ARYAN AGARWAL

Subject Name: **Source Code Management**

Subject Code: **CS181**

Cluster: **BETA**

Department: **DCSE**

Submitted by:

Aaryan Agarwal

2110990017

Submitted to:

Mr. Monit kapoor

INDEX

Experiment No. 01

Aim: Add collaborators on GitHub Repository

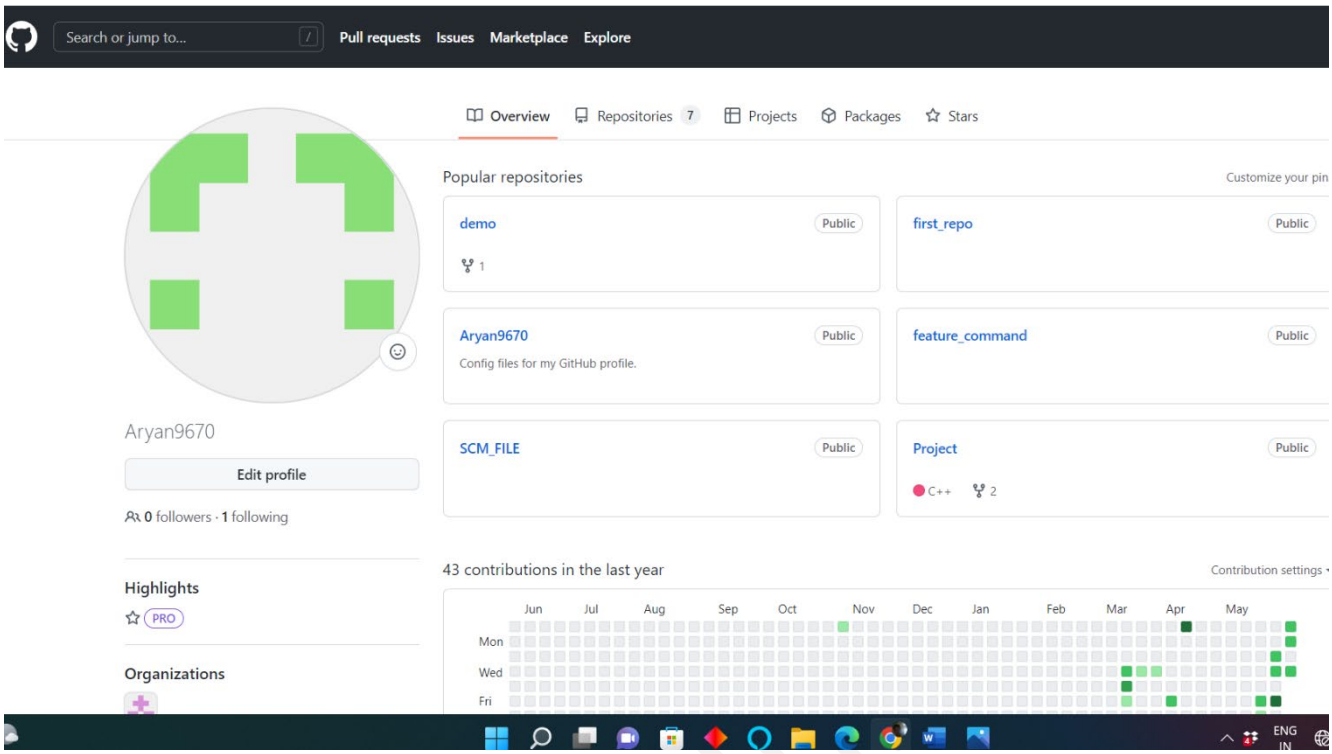
What is GitHub Collaboration?

GitHub collaboration is a space where you can invite another developer to your repository and work together at an organization level, splitting the task and 2nd person will have the rights to add or merge files to the main repository without further permission.

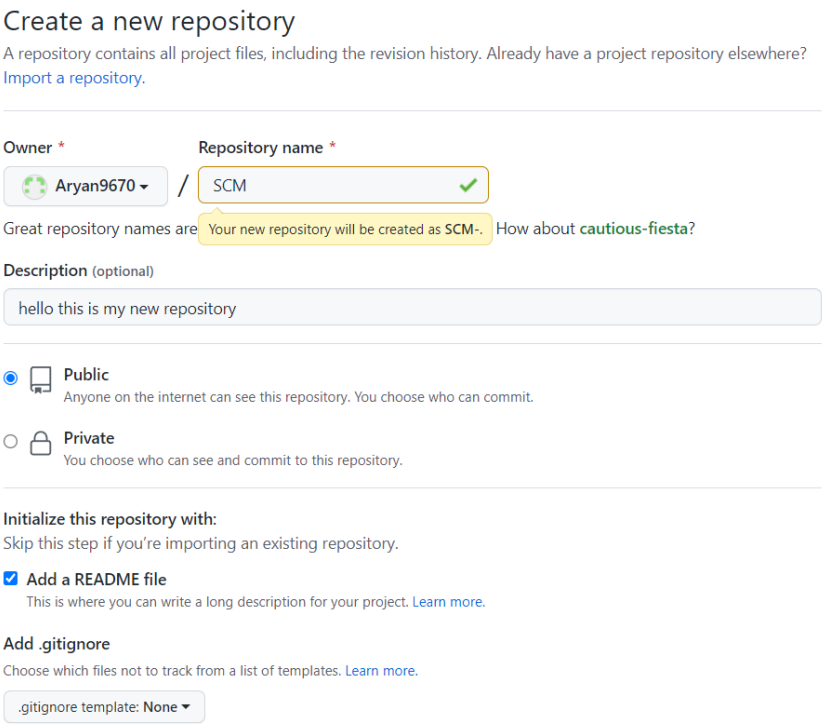
Let's invite a Collaborator.

In this post, we will create a new repository and invite a collaborator to our repository by sending an invitation. A detailed procedure on how to invite a collaborator to your GitHub repository is mentioned below.

Step 1: Went to Git hub and created new Repository, click on the + sign on top right side and drop down will appear click on New Repository.



