Subject Name: **Source Code Management**

Subject Code: **24CSE0106**

Cluster: i-**Alpha**

Department: **DCSE**

CHITKARA UNIVERSITY
CHANDIGARH-PATIALA NATIONAL HIGHWAY
RAJPURA (PATIALA) PUNJAB-140401 (INDIA)

Submitted By:

Lavnish Kumar
2410991255
2nd Sem, G-16, BE (CSE)

Submitted To:

Dr. Aman Kumar
Assistant Professor
Chitkara University, Punjab, 140401

Index Task 1.1

EXPERIMENT NO. 1:

Aim: Introducing Version Control – Git client (CLI, GUI), Linux environment Emulation, Advantages of VCS, Installing git CLI and git GUI.

Theory:

What is Git?

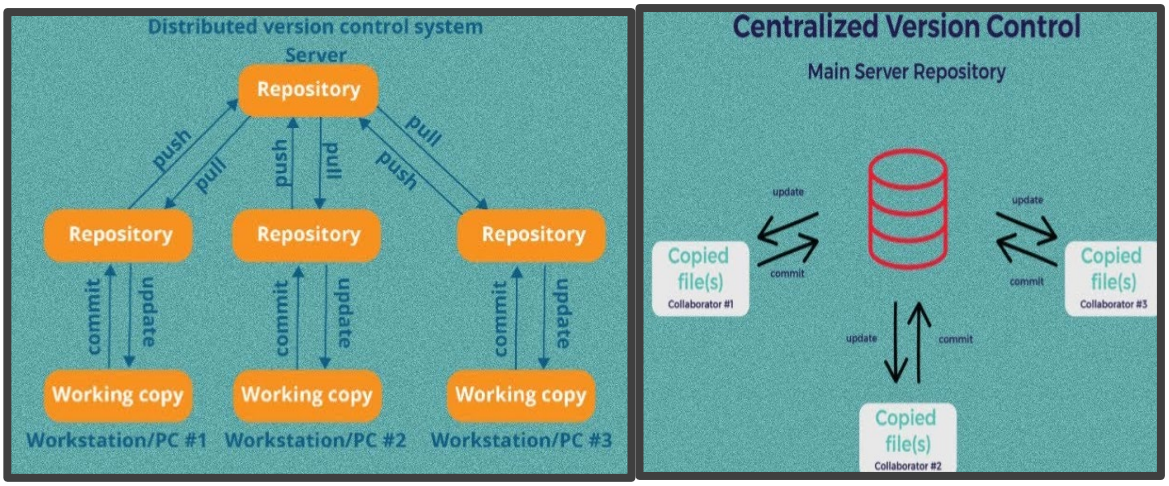
Git is a free and open-source version control system used to handle small to very large projects efficiently. This is also used for tracking changes in any set of files and usually helps in coordinating work among members of a team. Hence, enables multiple developers to work together on non-linear development.

History of VCS: The very first Version Control System was created in 1972 at Bell Labs where they also developed UNIX. The first one was called SCCS (Source Code Control System). It was available only for UNIX and only worked with Source Code files. Some types of Version Control Systems are:

• Local VCS: No internet is needed because it uses a database to keep and track of files.

• Centralized VCS: Centralized version control systems are based on the idea that there is a single "central" copy of your project somewhere (probably on a server), and programmers will "commit" their changes to this central copy. This simply means recording the change in the central system (OS).

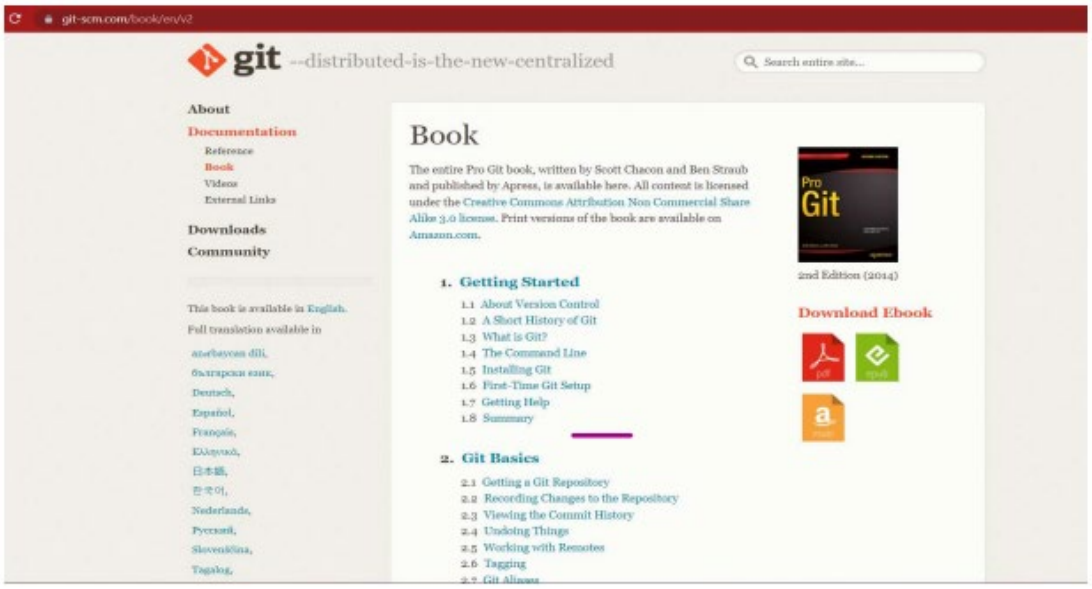
• Distributed VCS: A type of version control where the complete codebase including its full version history is mirrored on every developer's computer.



