Subject name: Source Code Management

Subject name: **Source code management**

Subject code: **CS181**

Cluster: **BETA**

Department: **DCSE**

Subject code: CS181

Cluster: Gamma

Department: DCSE

**Submitted by: Submitted to:** Danish Garg Dr. Monit Kapoor

2110990407

G08-B



**Department of Computer Science & Engineering**

Chitkara University Institute of Engineering and Technology, Punjab

**Table of Content**

|  |  |  |
| --- | --- | --- |
| **S. No.** | **Title** | **Page No.** |
| 1 | Version control with Git | 1-13 |
| 2 | Project | 14-15 |
| 3 | Task 1.2 | 16-24 |

1. Version control with Git

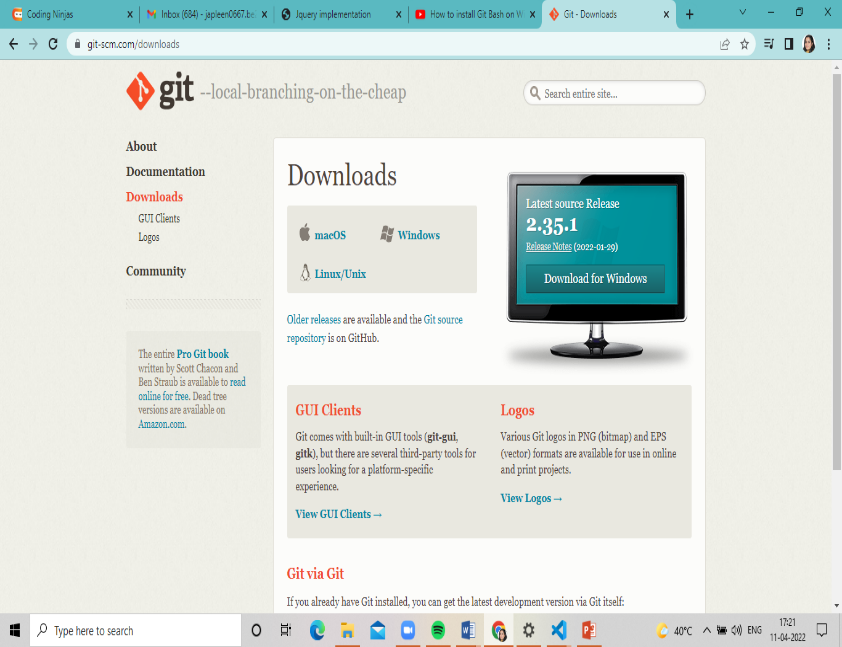
**Step 1 - Installing Git:**

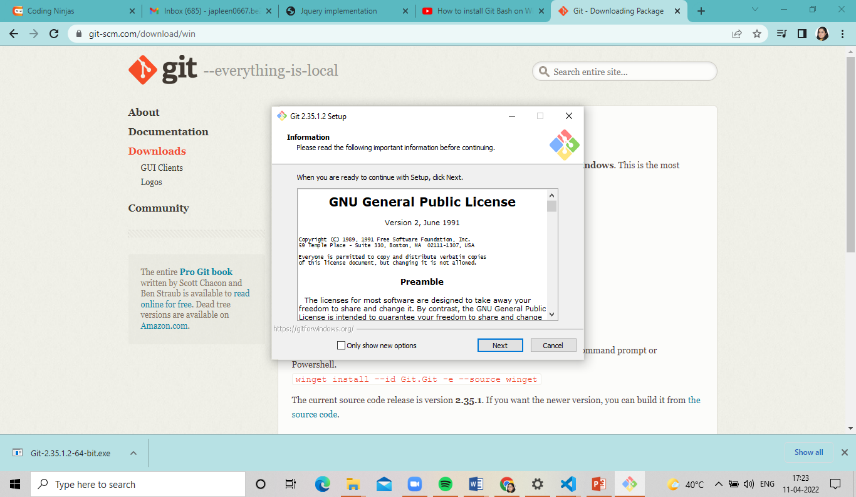
Before we start using Git, we have to make it available on your computer. Even if it’s already

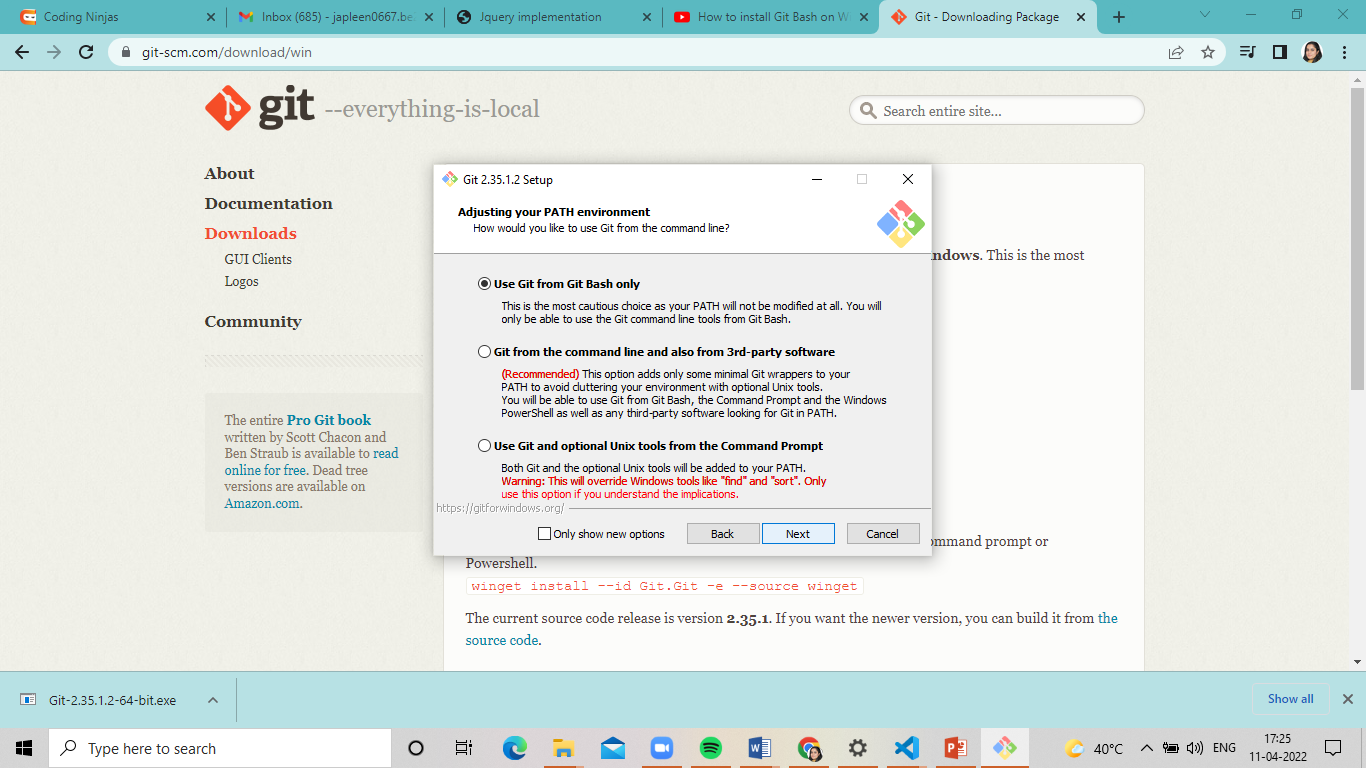
Installed , it’s probably a good idea to update to the latest version. We can either install it as a package or via another installer, or download the source code and compile it yourself.

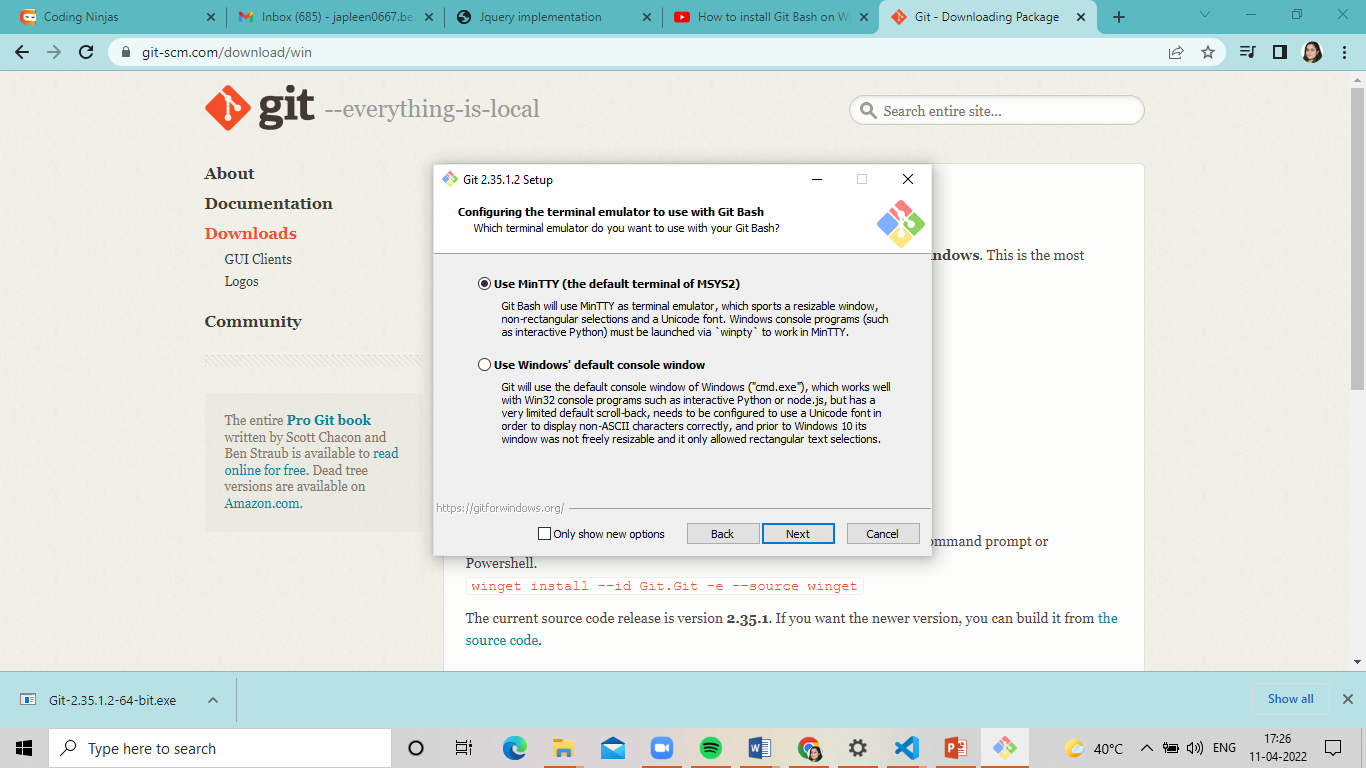
**Step 2 - Installing on windows:**

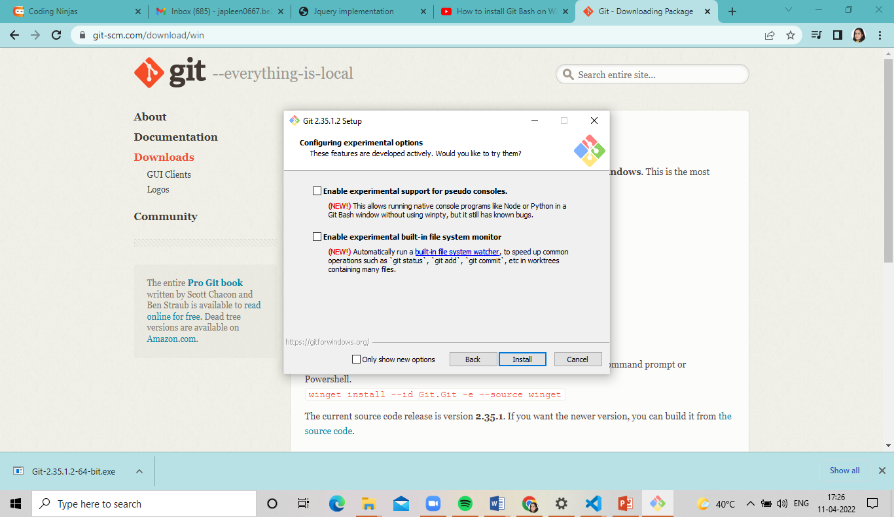
There are also a few ways to install Git on Windows. The most official build is available for download on the Git website. Just go to https://git-scm.com/download/win and the download will start automatically. Note that this is a project called Git for Windows, which is separate from Git itself; for more information on it, go to <https://gitforwindows.org>.