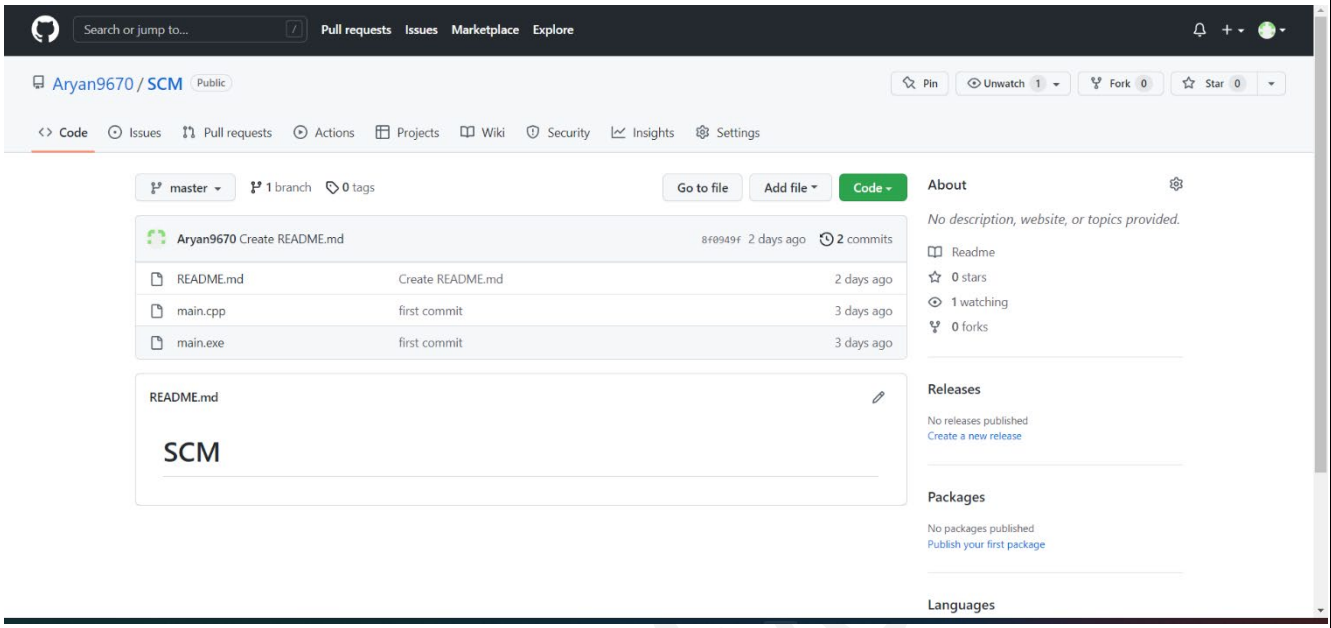
Step 2: Specify the Name of the Project, make It public or primate, check on the readme file. Then click on Create repository. You can also see by default Brach name is main If you want you can change it. **(optional)**



Step 3: Now you can see the file created on that name with a default readme file. You can add content to the readme file. Now the next step is to invite the collaborators to the repository. currently, you have your repository ready to collaborate.

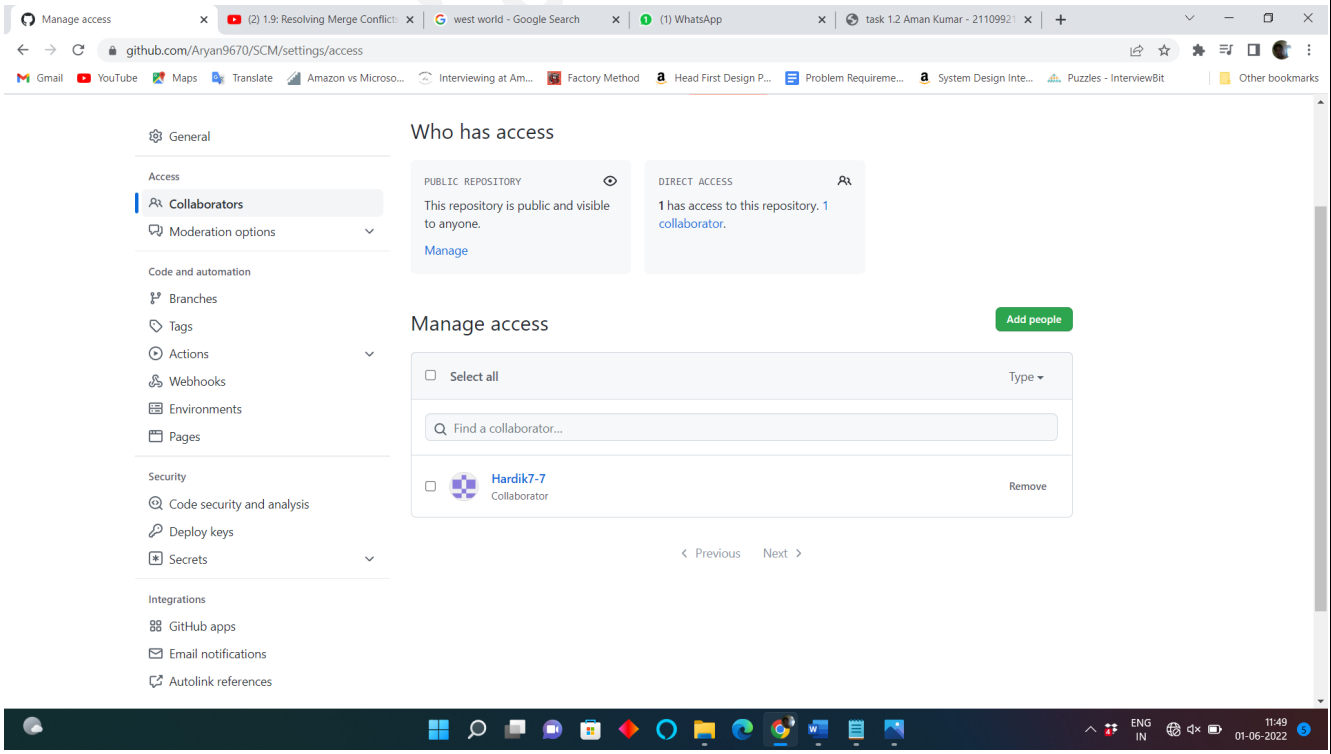
Link to GitHub Main Repo: https://github.com/Aryan9670/SCM

Step 4: Click on **settings,** you should be the one who created the repo to do this work, because only the author can send the invite to others.