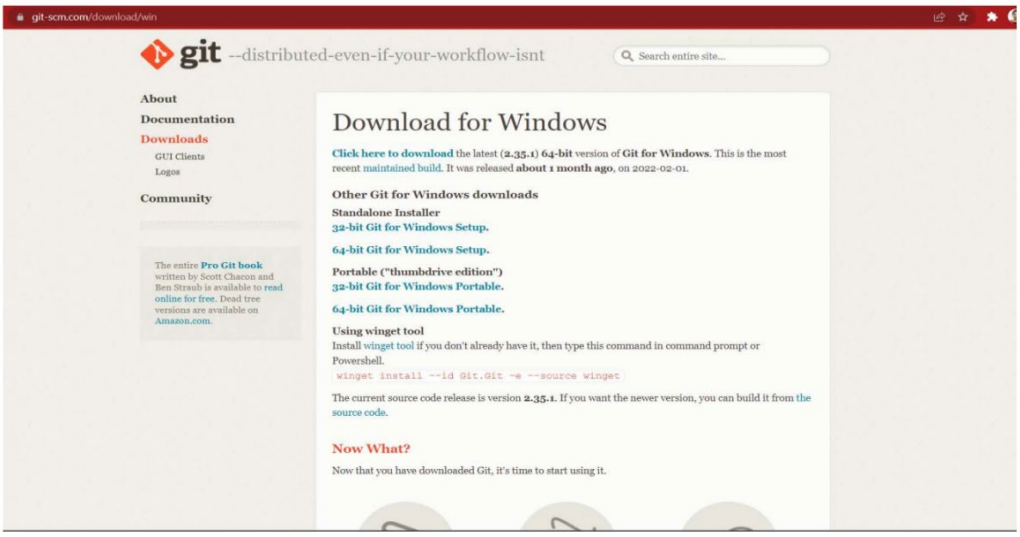
How to install GIT on Windows?

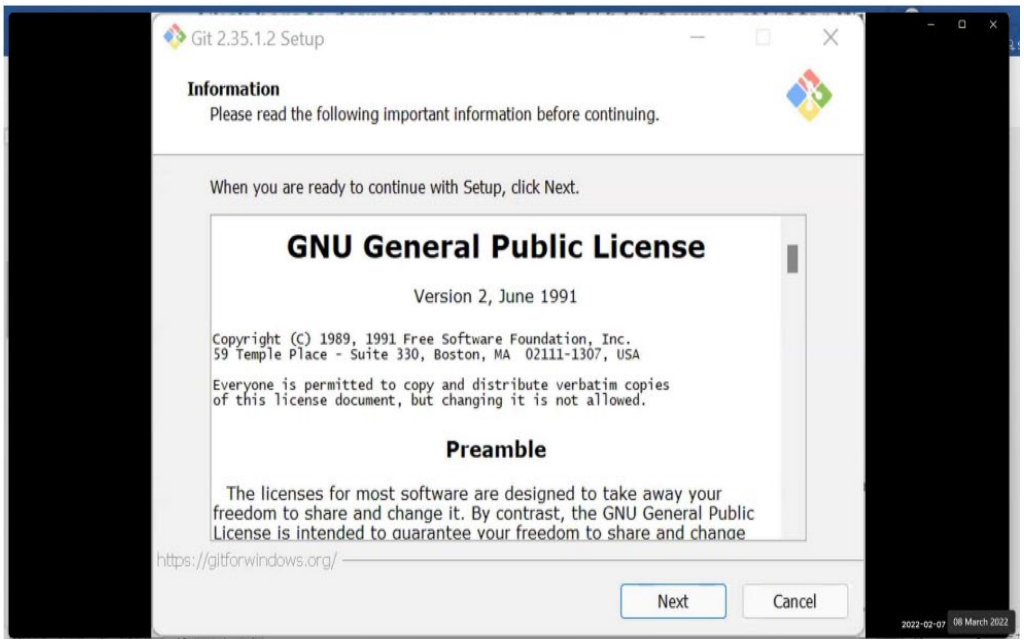
There are many ways to install Git on Windows. The most official build is available for download on the Git website. Go to https://gitscm.com/download/win and after a few settings the download will start automatically.