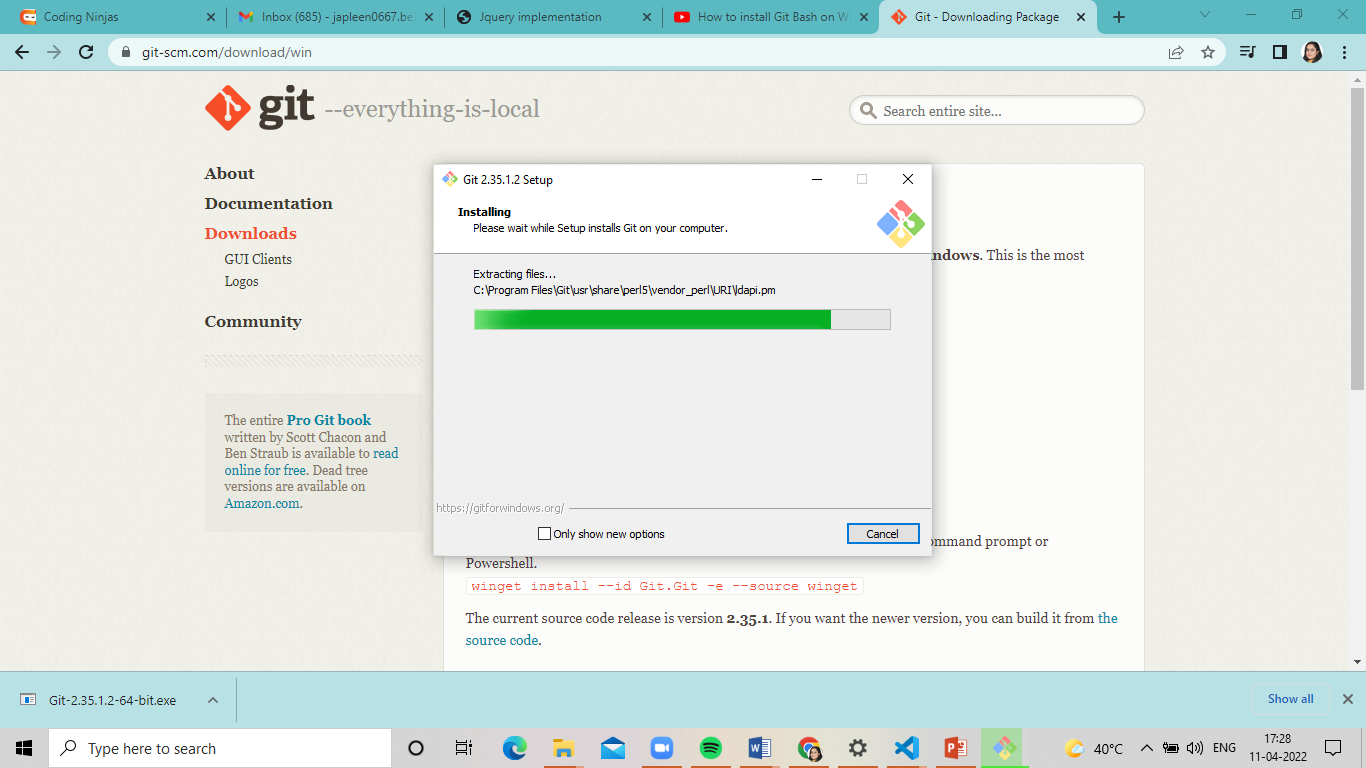
**1.**

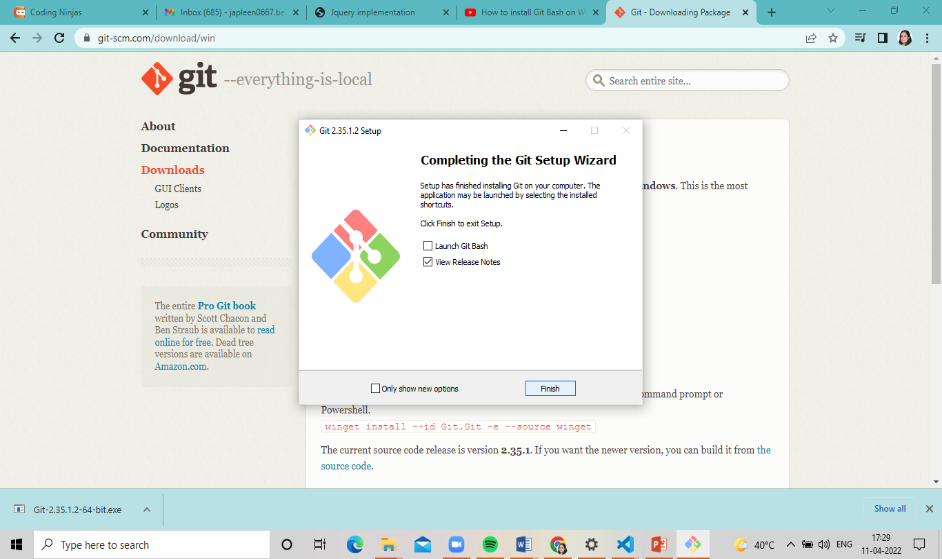
2.

3.

4.

5.

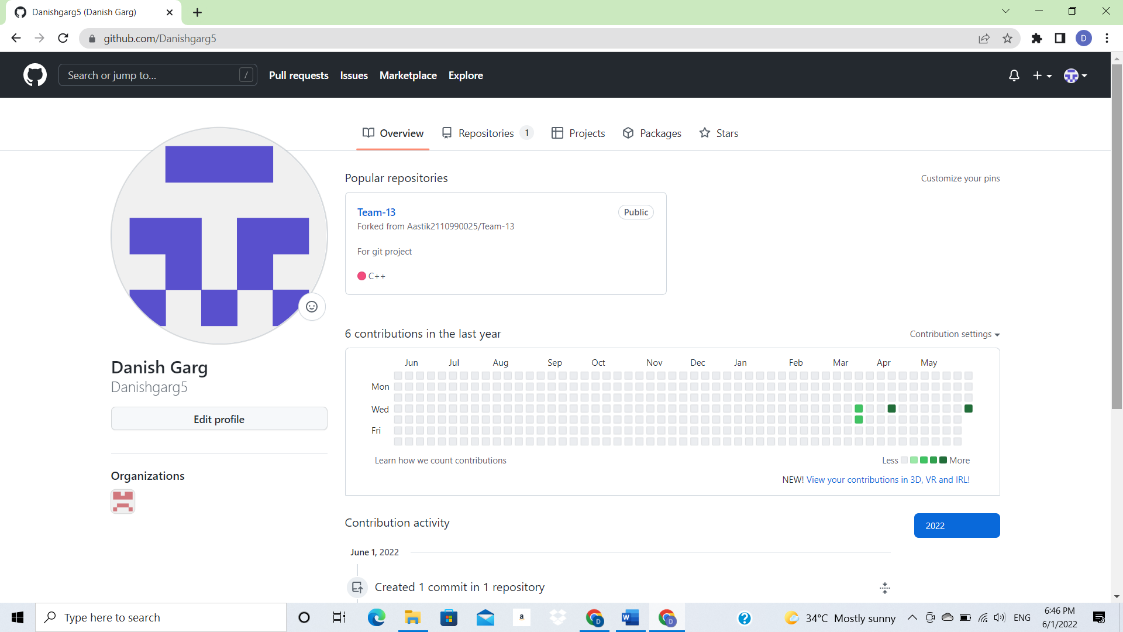
6.

7.

Step 3 – Setting up GitHub account:

1.

2.



3. Configuration of Git:

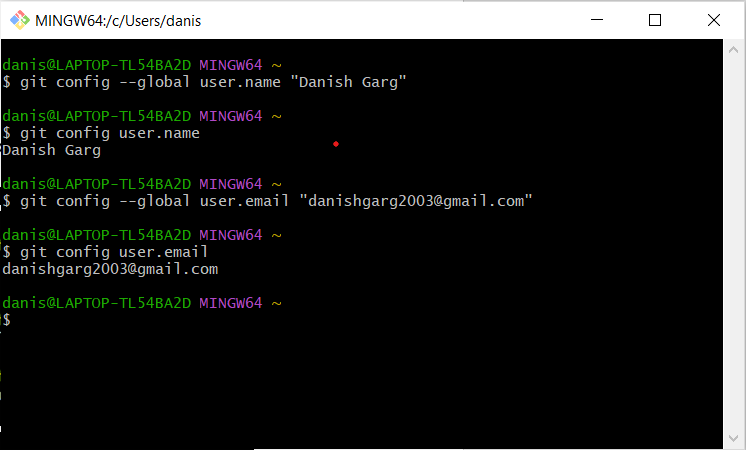
3. Configuration of Git:

**Step1)**

You can configure your Git by typing: -

1. Set your username: git config --global user.name "Your Name"
2. Set your email address: git config --global user.email "Your Email

The page looks like as: -

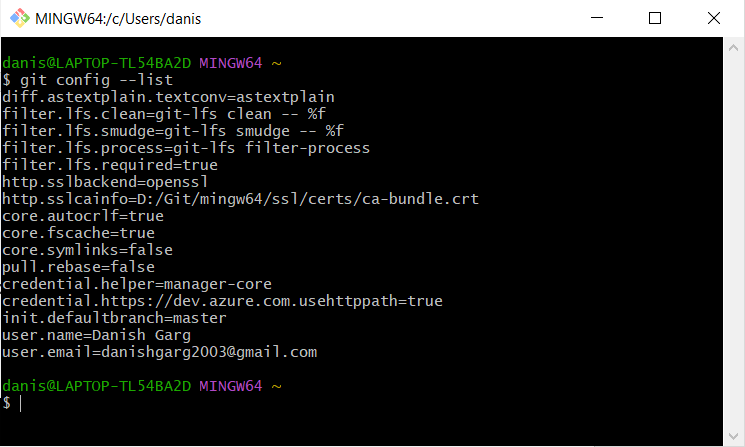


**Step2)**

**You can check configuration of Git by typing -**

1. **git config --list**

**The page looks like as: -**

****

4. Program to generate Logs:

**Program to Generate logs:**

**Advantage of version control systems like git is that it can record changes.**

**‘Git log’ command is used to display all these changes that were made by the user in the repository. Log is a record of all the previous commits.**

**To understand Logs we need to get familiar with all the commands that are used in making changes to a repository.**

1. **Repository:  A repository is a directory that contains all the project-related data.**
2. **Git init: The git init command is used to create a new blank repository.**
3. **Git status: We can list all the untracked, modified and deleted files using the git status command.**
4. **Git add: Adds all the untracked and modified files to the staging area.**
5. **Git commit: Git commit finalizes the changes in the repository. Every commit is saved in the branch you are working in and can be used to revert back to older versions of the project.**