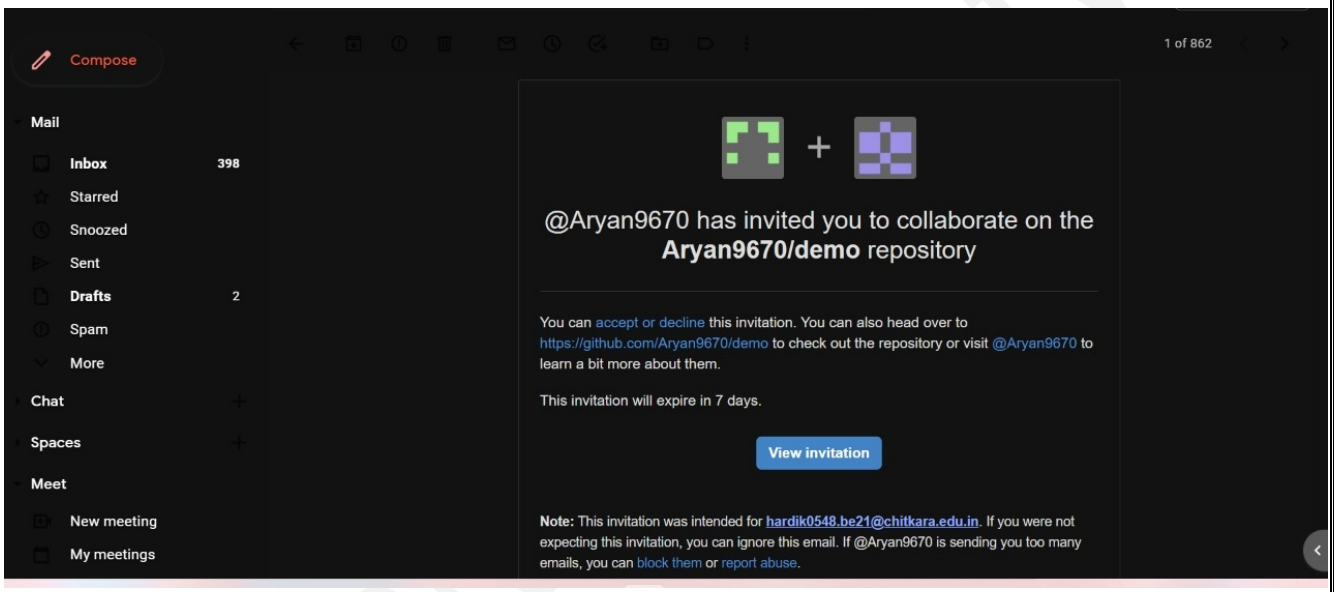


Step 5: Click on the invited collaborator and enter the email ID of people you want to add to this repo. Email will go to them to accept the request.

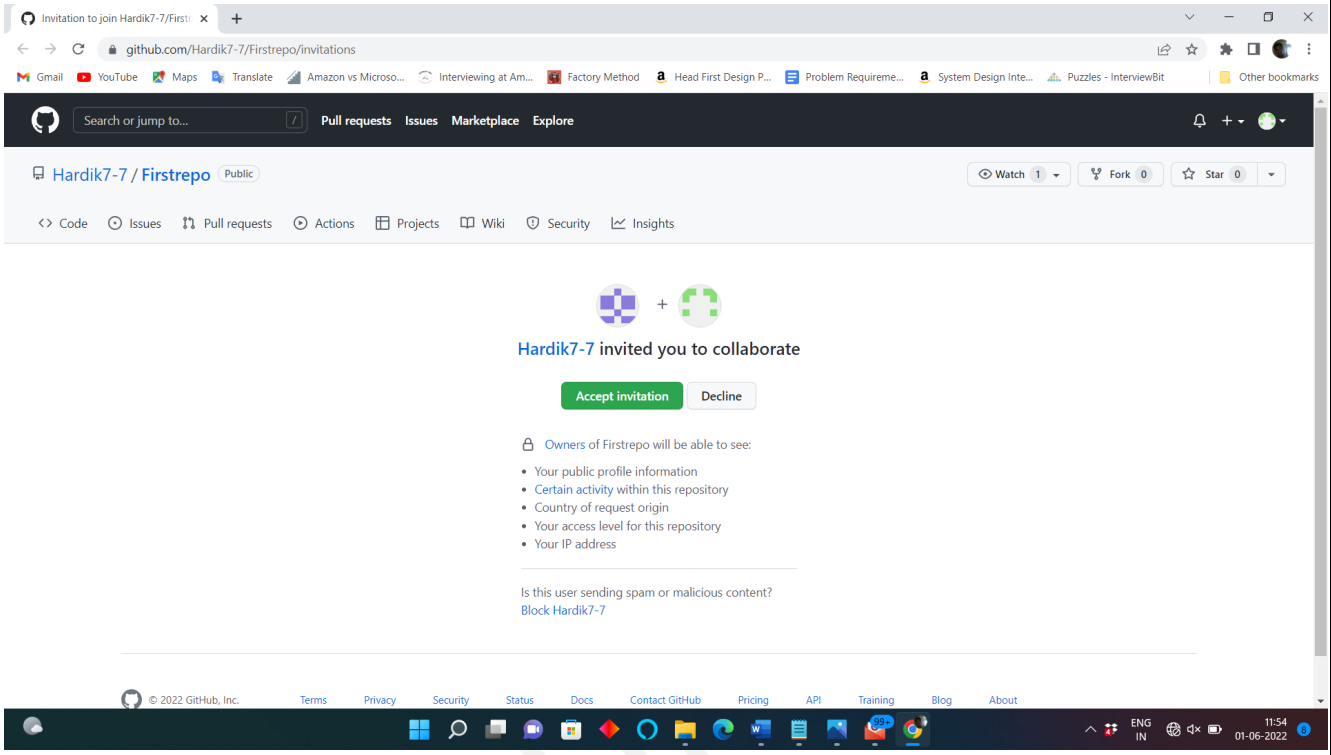
Once it accepted, they can push the changes to main branch.



Step 6: The person will be getting an email invitation like this for GitHub collaboration, click on the Email View and accept invitation.

