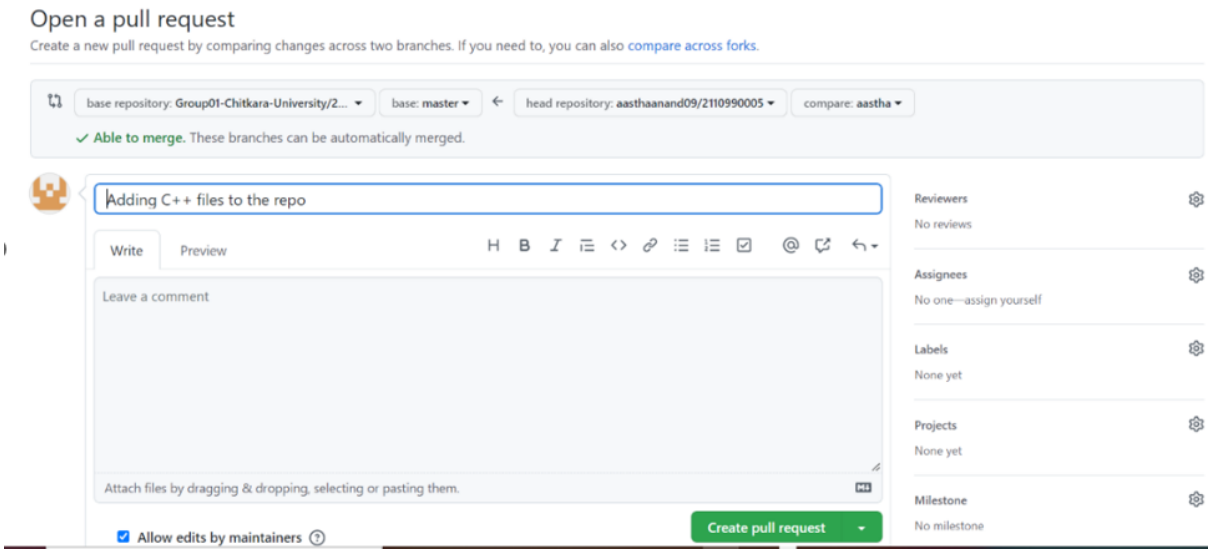
1. Do the required changes in the repository, add and commit these changes in the local repository in a new branch.