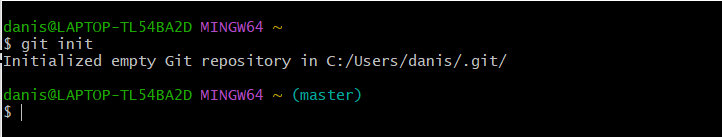
**Making GIT Repository**

**Step1: GIT INIT**

**Initializing a new repository, You Can do it by typing: -**

1. **git init**

**The page looks like as: -**

****

**Step2: ADDING THE FILES TO THE FOLDER**

**Just like (Samples...)**

**The page looks like as: -**

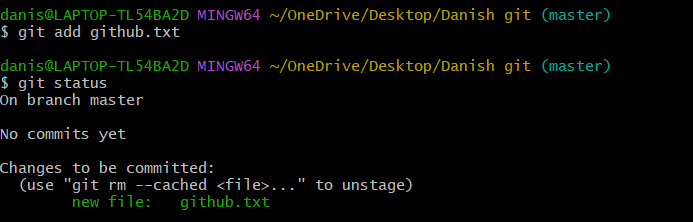
**Step3: GIT ADD**

**The git add command adds a change in the working directory to the staging area.**

**You Can do it by typing: -**

1. **git add**
2. **git add (current file name)**

**The page looks like as: -**

****

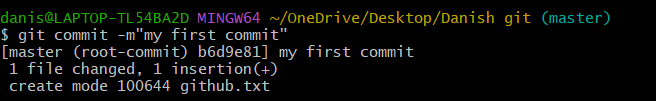
**Step4: GIT COMMIT**

**The "commit" command is used to save your changes to the local repository.**

**You Can do it by typing: -**

1. **git commit -m”any text”**

**The page looks like as: -**

****

**Step5: GIT LOG**

**Git log will show all the commits made by the author with their time.**

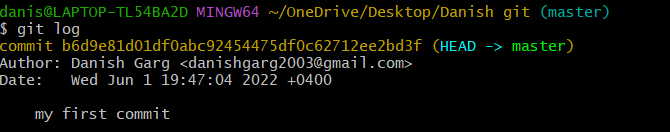
**After every commit the checksum value (written In yellow color) of the folder changes.**

**Checksum is used to verify that the data in that file has not been tampered with or manipulated, possibly by a malicious entity.**

**You Can do it by typing: -**

1. **git log**

**The page looks like as: -**

****

**Step6: Create and Visualize Branches**

**A branch in Git is simply a lightweight movable pointer to one of these commits. The default branch name in Git is master.**

5. Create and Visualize Branches

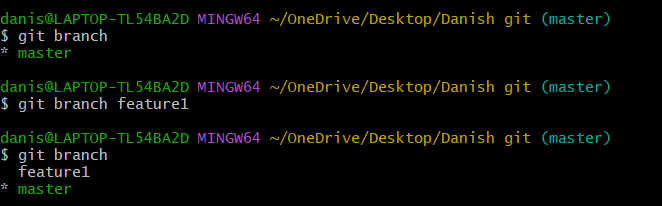
**Step1: CHECKING UP THE BRANCHES**

**You can check which branch you are working in by using the command**

**1.’git branch’.**

**The default branch is always the master branch.**

**The page looks like as: -**

****

**Step3: CHANGING BRANCHES**

**To switch to the other branch**

**You Can do it by typing: -**

1. **git checkout (BRANCH NAME)**

**The page looks like as: -**

****

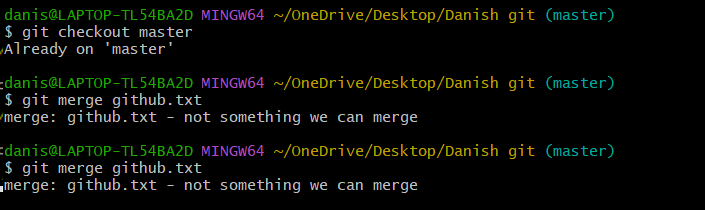
**Step6: GIT MERGING**

**Now you can merge two branches by command.**

1. **git merge (BRANCH NAME)**

**If you want to merge a new branch in master branch you need to first checkout into the master branch and then run the command.**

**The page looks like as: -**

****

6. Git Lifecycle Description:

**There are three stages for git lifecycle:**

1. **Working directory**
2. **Staging area**
3. **Git repository**

**Working Directory:**

**The working directory is the folder in your local computer where the project files and folders are stored.**

**The local directory is created by the command ‘git init’ which creates a ‘.git’ named folder which is used to track the files in the directory.**

**‘.git folder’ is generally hidden but can be tracked enabling hidden files.**

****

**Staging area:**

**The staging area has those files which are supposed to go to the next commit. Only those files which are needed to go to the next commit stay in the staging area.**

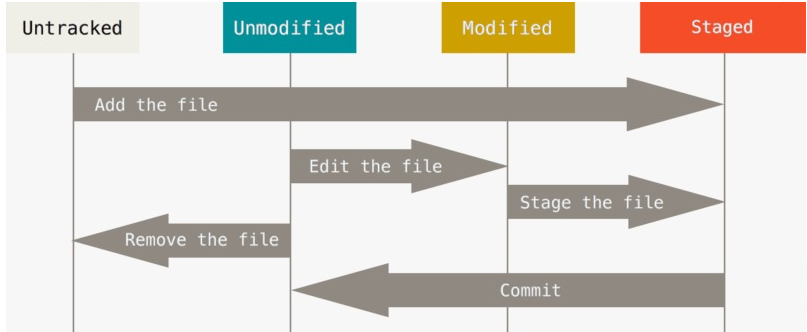
**You can shift the files to the git repository by using the command**

**‘git add --a’.**

**Git repository:**

**Now since we have all the files that are to be tracked and are ready in the staging area, we are ready to commit our files using the git commitcommand. Commit helps us in keeping the track of the metadata of the files in our staging area. We specify every commit with a message which tells what the commit is about.**

**You can commit files by using command ‘git commit -m “message”’**



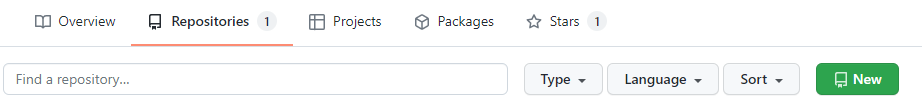
7. Uploading data on github

**NOTE-**

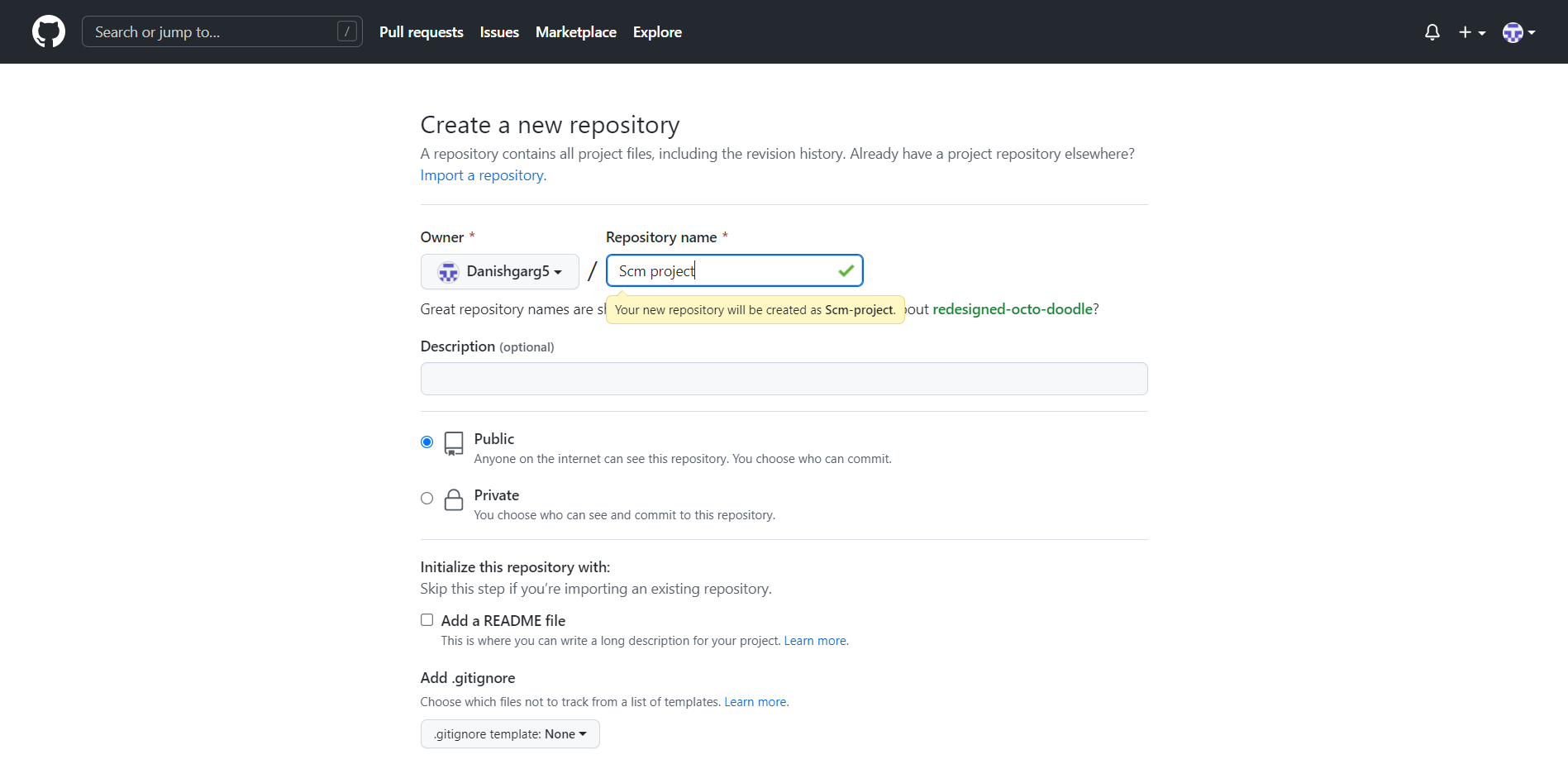
**YOU HAVE TO MAKE A REPOSITORY IN GITHUB.**