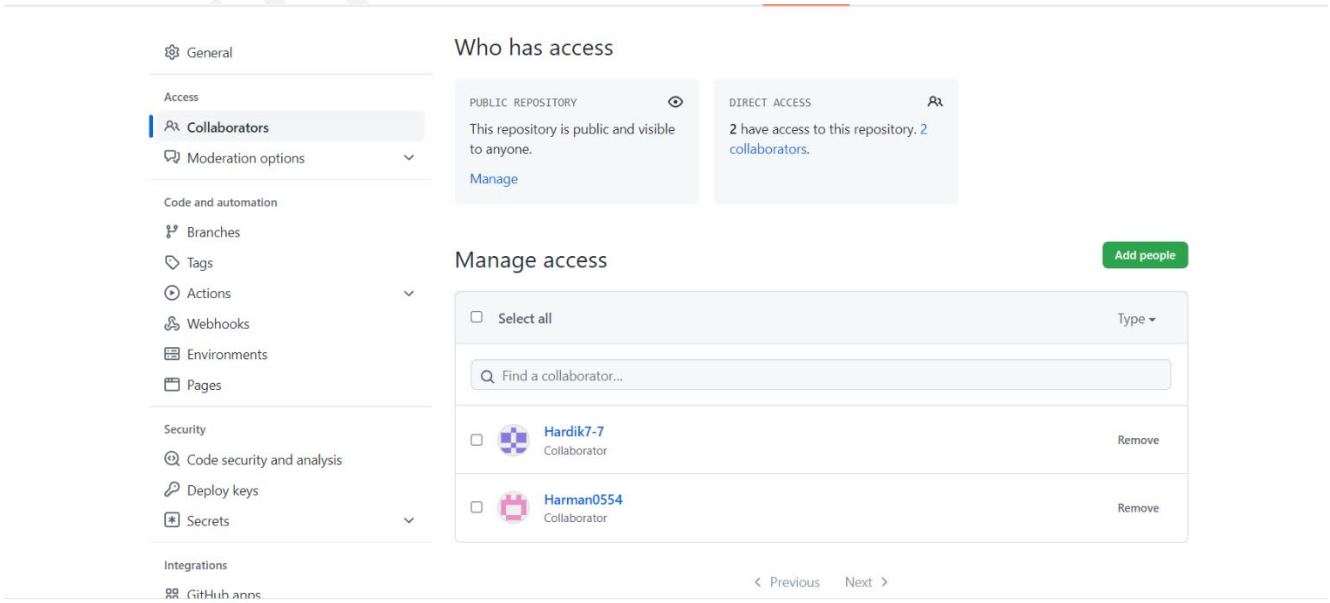


Step 7: You will be redirected to GitHub Windows there click on Accept Invitation.



How to see Existing GitHub Repository collaborator

Go to your respective Repository and click on Manage access under Security tab, sometimes it will ask you to authenticate your account by entering the password. You can see list of people under the Manage access.



Fatal: unable to access | Error: 403

What will happen if the person not added you as a collaborator and you tried to clone the file and push the repository to the main file? It will lead to the error permission denied. So it's important to give enough permission.

```
fatal: unable to access 'URL': The requested URL returned error: 403
```

FINAL VERDICT:

In conclusion, I hope you enjoyed reading this article on "**How to add Collaborators into GitHub Repository?**". Again, inviting a collaborator to GitHub is good to approach, but making the changes and pushing to the main repository is not a good approach, instead, we create individual branches and push the work to branch and later merge to the main repository.

How to delete a collaborator from GitHub Repository?

Go to the Repository, Click on Manage access under settings tab, you can see a list of collaborators under Mange access, on right side of each listing there is a delete icon.

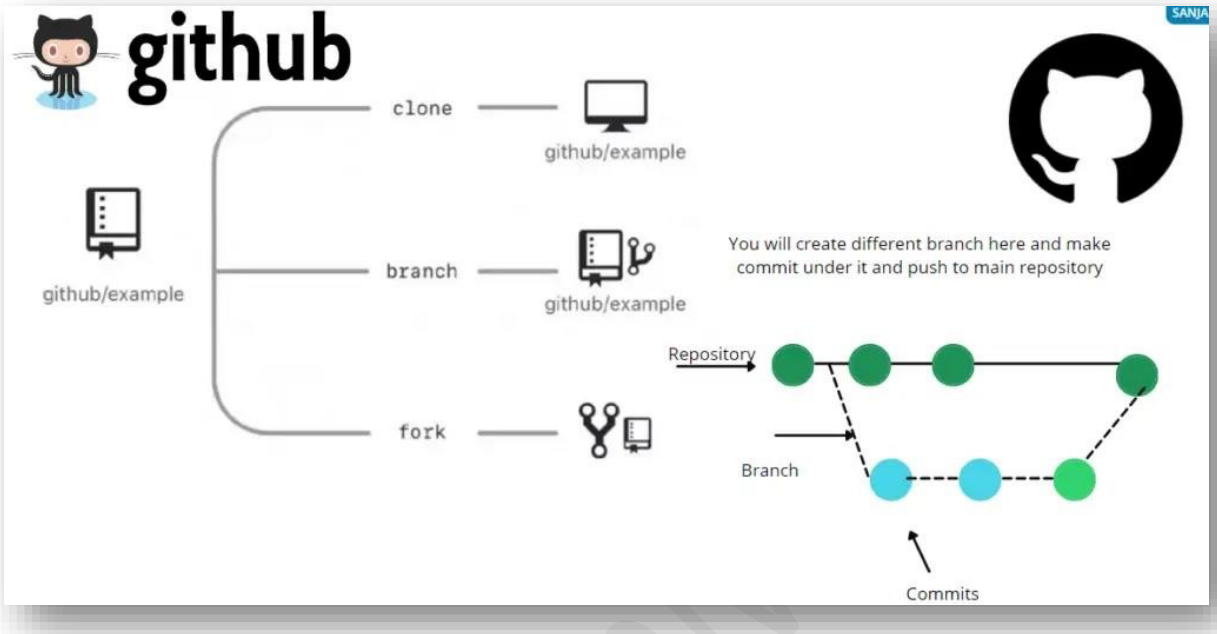
Experiment No. 02

Aim: Fork and Commit

What is Forking in GitHub?

Forking an Existing GitHub repository helps to create a **copy of the main file in production**, hence provide the user to experiment with the copied repository on their GitHub branch. So, any changes made to the forked file won't affect the main file in production.

The above image shows how the fork is working once you added the new feature to the copied branch and this committee will be happening in the branch, once you tested this out, you have an option **to pull request and merge the change with the main branch**. Then the owner of the particular repository will look at the changes and do the code review and accept the changes.