2. Push the modified branch using git push origin *Branch name*.

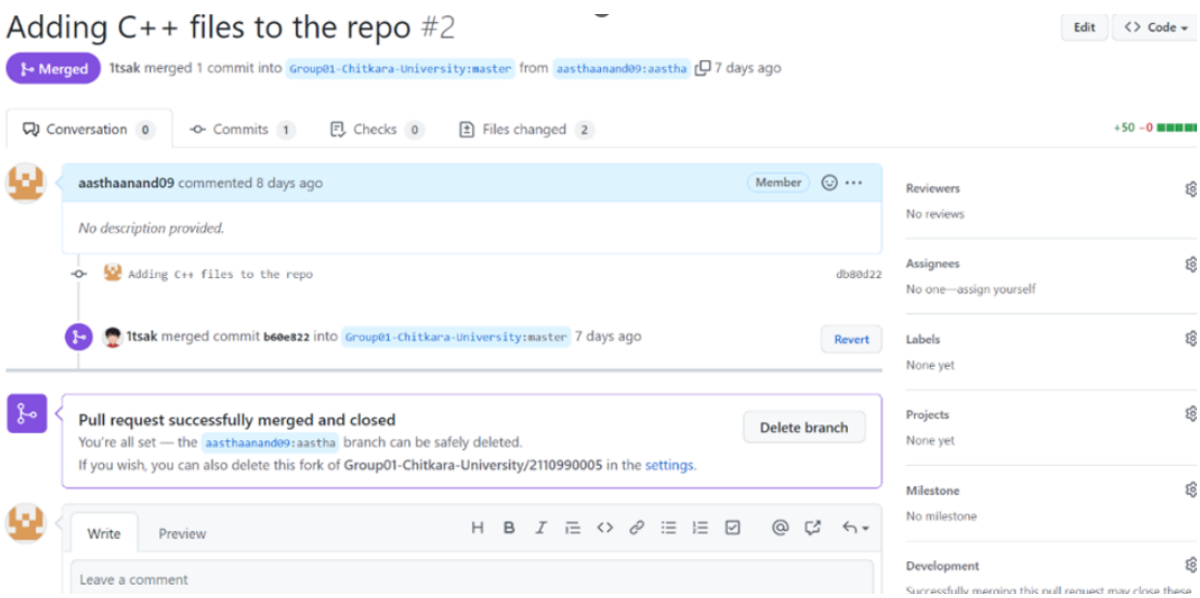


3. Open a pull request by following the procedure from the above experiment.



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.

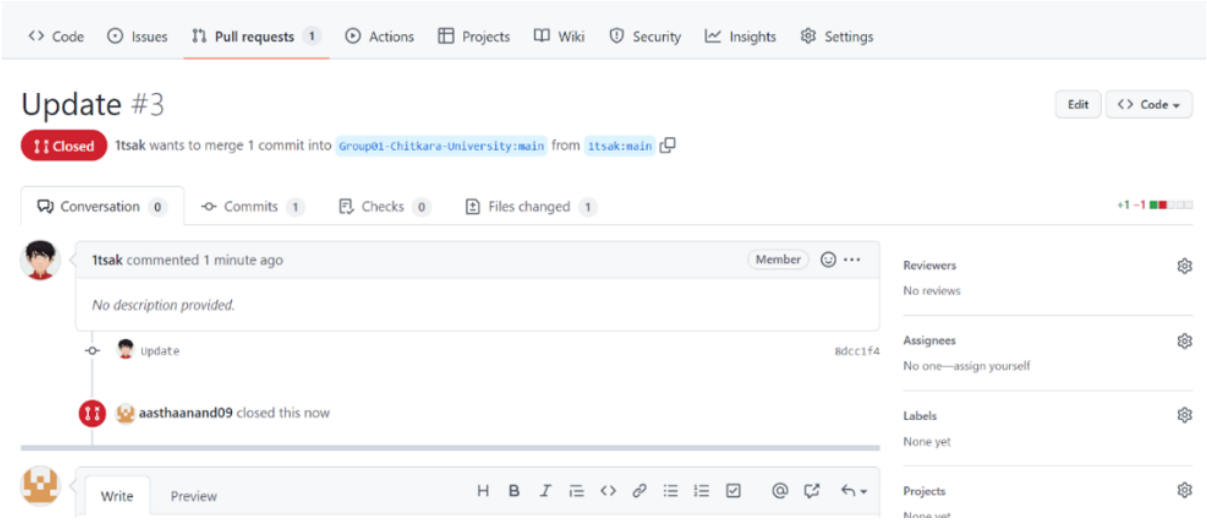


8. Click on the pull request. Two options will be available, either to close the pull request or Merge the request with the main branch.
9. By selecting the merge branch option the main branch will get updated for all the team members.

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.

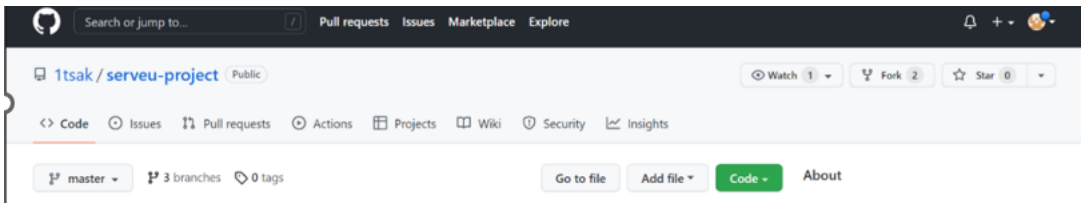
A repository's graphs give you information on traffic, projects that depend on the repository, contributors and commits to the repository, and a repository's forks and network. If you maintain a repository, you can use this data to get a better understanding of who's using your repository and why they're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