**Step1) CREATING REPOSITORY IN GITHUB**

**The page looks like as: -**



**By clicking on new you are able to make a new repository.**

****

**Write the repository name and click on next.**

****

**Your GITHUB Repository has been created.**

**Step2) GIT ADDING REMOTE BRANCH**

**Git stores a branch as a reference to a commit, and a branch represents the tip of a series of commits.**

**You Can do it by typing: -**

1. **git branch -M main**

**The page looks like as: -**

****

**Step3) GIT ADDING REMOTE ORIGIN**

**Is a Git repository that's hosted on the Internet**

**You Can do it by typing: -**

1. **git remote add origin (URL)**

**The page looks like as: -**



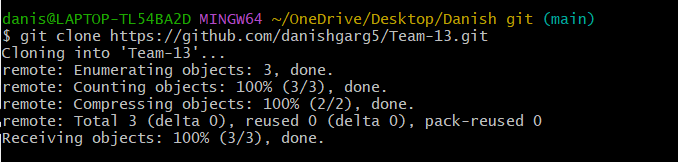
**Step4) GIT CLONE**

**The git clone command is used to upload local repository content to a remote repository.**

**You Can do it by typing: -**

1. **git clone URL**

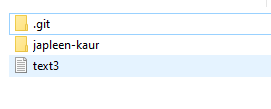
**The page looks like as: -**

****

**Final Result**

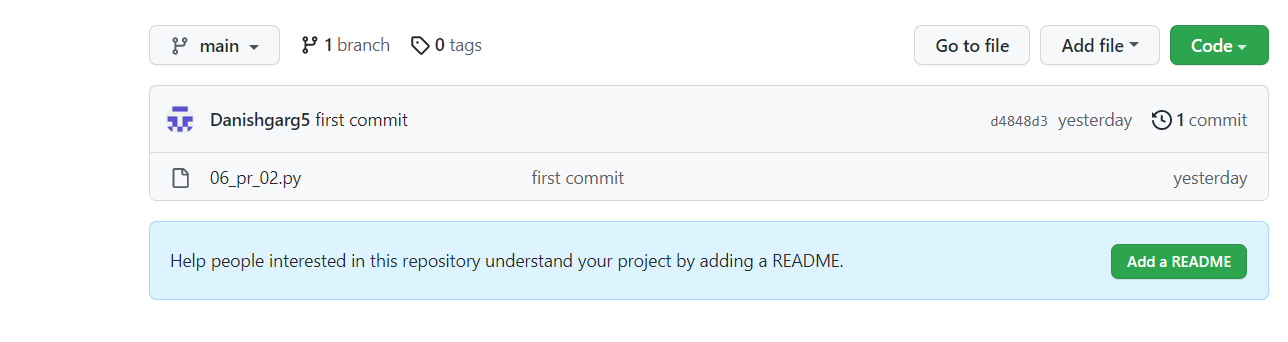
1. **Document in your system**

**The page looks like as: -**



2. **Document in your GITHUB Repository**

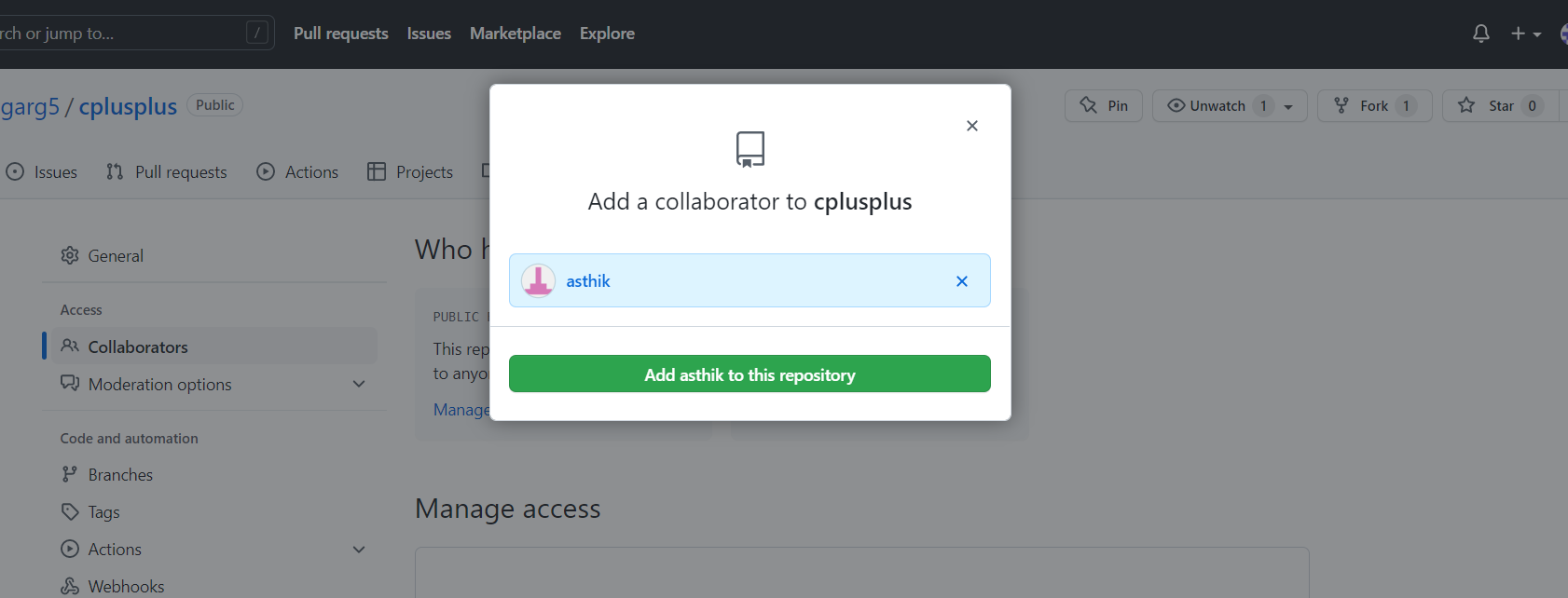
**The page looks like as: -**

****

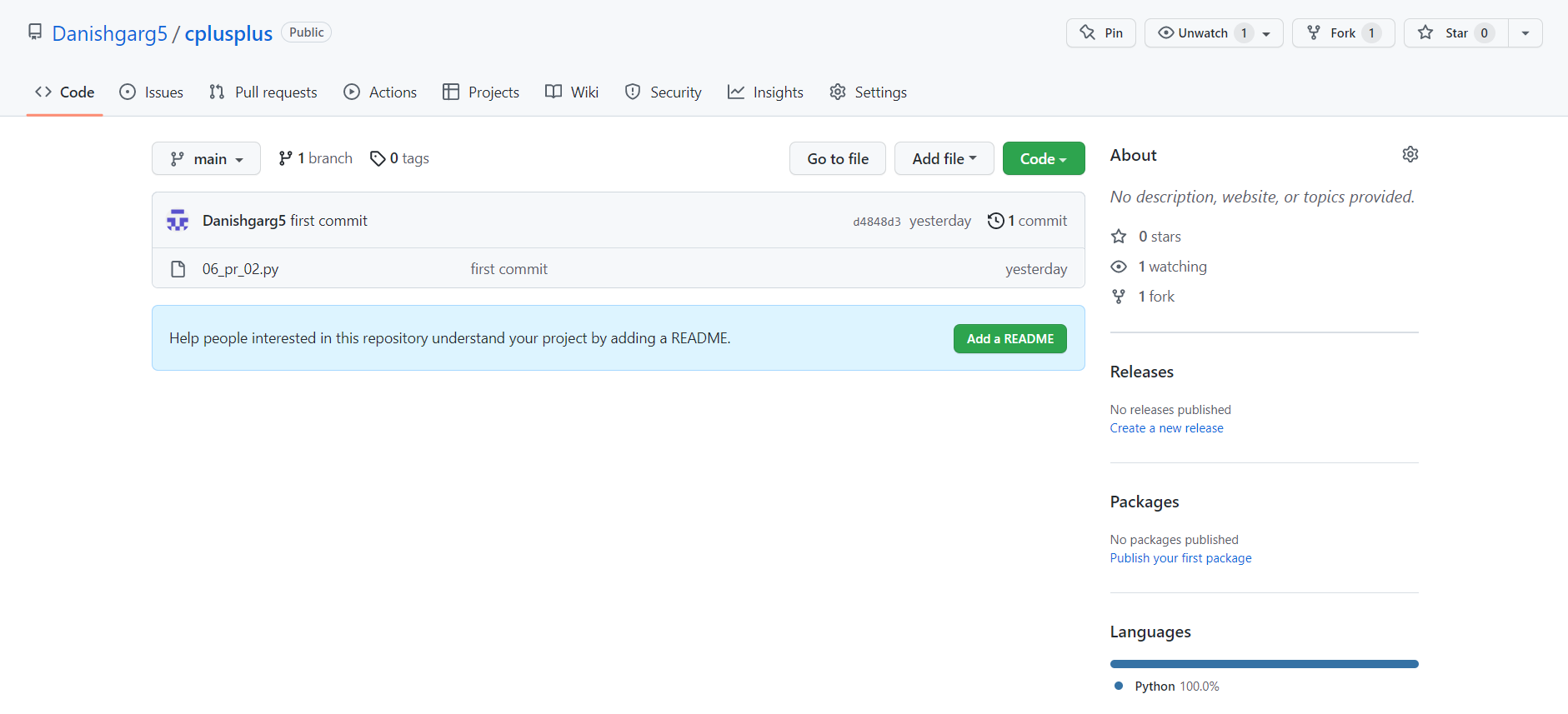
Task 1.2

**1. Add Collaborators on GitHub Repository: -**

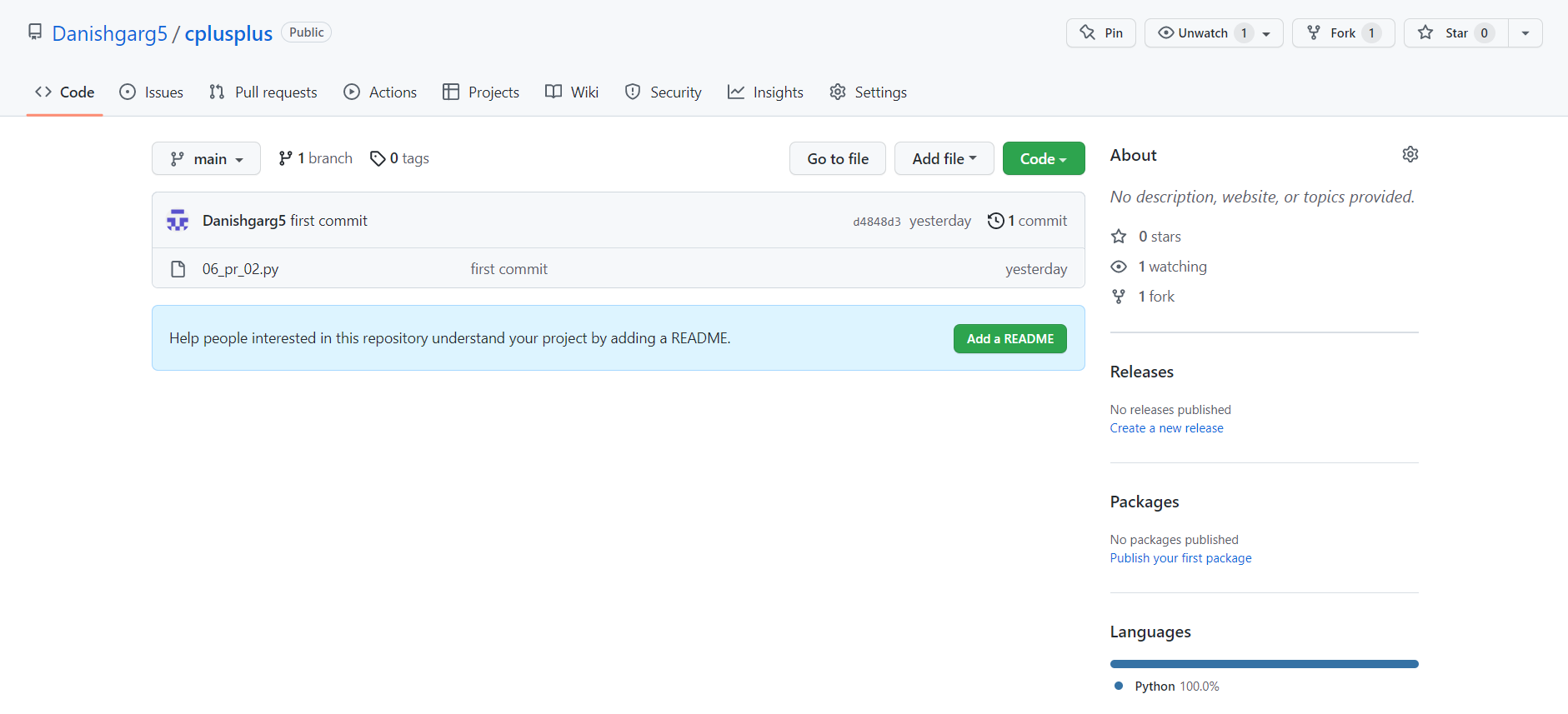
1. Ask for the username of the person you're inviting as a collaborator. ...



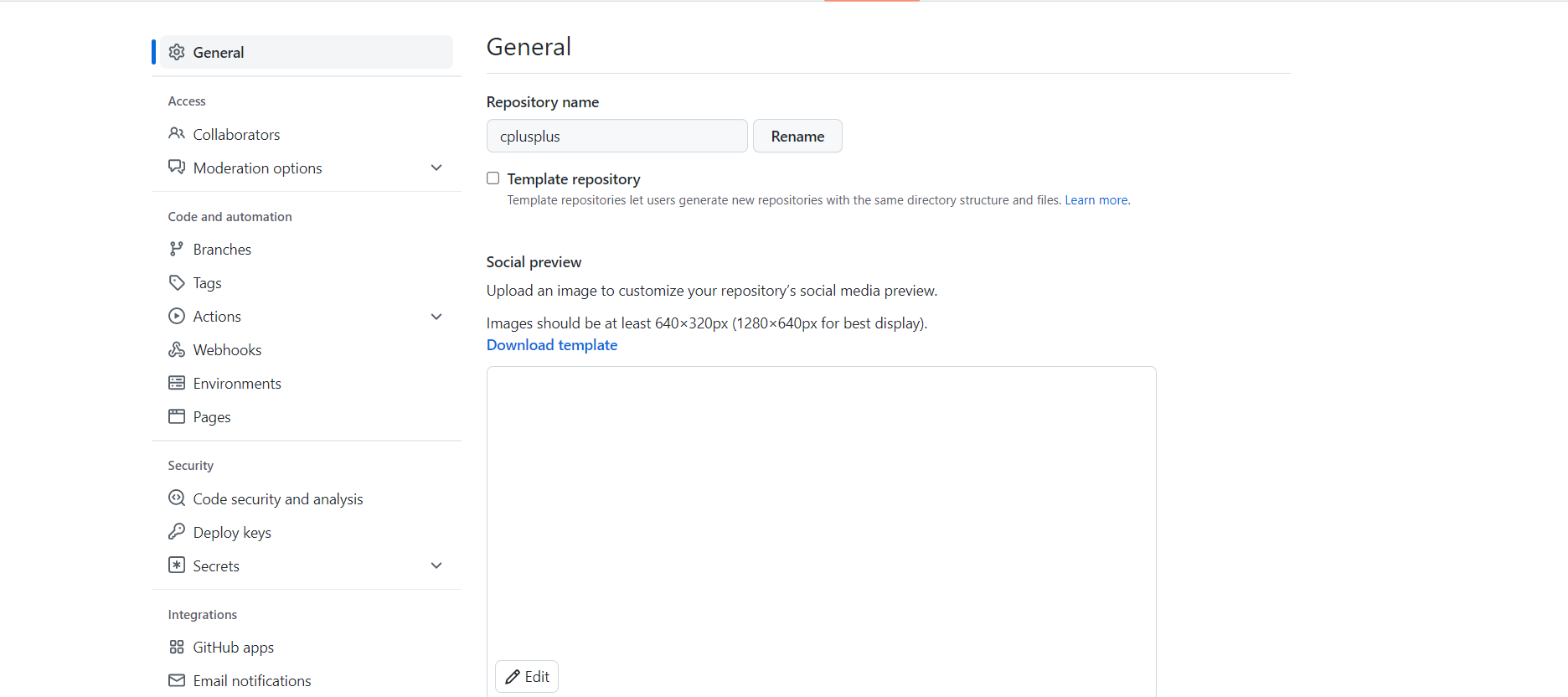
2. On GitHub.com, navigate to the main page of the repository.