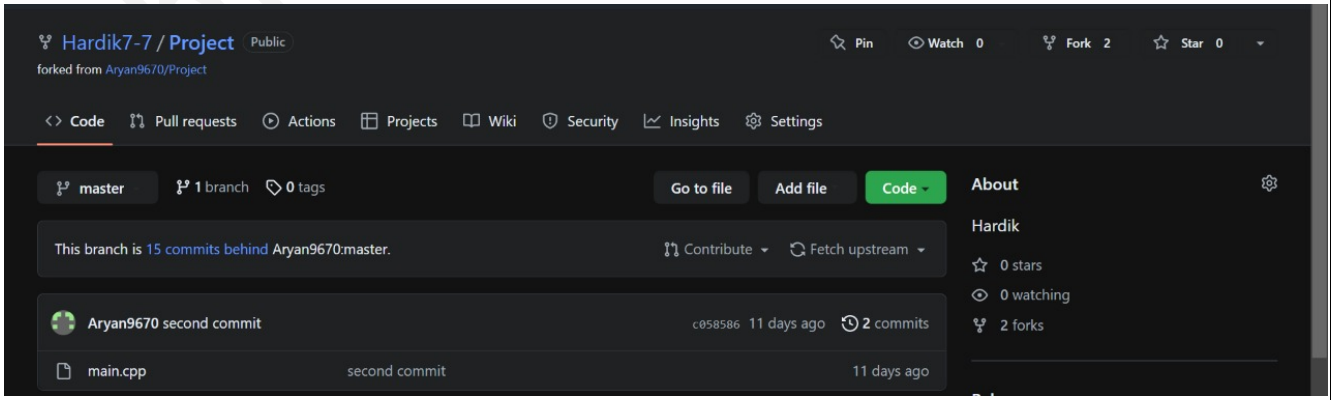


Forking a repository from GitHub

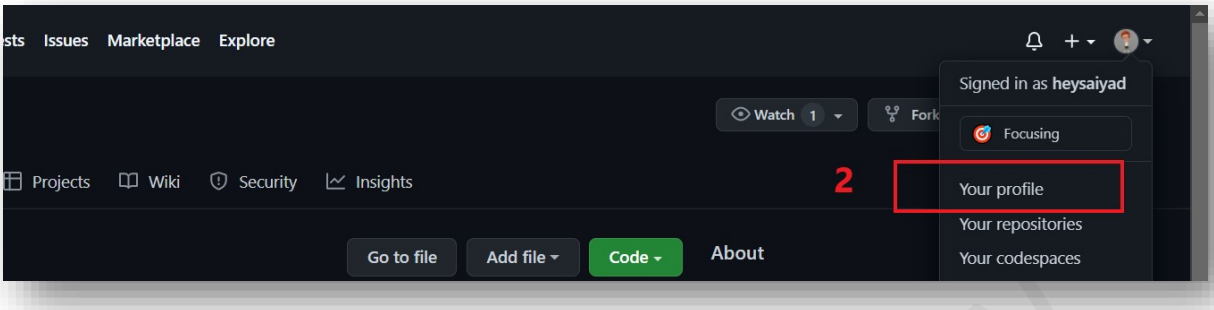
You might a project to propose changes to the upstream, or original, repository. In this case, it's good practice to regularly sync your fork with the upstream repository. To do this, you'll need to use Git on the command line. You can practice setting the upstream using the same https://github.com/Aryan9670/SCM repository you just forked.

Step 1: On GitHub.com, navigate to the https://github.com/Aryan9670/SCM repository.

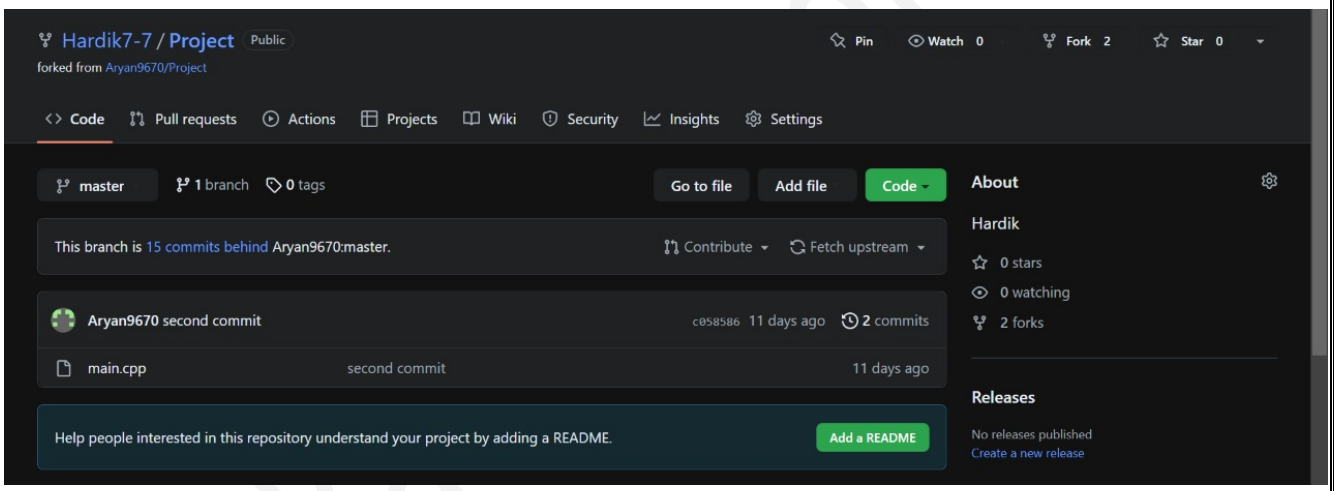
Step 2: Go In the top-right corner of the page, click fork.



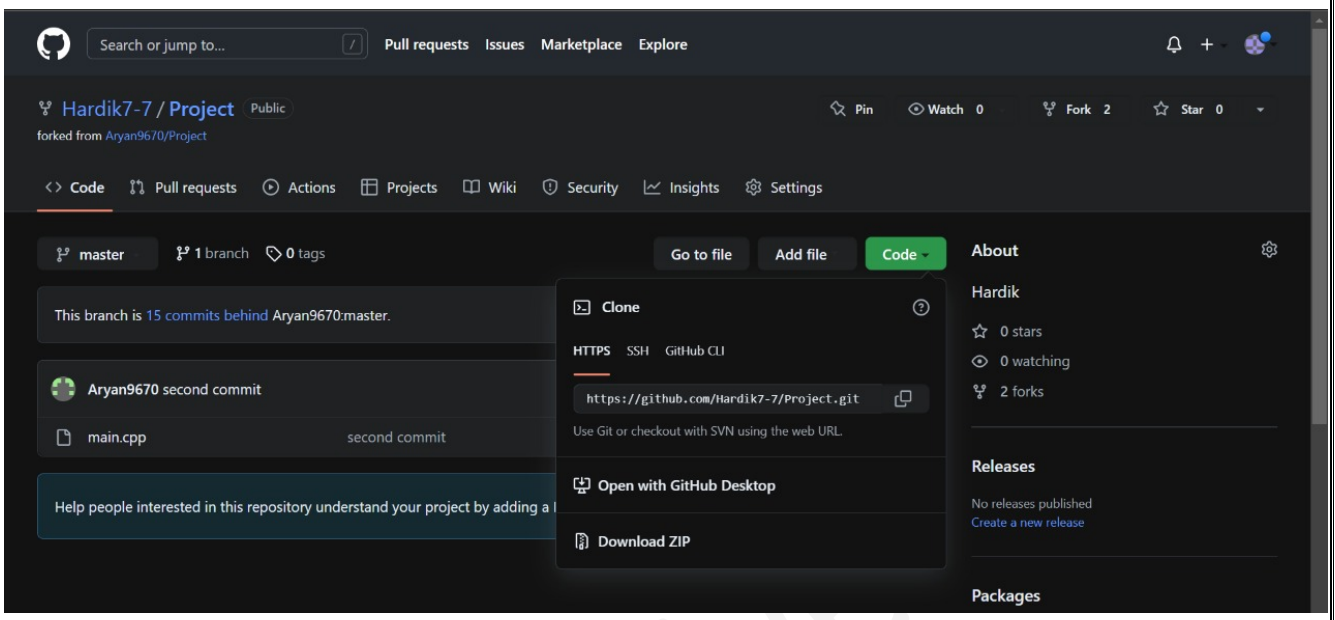
Step 3: Now go to your profile and see the latest forked repository in your repository tab.