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

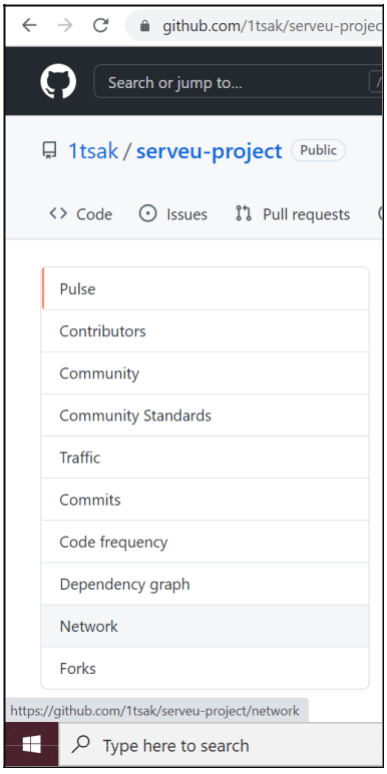
- Pulse
- Contributors
- Traffic
- Commits
- Code frequency
- Network

Steps to access network graphs of respective repository

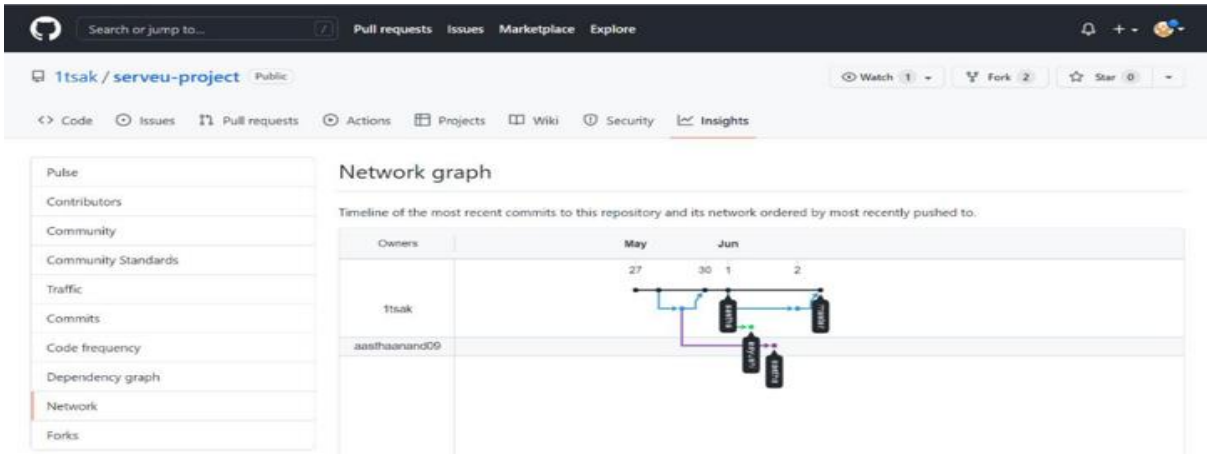
1. On GitHub.com, navigate to the main page of the repository.
2. Under your repository name, click **Insights**.



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



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.