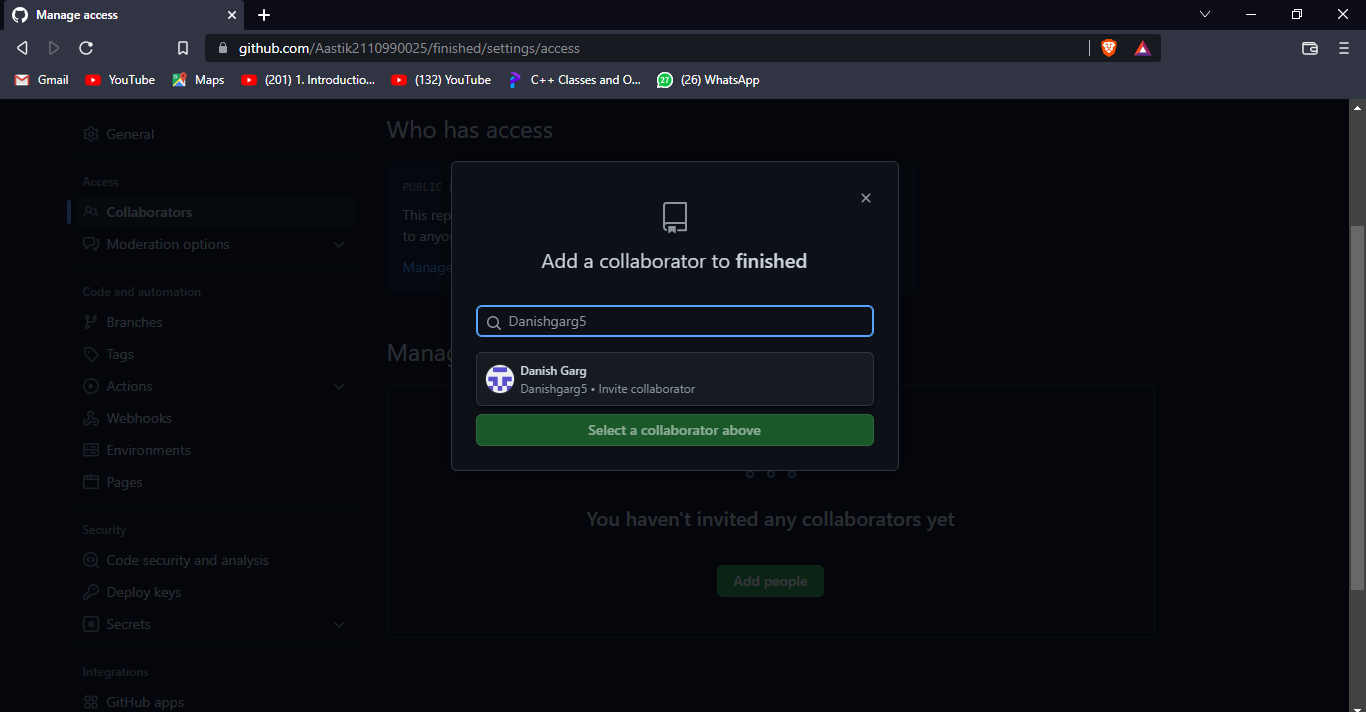
3. Under our repository name, click Settings