Step 4: The below screen shows the GitHub Profile that I have forked now. Then the next process is to clone the project for this click on the code button as highlighted below.



To clone the repository using HTTPS, under "Clone with HTTPS", click. To clone the repository using an SSH key, including a certificate issued by your organization's SSH certificate authority, click **Use SSH**, then click. To clone a repository using GitHub CLI, click **Use GitHub CLI**, then click.



1. Open Git Bash.
2. Change the current working directory to the location where you want the cloned directory.
3. Type git clone, and then paste the URL you copied earlier. It will look like this, with your GitHub username instead of YOUR-USERNAME:

```
$git clone https://github.com/Hardik7-7/2110990017_SCM.git
```

4 Press **Enter**. Your local clone will be created.

```
hardi@OGS MINGW64 ~/OneDrive/Desktop
$ git clone https://github.com/Hardik7-7/2110990017_SCM.git
Cloning into '2110990017_SCM'...
remote: Enumerating objects: 10, done.
remote: Counting objects: 100% (10/10), done.
remote: Compressing objects: 100% (8/8), done.
remote: Total 10 (delta 1), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (10/10), 1.01 MiB | 360.00 KiB/s, done.
Resolving deltas: 100% (1/1), done.

hardi@OGS MINGW64 ~/OneDrive/Desktop
$ cd 2110990017_SCM/
```

Step 5: That's it you have made changes to the file and it's time to stage and commit the file and push. **Git add** will add the file to commit.

```
git add README.md
```

```
$ nano main.cpp

hardi@OGS MINGW64 ~/OneDrive/Desktop/2110990017_SCM (hardik)
$ git status
On branch hardik
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   main.cpp

no changes added to commit (use "git add" and/or "git commit -a")

hardi@OGS MINGW64 ~/OneDrive/Desktop/2110990017_SCM (hardik)
$ git add .
```

Step 6: Commit the changes by below commit code. Make sure you add the comment this will make the other user understand the other user to find what's your change is all about.

```
git commit -m "Add README.md"
```

```
hardi@OGS MINGW64 ~/OneDrive/Desktop/2110990017_SCM (hardik)
$ git add .

hardi@OGS MINGW64 ~/OneDrive/Desktop/2110990017_SCM (hardik)
$ git commit -m "Modified main.cpp file | From hardik branch"
[hardik e606c23] Modified main.cpp file | From hardik branch
 1 file changed, 23 insertions(+), 19 deletions(-)
```

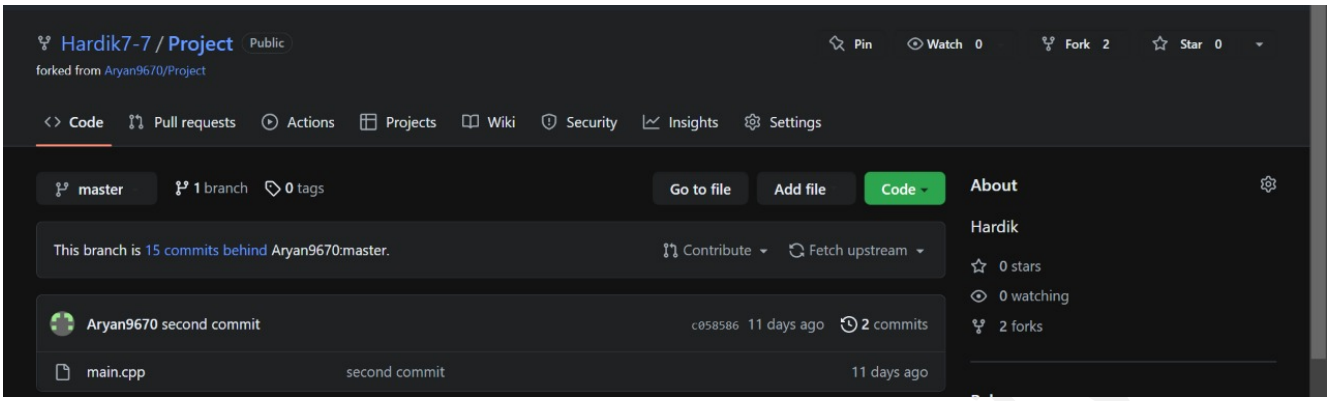
Step 7: Pushing your repository to GitHub, here I created this master branch so I'm pushing the file to that branch.

```
git push origin <name of your branch>
```

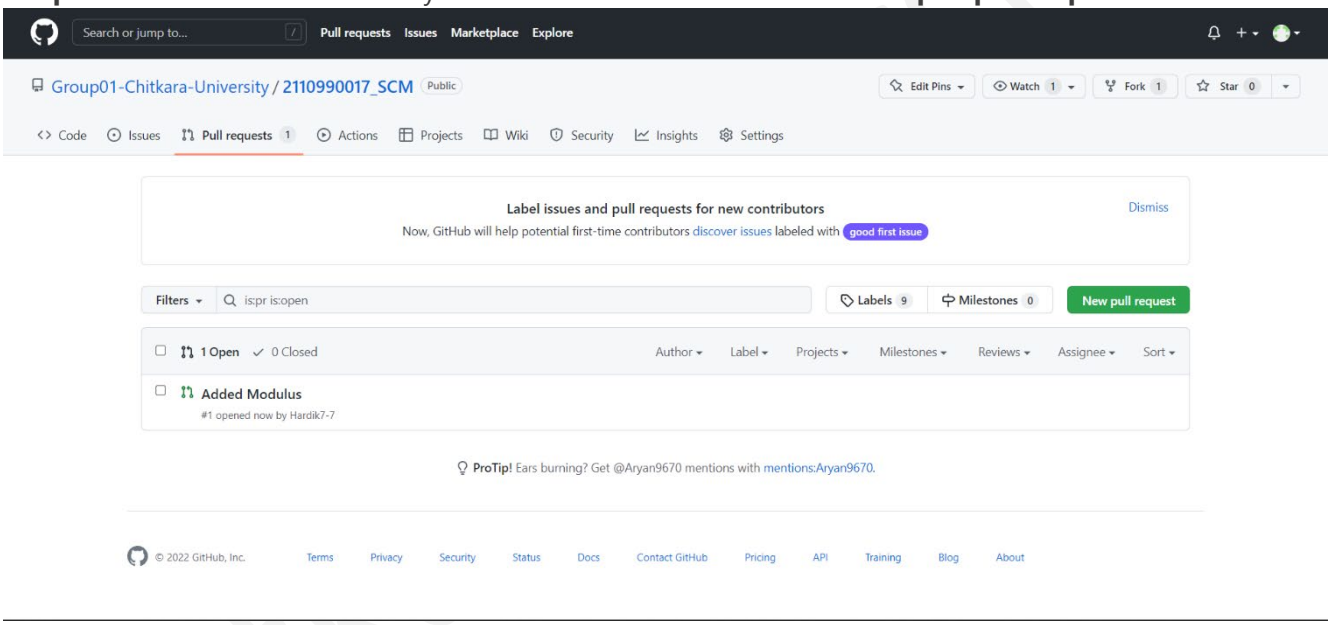
```
hardi@OGS MINGW64 ~/OneDrive/Desktop/2110990017_SCM (hardik)
$ git push origin hardik
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 507 bytes | 507.00 KiB/s, done.
Total 3 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
remote:
remote: Create a pull request for 'hardik' on GitHub by visiting:
remote:      https://github.com/Hardik7-7/2110990017_SCM/pull/new/hardik
remote:
To https://github.com/Hardik7-7/2110990017_SCM.git
 * [new branch]      hardik -> hardik
```