- Visit directly on git book page by https://git-scm.com/book/en/v2

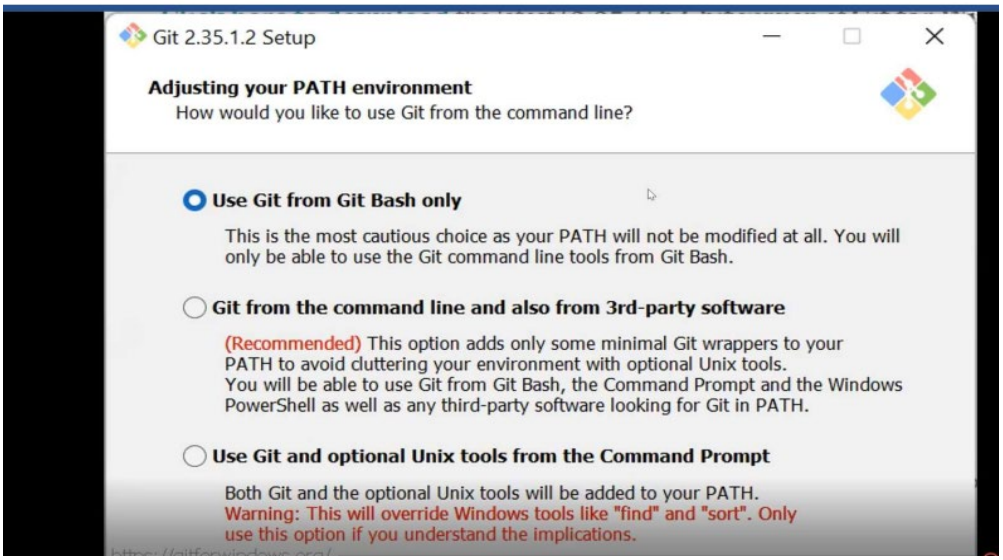


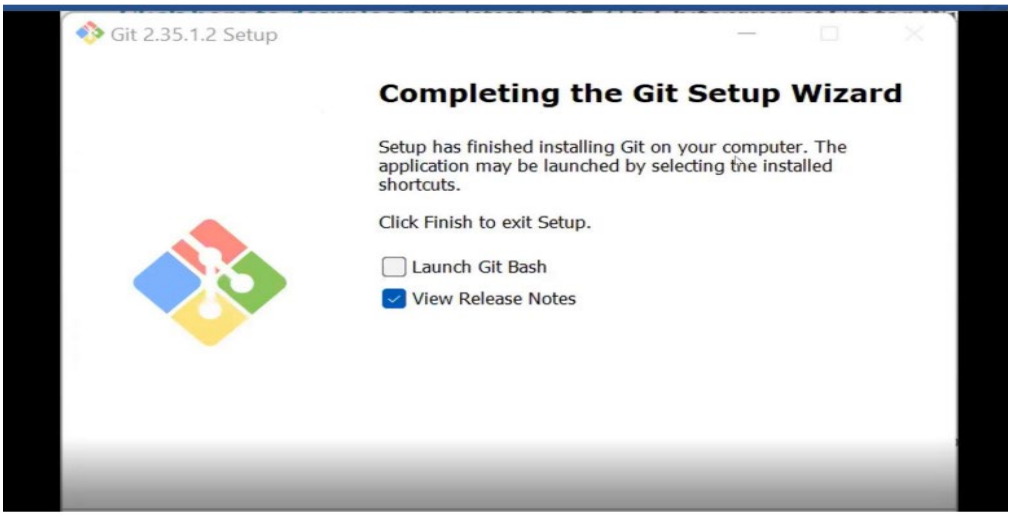
- Then click on Installation Git and click on whatever system you want, available are three- Windows, Apple and Linux.