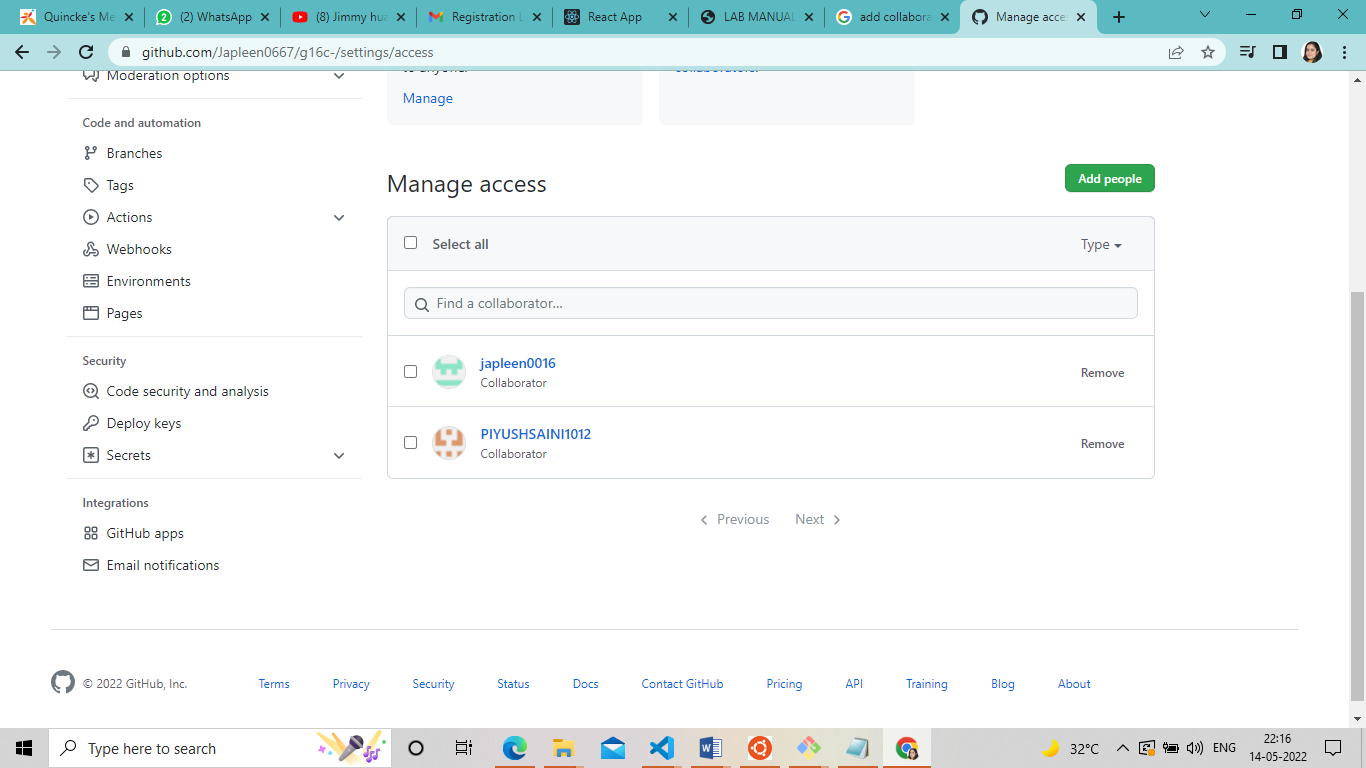


4. In the "Access" section of the sidebar, click Collaborators & teams.



5. Click Invite a collaborator.

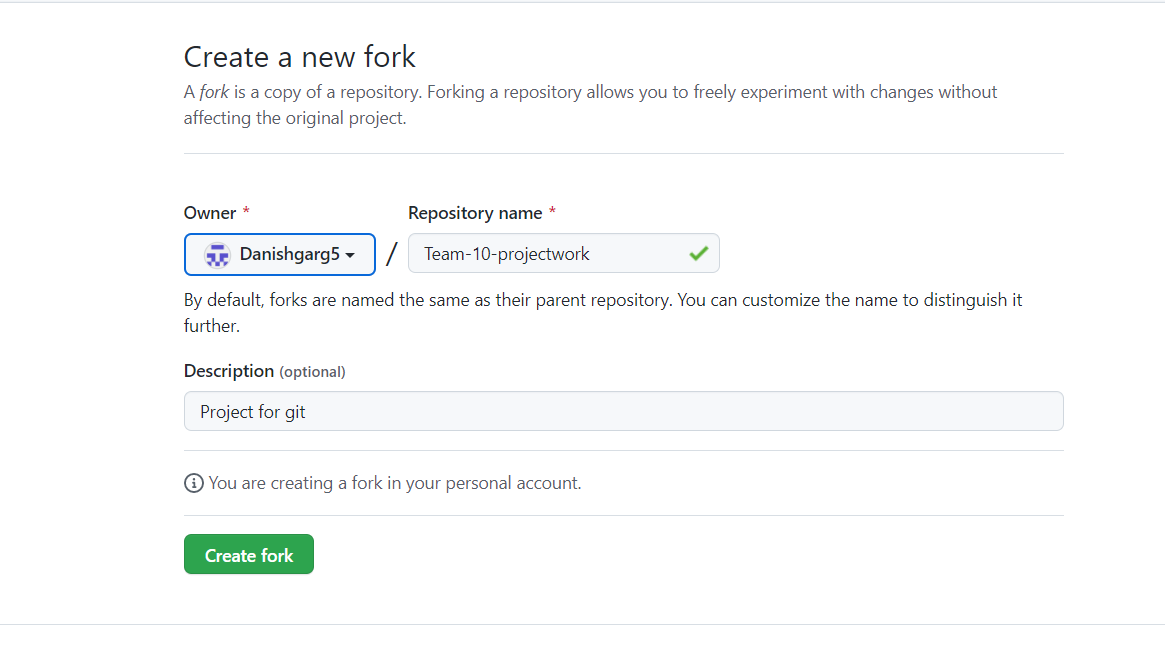


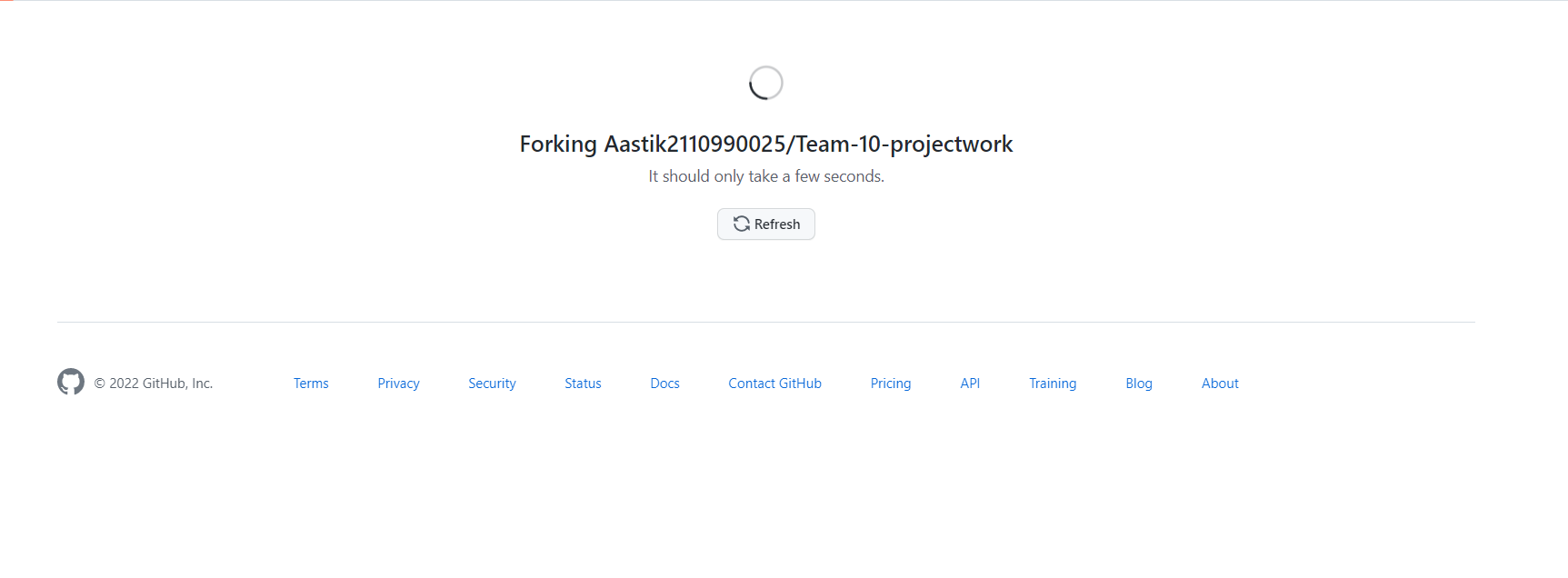


**2. Fork And Commit :-** A fork is a copy of a repository that we manage.

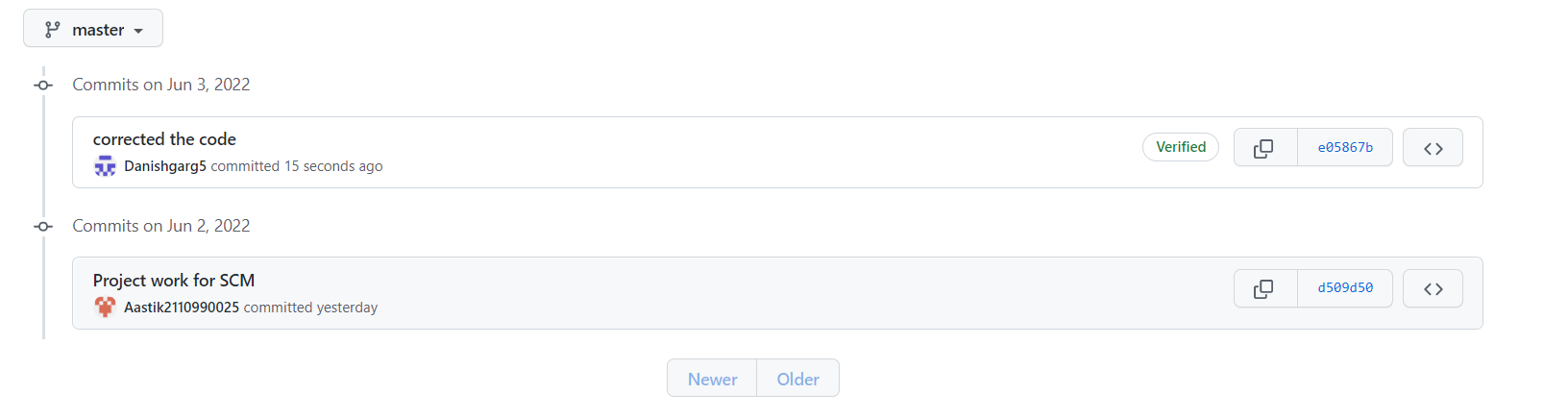
Forks let us make changes to a project without affecting the original repository. We can fetch updates from or submit changes to the original repository with pull requests.

If we need to fork a GitHub or GitLab repo, it’s as simple as navigating to the landing page of the repository in your web browser and clicking on the *Fork* button on the repository’s home page. A forked copy of that Git repository will be added to your personal GitHub or GitLab repo. That’s it. That’s all we have to do to fork a Git repo.









**3.** **Merge and Resolve conflicts created due to own Activity and Collaborators activity:-**

There are a few steps that could reduce the steps needed to resolve merge conflicts in Git.

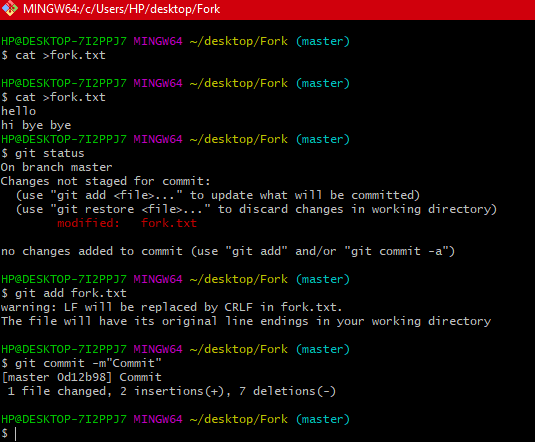
1. The easiest way to resolve a conflicted file is to open it and make any necessary changes.

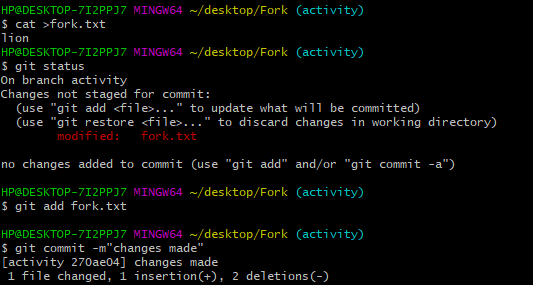
2. After editing the file, we can use the git add a command to stage the new merged content.

3. The final step is to create a new commit with the help of the git commit command.

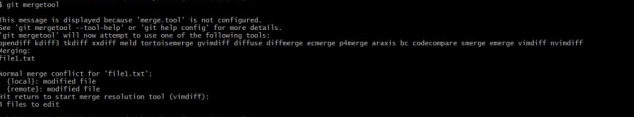
4. Git will create a new merge commit to finalize the merge

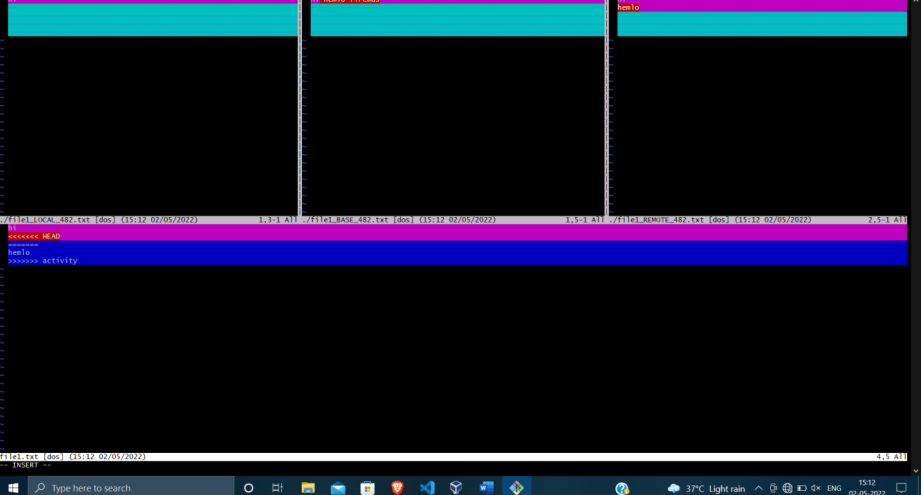
Let us now look into the Git commands that may play a significant role in resolving conflicts.