Step 8: Now jump to your repository, under your branch that you pushed you can see an option to merge the pull request to the main branch.

Note: A pull request allows your changes to be merged with the original project.

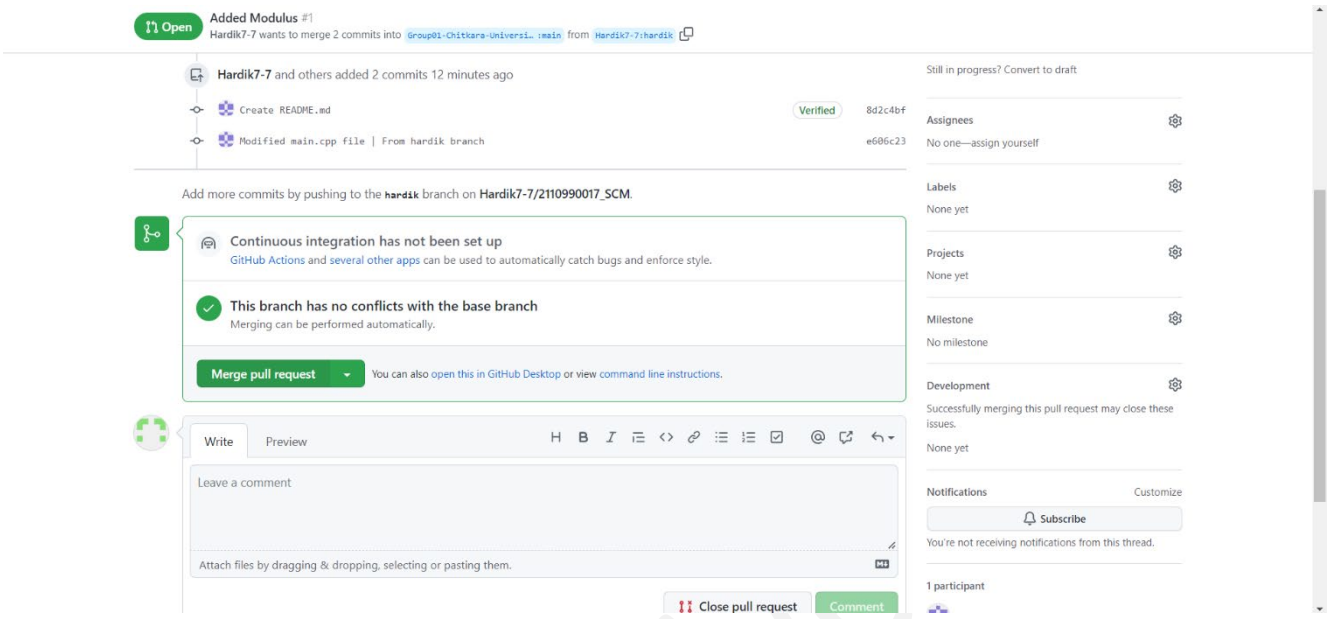


Step 9: Now on the main branch you will be able to see **contribute** the **open pull request.**

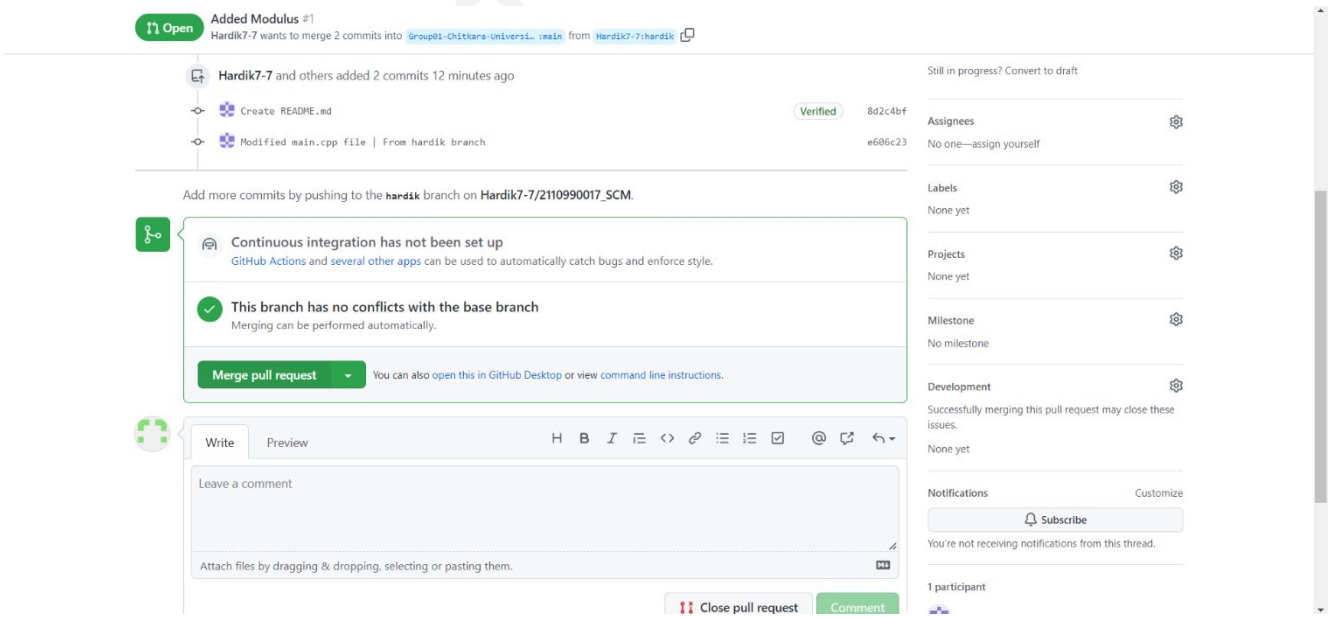


Step 10: Make sure to add a good comment in the merge and explain what's your commit is all about.