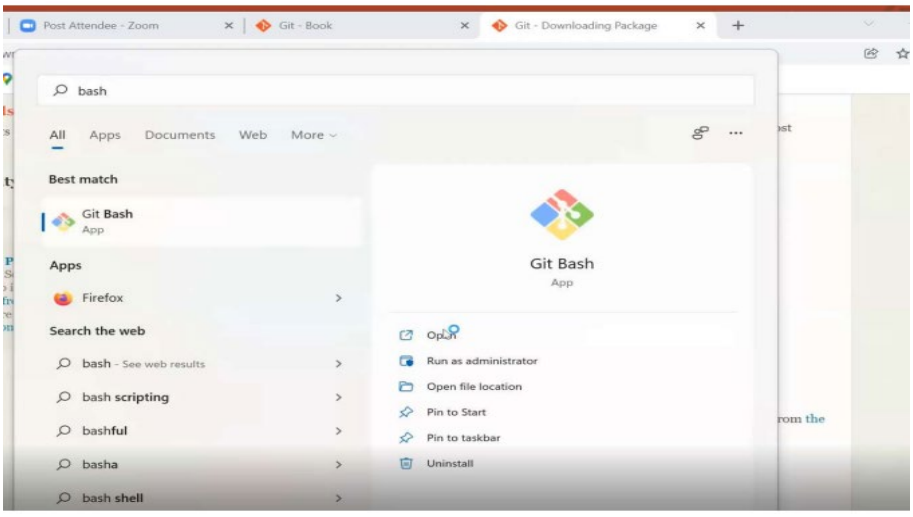


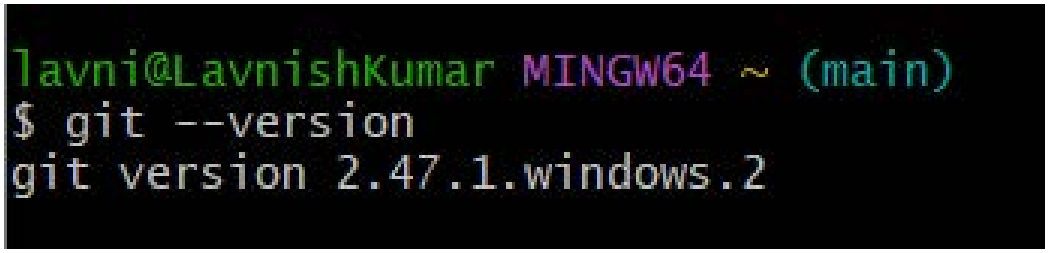
- After some more simple and easy settings and choosing your favourable environment and doing some SSH settings, it finally starts exporting the files in system and completes the Git hub wizard.





- Git bash got installed in system and seemed and opened on clicking seems of like:





You can also check the version of installed software by checking git version.

Experiment 2

Aim: Setting up GitHub Account and Adding Collaborators on GitHub Repository

Theory:

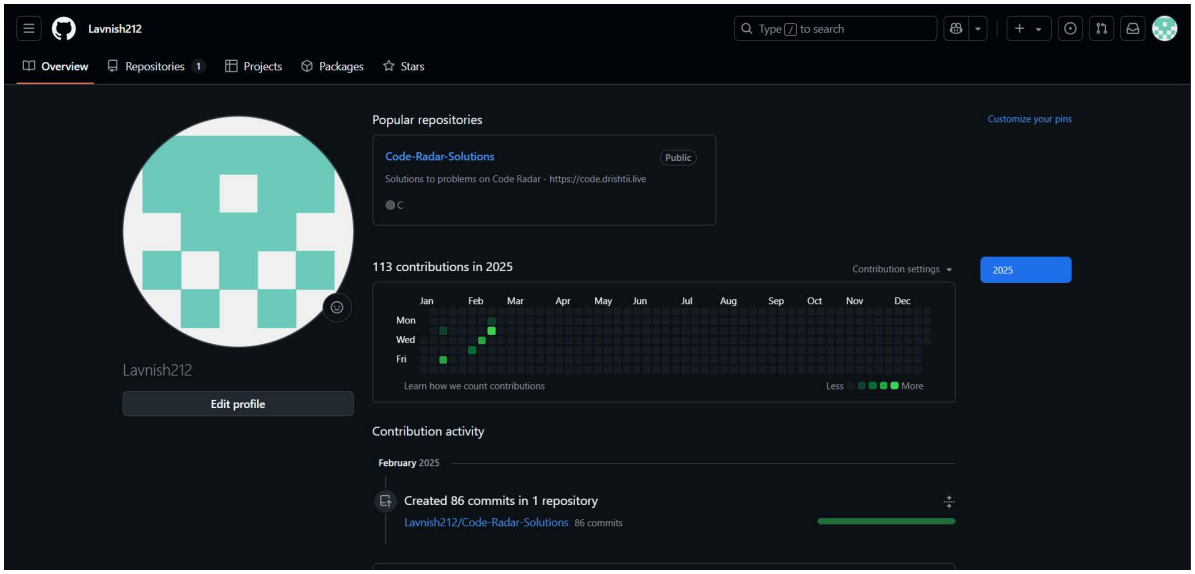
Whenever you make a repository in GitHub, not everyone has the permission to change or push codes into your repository. The users have a read-only access. In order to allow other individuals to make changes to your repository, you need to invite them to collaborate to the project.