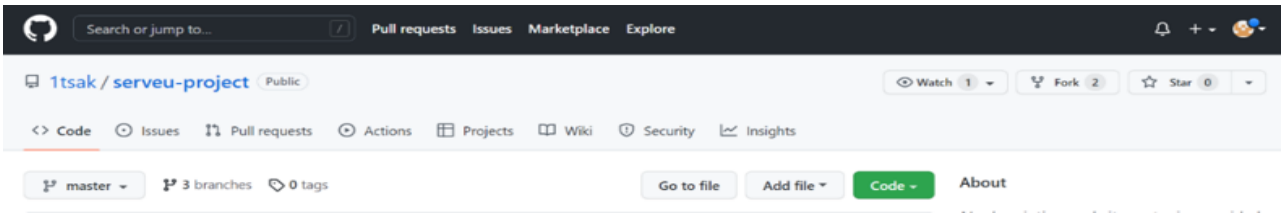


Listing the forks of a repository

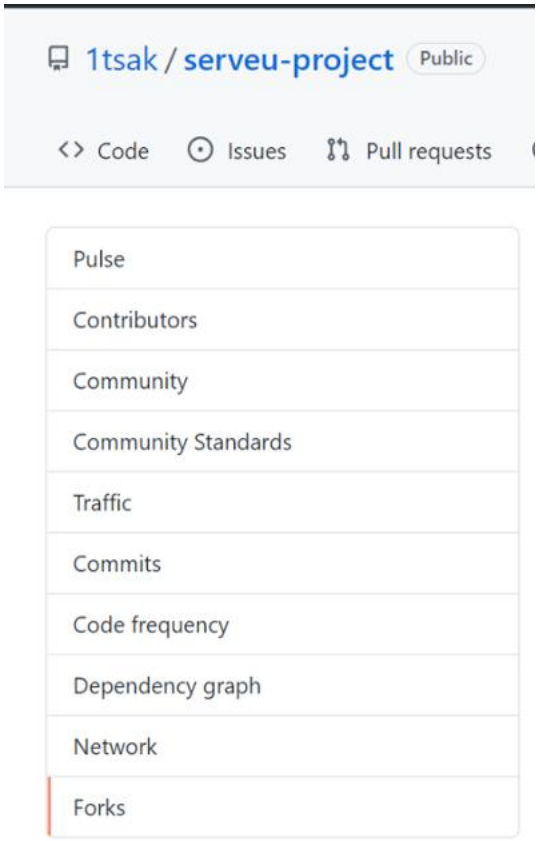
Forks are listed alphabetically by the username of the person who forked the repository

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

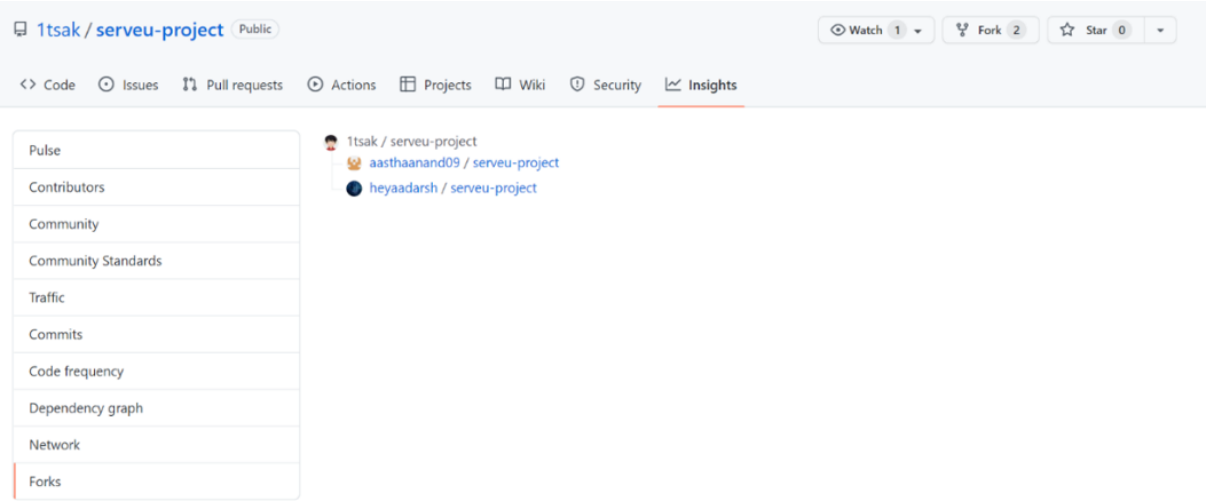
1. On GitHub.com, navigate to the main page of the repository.
2. Under your repository name, click **Insights**.



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.