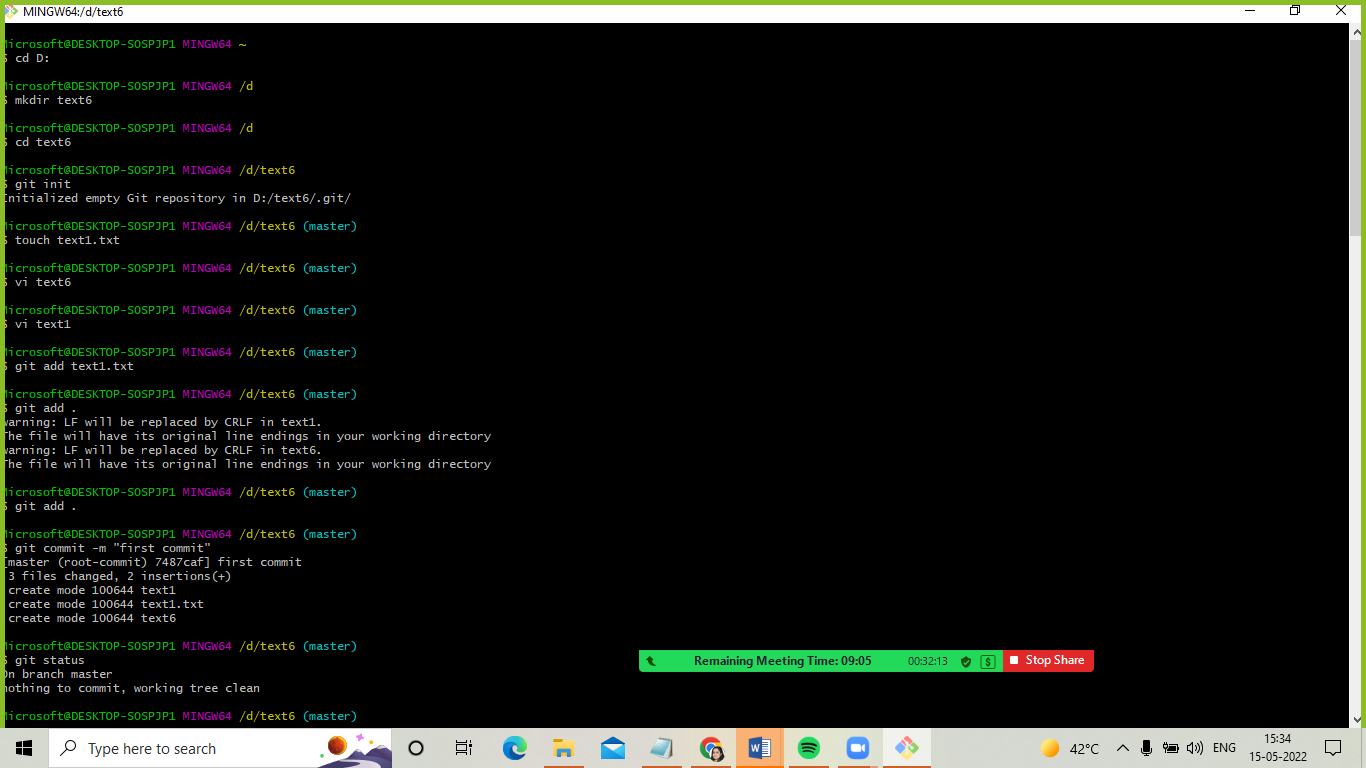


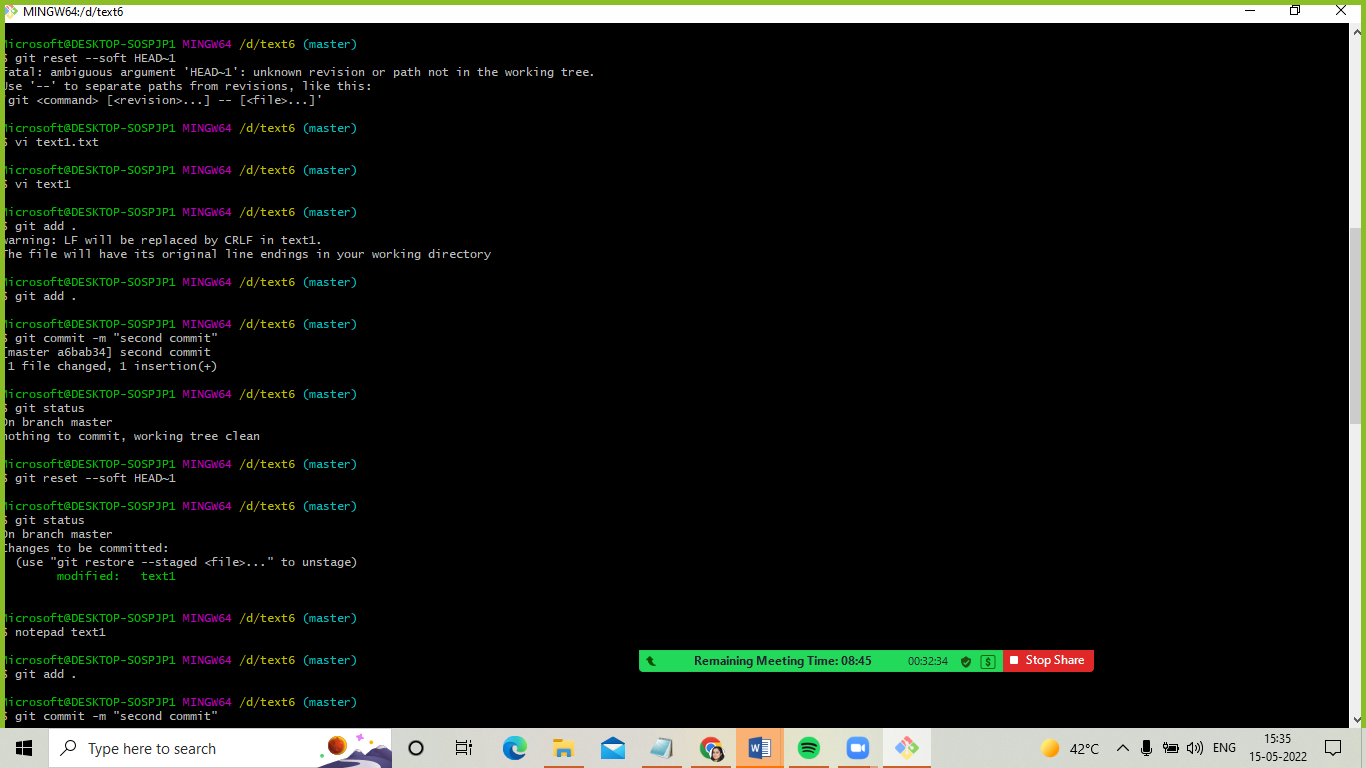


**4. Git Reset :-** Git reset is a powerful command that is used to undo local changes to the state of a Git repo. Git reset operates on "The Three Trees of Git". These trees are the Commit History (HEAD), the Staging Index, and the Working Directory.

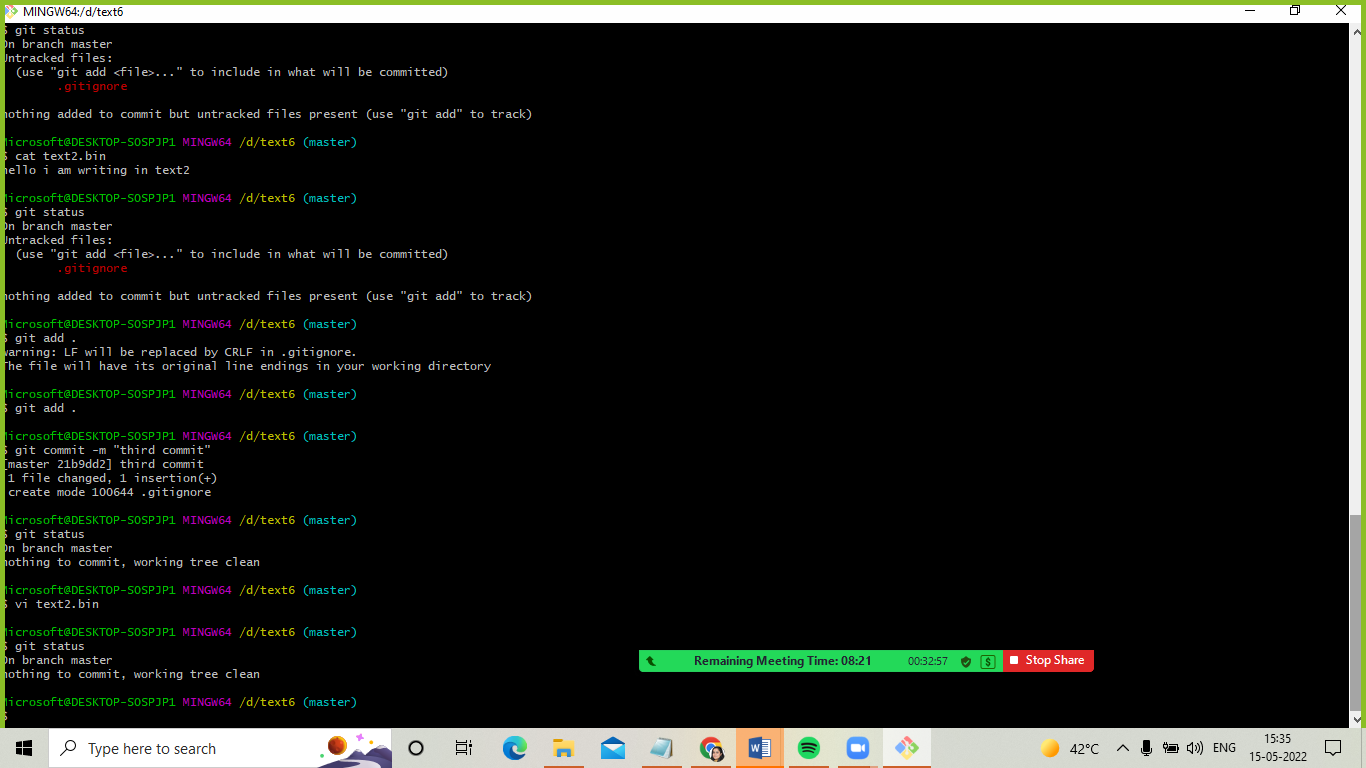
The easiest way to undo the last Git commit is to execute the “git reset” command with the “–soft” option that will preserve changes done to your files.

Git reset --hard , which will completely destroy any changes and remove them from the local directory.