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.

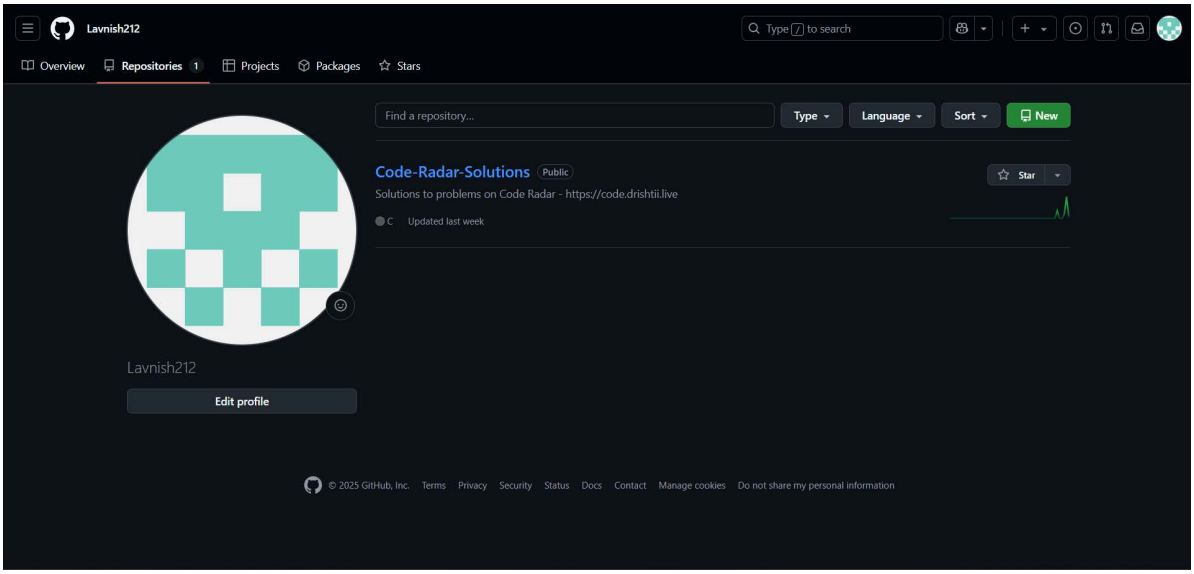
Being a collaborator, the user can create, merge and close pull requests in the repository. They can also remove them as the collaborator.

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.

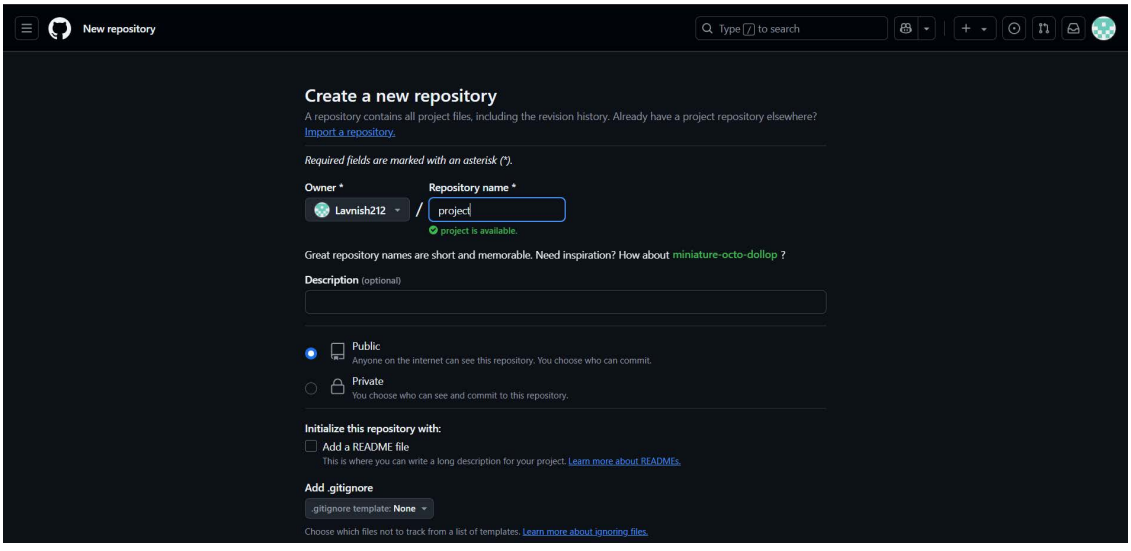


2. Click on the ‘New’ button in the top right corner.

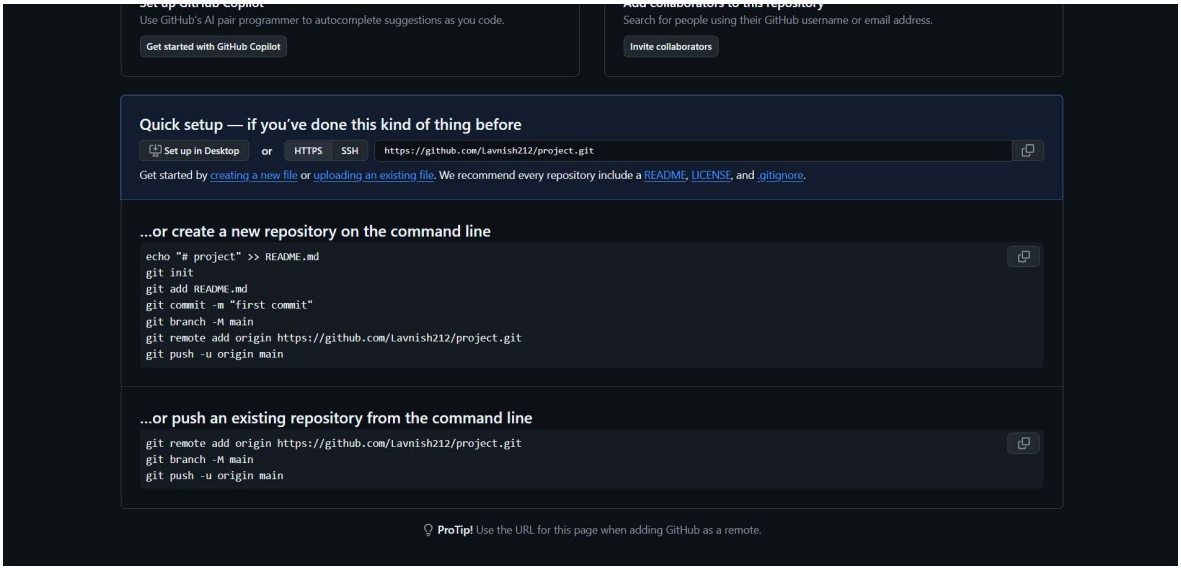


3. Enter the Repository name and add

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

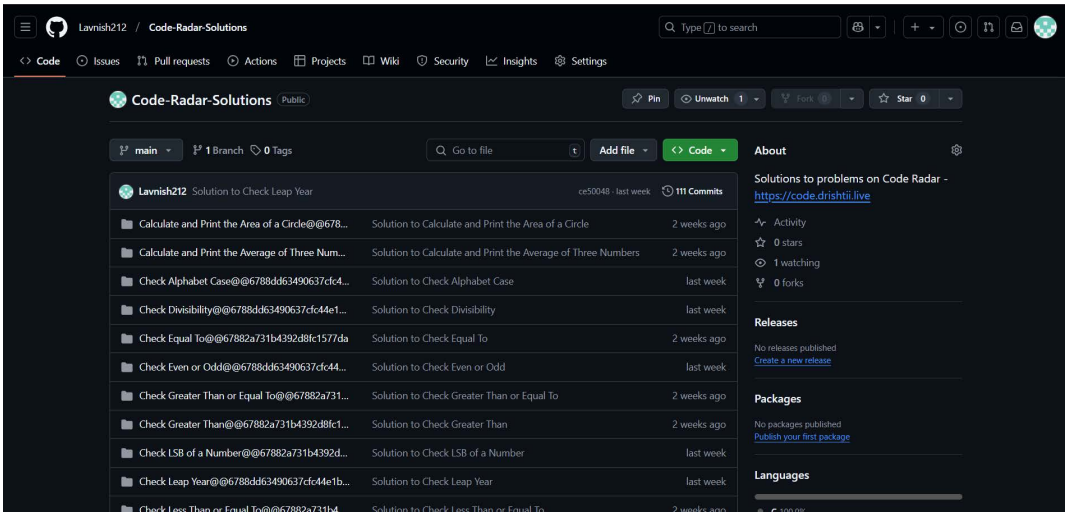