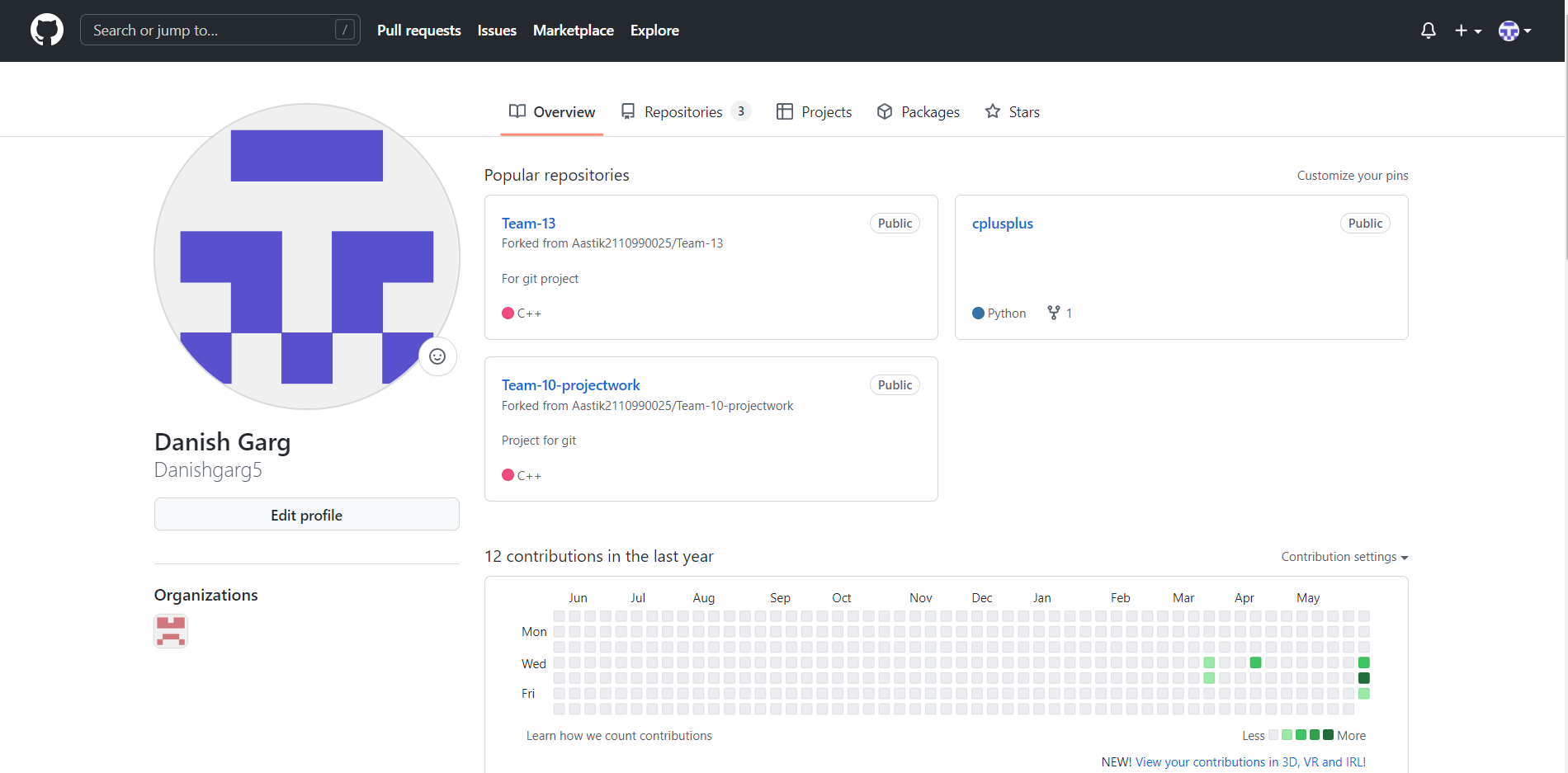




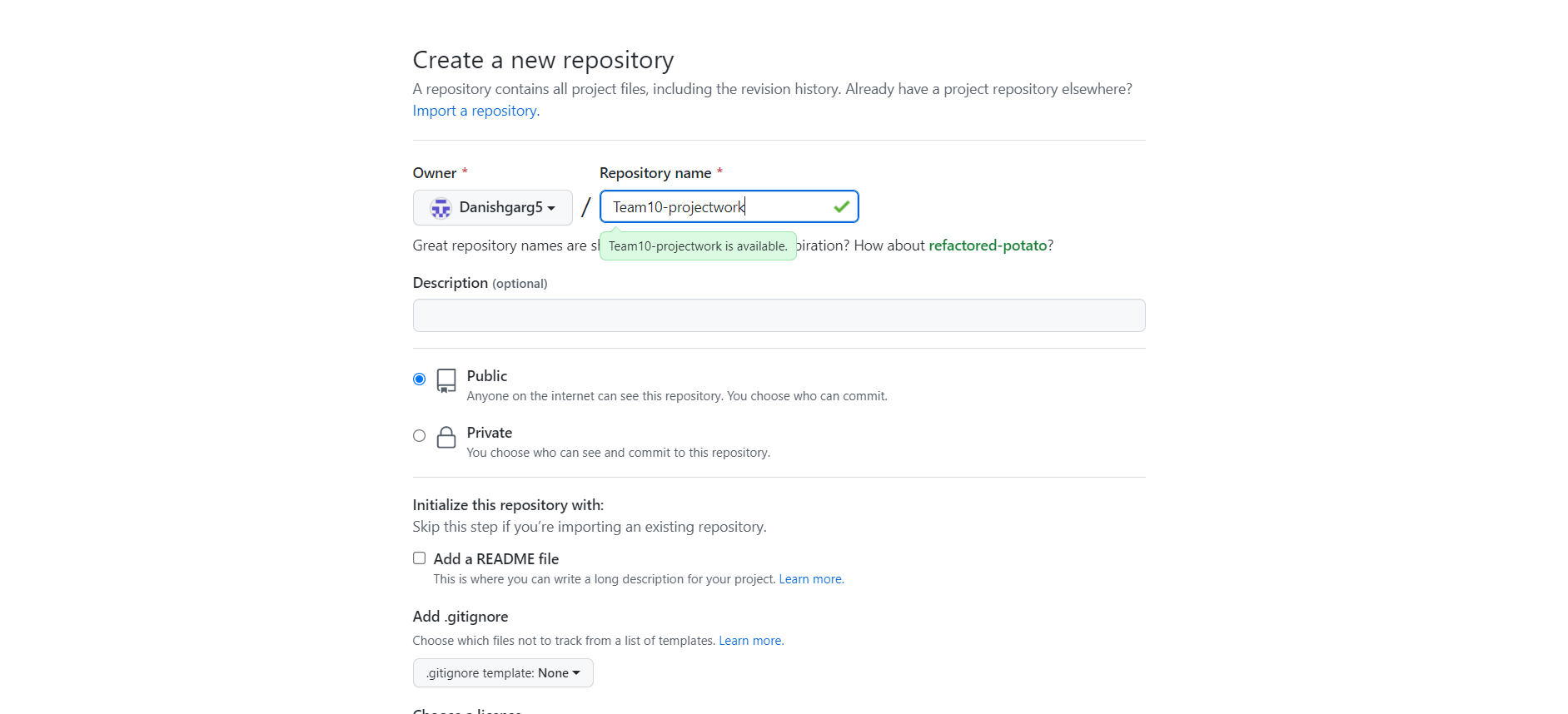
PROJECT WORK

TASK-2

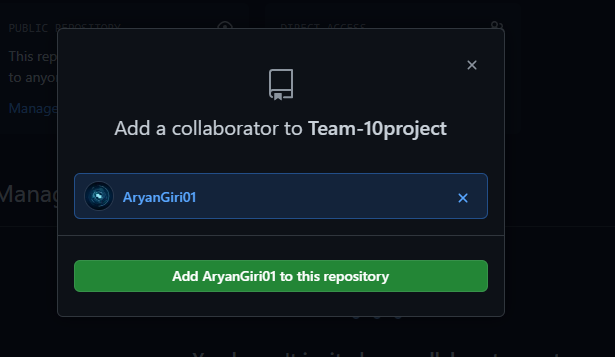
1. Login to your GitHub account and you will land on the homepage as shown below. Click on Repositories option in the menu bar.



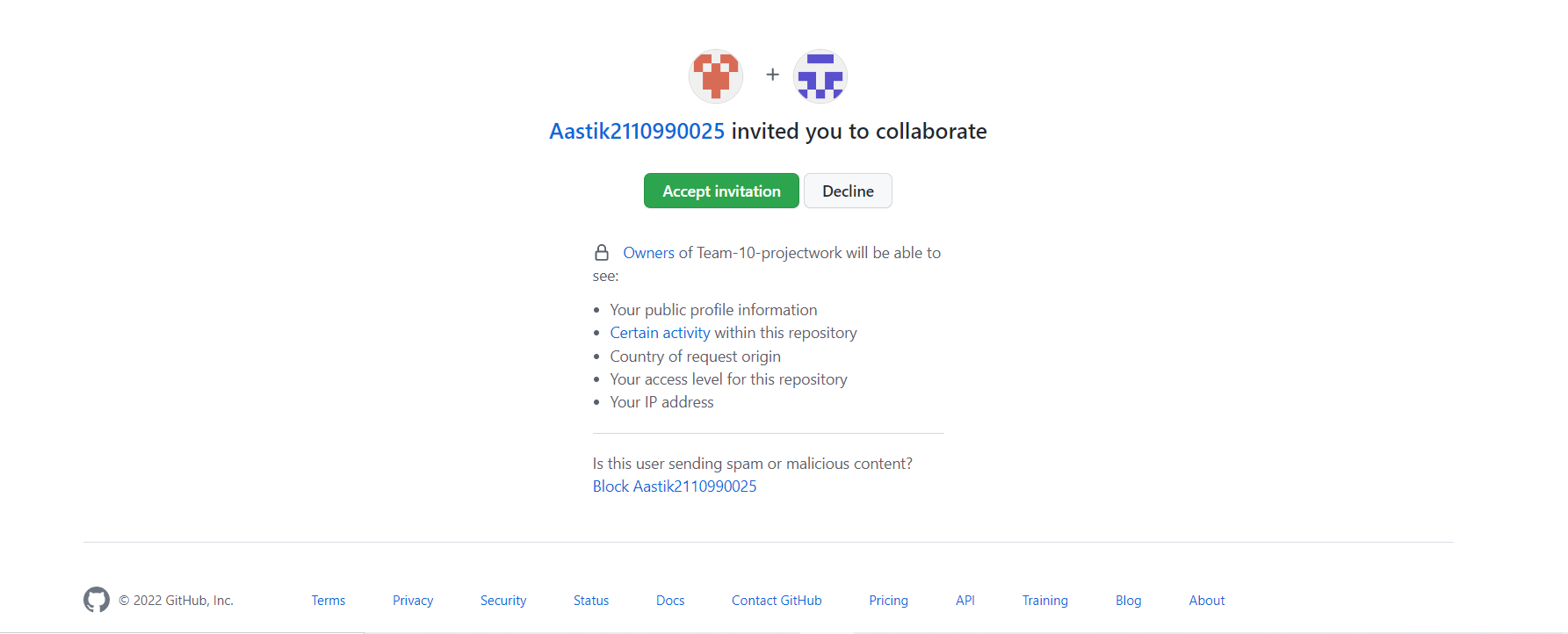
1. Enter the Repository name and add the description of the repository. Select if you want the repository to be public or private.



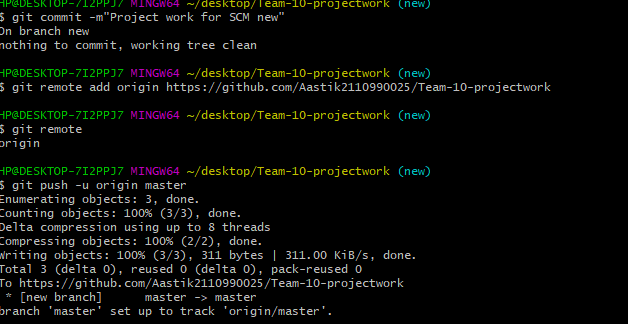
1. Now add collaborators in the repository.



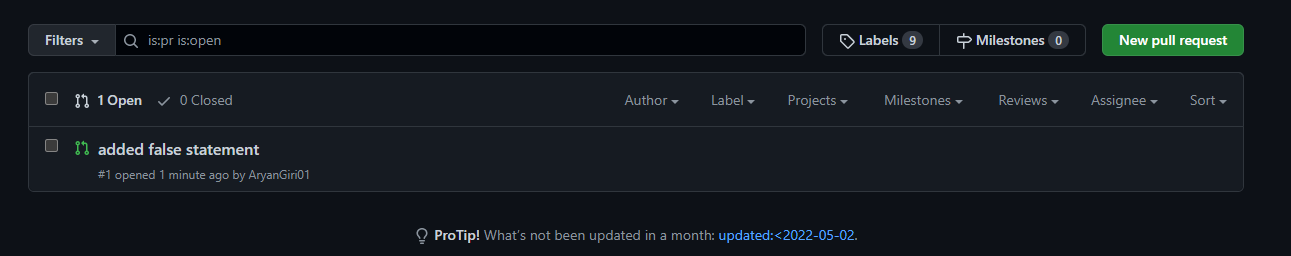
1. Now the collaborators will accept the invite that was sent to them.



1. To open a pull request we first have to make a new branch, by using git branch nameoption. After making new branch we add a file to the branch or make changes in the existing file. Add and commit the changes to the local repository.
2. Use git push origin branch nameoption to push the new branch to the main repository.

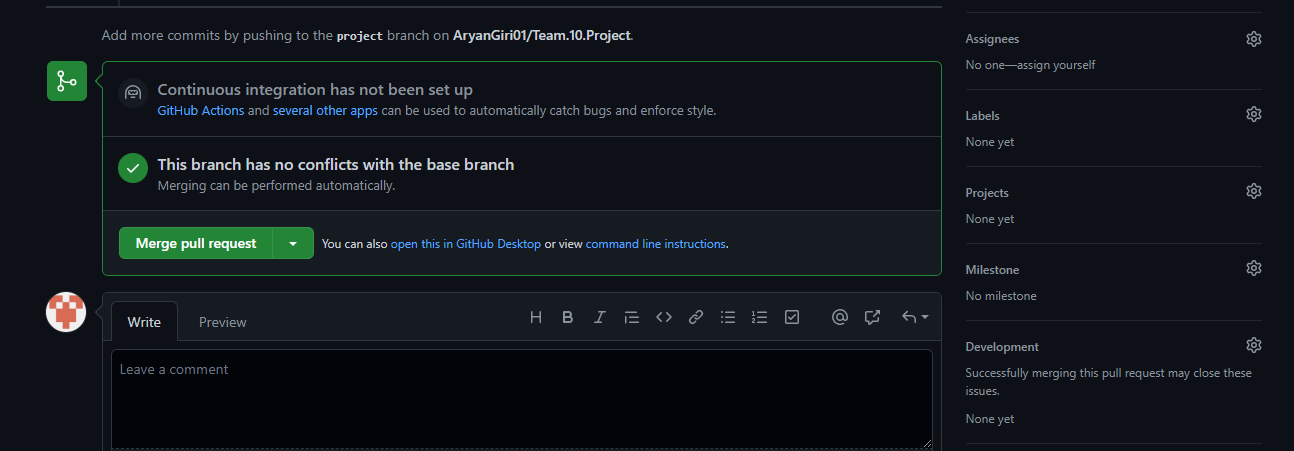


1. Now collaborator makes changes and creates a pull request.
2. Now the After opening a pull request all the team members will be sent the request if they want to merge or close the request. If the team member chooses not to merge your pull request, they will close you’re the pull request. To close the pull request simply click on close pull request and add comment/ reason why you closed the pull request.

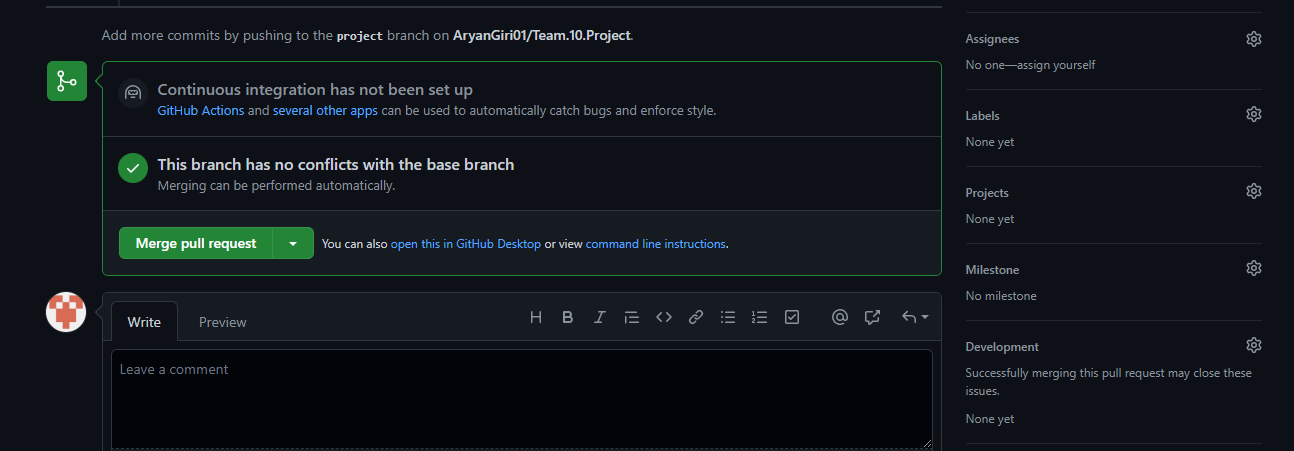


1. Now Do the required changes in the repository, add and commit these changes in the local repository in a new branch. Push the modified branch using git push

-u origin branch name. Open a pull request by following the procedure from the above experiment. The pull request will be created and will be visible to all the team members. Ask your team member to login to his/her GitHub account. They will notice a new notification in the pull request menu. Click on it. The pull request generated by you will be visible to them.



1. Click on the pull request. Two options will be available, either to close the pull request or merge the request with the main branch. By selecting the merge branch option, the main branch will get updated for all the team members. By selecting close the pull requests the pull request is not accepted and not merged with main branch.



1. If the file is merged it will show a message to you.



1. Network Graph need to be recorded.