5. If you want to import code from an existing

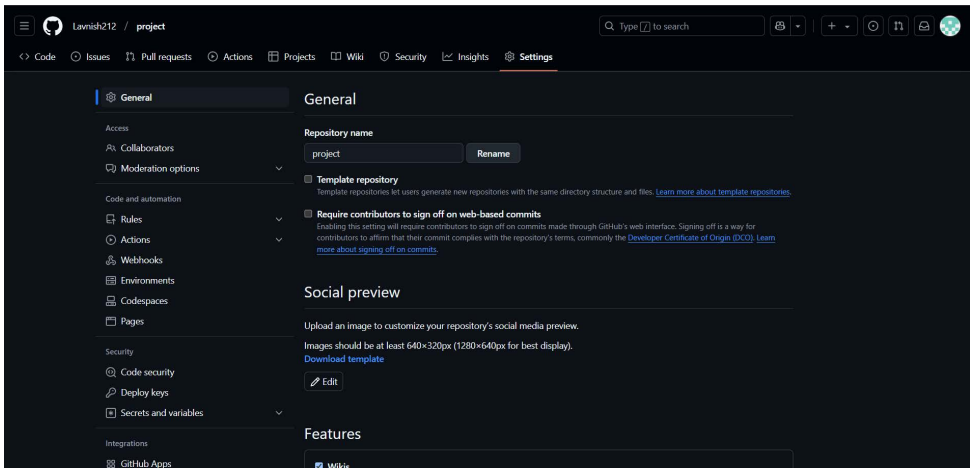


6. Now, you have created your repository successfully.

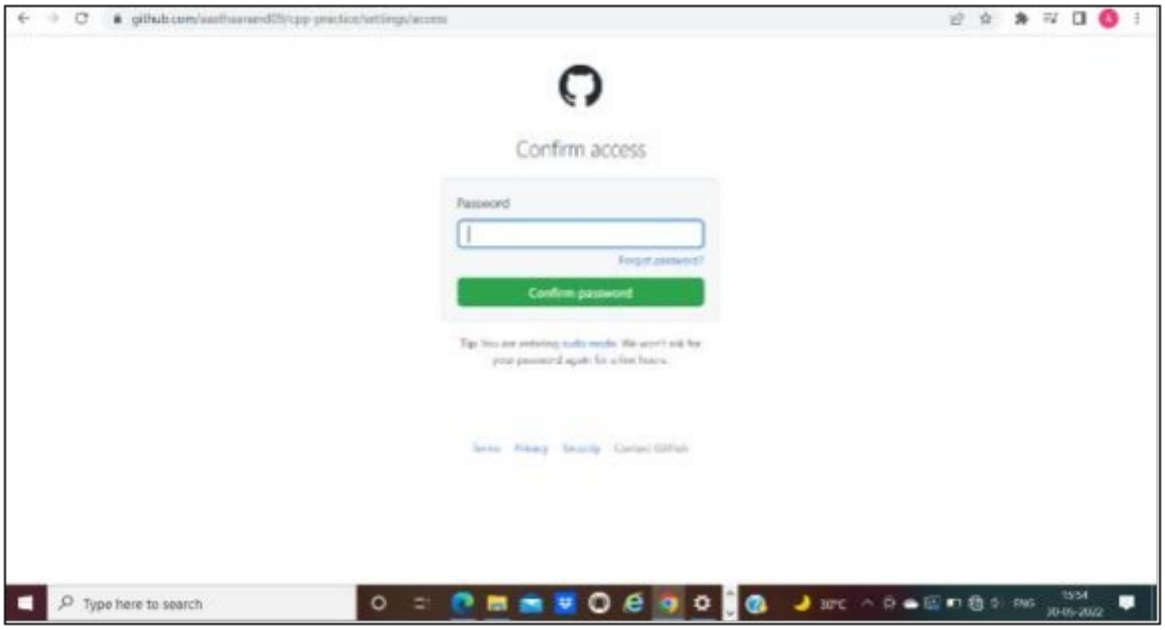
7. To add collaborators to your repository, open your repository and select settings option in the navigation bar.



8. Click on Collaborators option under the access tab.

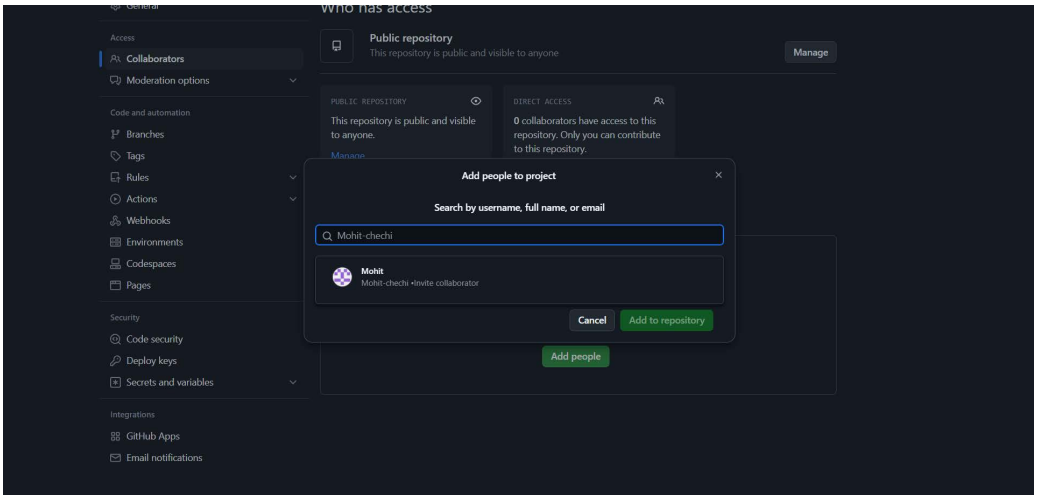


9. After clicking on collaborators, GitHub asks you to enter your password to confirm the access to the repository.

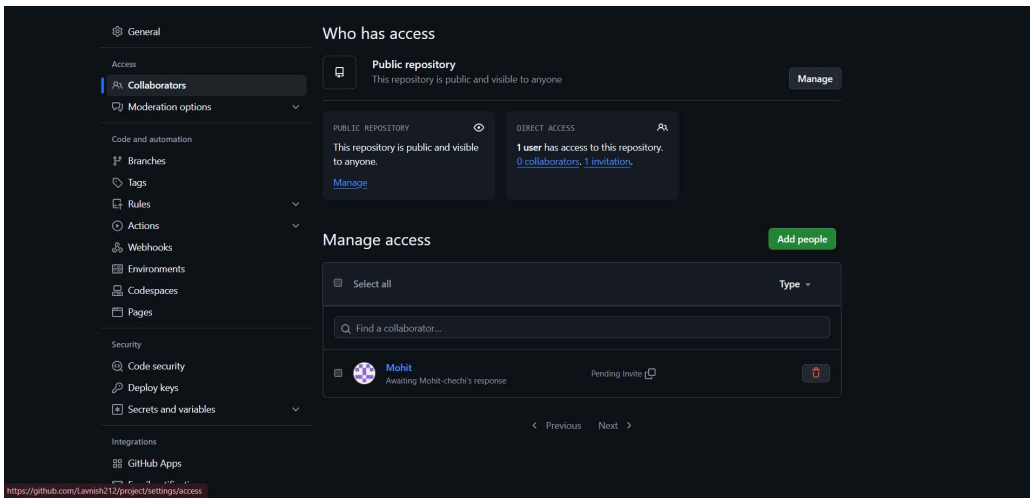


10. After entering the password, you can manage access andadd/remove team members to your project.

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 3

Aim:

To merge two branches within a repository in Git.

Theory:

Branching in Git allows developers to work on different features or bug fixes independently. Once changes are complete, they need to be merged back into the main branch to integrate the new code. Git provides various methods for merging branches, including:

- **Fast-forward merge**: When no new commits are added to the main branch since branching.
- **Three-way merge**: When the main branch has diverged, requiring Git to create a new merge commit.

Procedure:

1. Open a terminal and navigate to the Git repository.
2. Check the existing branches using:

```
lavni@LavnishKumar MINGW64 /c/scm (st2)
$ git branch
  st1
* st2
```

3.Switch to the main branch:

```
lavni@LavnishKumar MINGW64 /c/scm (st2)
$ git checkout main
Switched to branch 'main'

lavni@LavnishKumar MINGW64 /c/scm (main)
$
```

4. Merge the feature branch into the main branch:

```
lavni@LavnishKumar MINGW64 /c/scm (main)
$ git merge st1
Already up to date.
```

5. If there are conflicts, resolve them manually in the affected files.

6. After resolving conflicts, add the changes:

7. Commit the merge:

```
lavni@LavnishKumar MINGW64 /c/scm (main)
$ git commit -m "Merged feature-branch into main"
[main 9ab73a8] Merged feature-branch into main
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 new.html.txt
```

Experiment 4

Aim:

To demonstrate push and pull operations in Git.

Theory:

The push and pull commands are used to synchronize a local repository with a remote repository:

- **Push (git push****)**: Uploads local commits to the remote repository.
- **Pull (git pull****)**: Fetches the latest changes from the remote repository and merges them into the current branch.

Procedure:

1. **Pushing to a remote repository:**

1.1 Ensure you are in the correct branch:

```
lavni@LavnishKumar MINGW64 /c/scm (main)
$ git branch
* main
  st1
  st2
```

1.2 Push the branch to the remote repository:

```
lavni@LavnishKumar MINGW64 /c/scm (main)
$ git  push origin main
info: please complete authentication in your browser...
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 12 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (7/7), 518 bytes | 172.00 KiB/s, done.
Total 7 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/Lavnish212/project
 * [new branch]      main -> main
```

2.Pulling updates from a remote repository:

2.1 Fetch the latest changes:

2.2 If conflicts arise, resolve them manually and commit the changes.