Step 11: In few Organisations, there will be Github actions scheduled for auto-check and any one of the code reviewers will review the file and approve the changes or suggest any issue with your pull request.



Step 12: Here you can see one of the community members has reviewed my code and left a comment.



Experiment No. 03

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

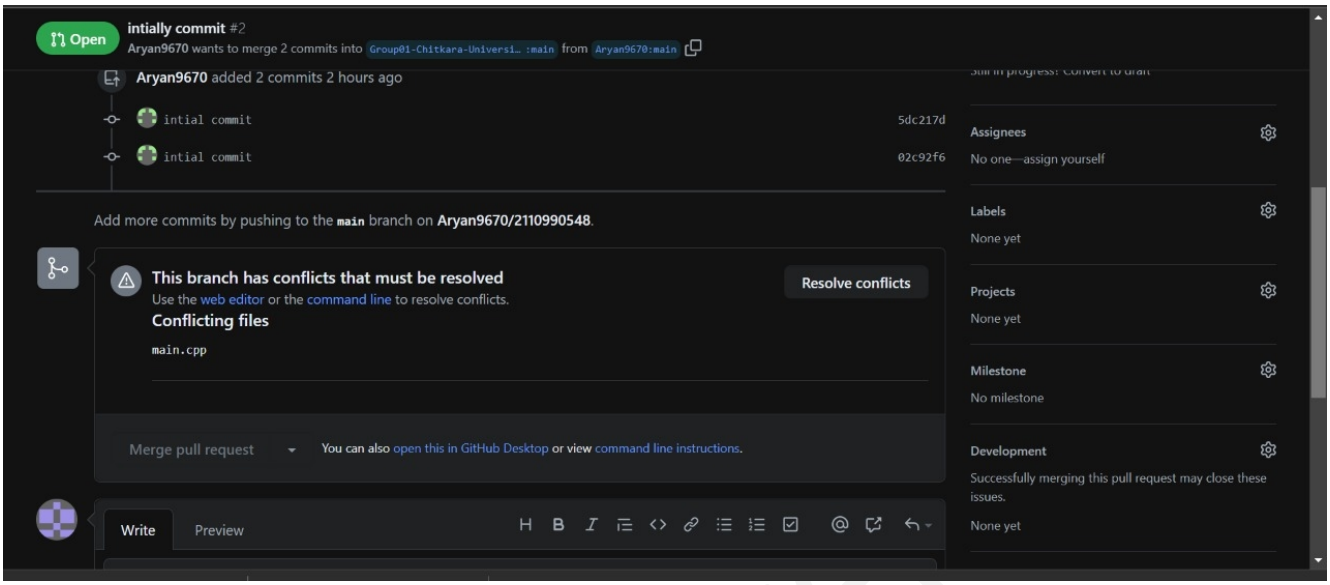
Understanding GitHub Conflicts

Let's imagine a case where two developers working on one repository by creating 2 different branch and same code line and both of them pulled the file to main repository, Now the GitHub is good understanding the code and merge the conflicts automatically but it fails understanding if developer has put any comments. Now Let's look into How to resolve merge conflicts in GitHub.

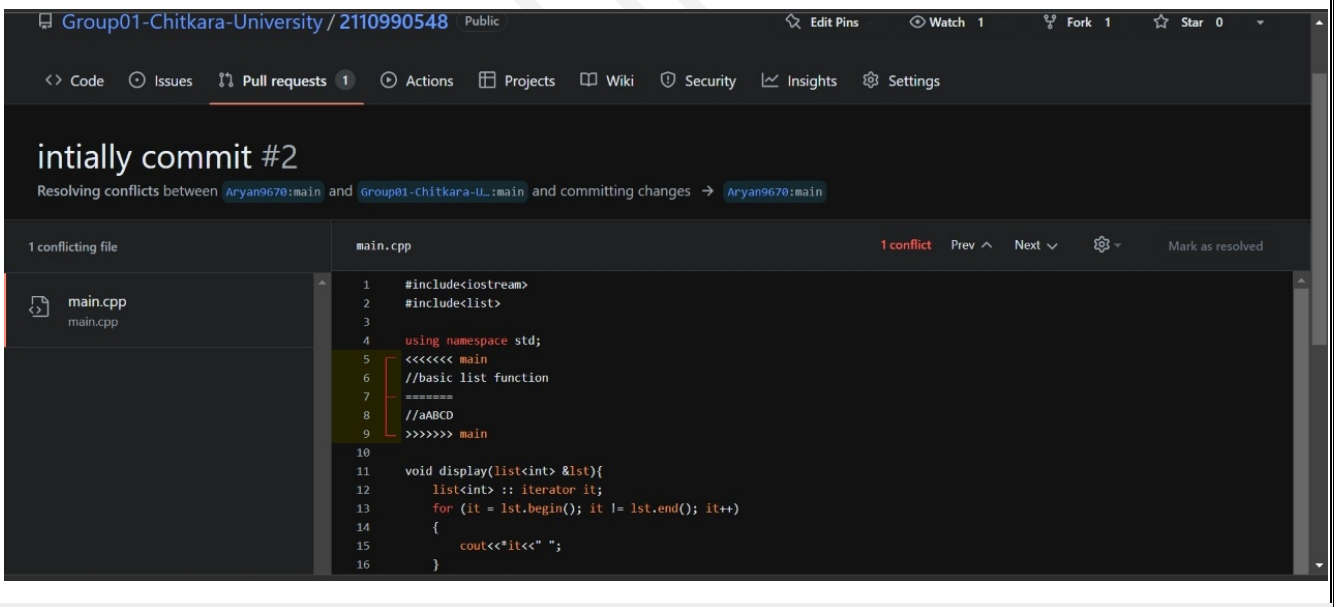
Let's resolve GitHub Conflicts

Step 1: In case if you don't know how to do pull request in GitHub, Once you press compare and pull request, GitHub will look into the possibilities of merging that itself if not it will show developer to solve the resolve conflicts Manually.

Step 2: Click on Resolve Conflict Button.



Step 3: Let's look into how to resolve these GitHub conflicts. As highlighted in 1 there are 16 conflicts here, and these conflicts are highlighted as below. Here GitHub found there are codes to merge in the same lines, <<<<< Sanjay represents your branch. Now if you want to keep the changes as it is just deleting the below lines from your entire code base. Click and backspace will do the work.



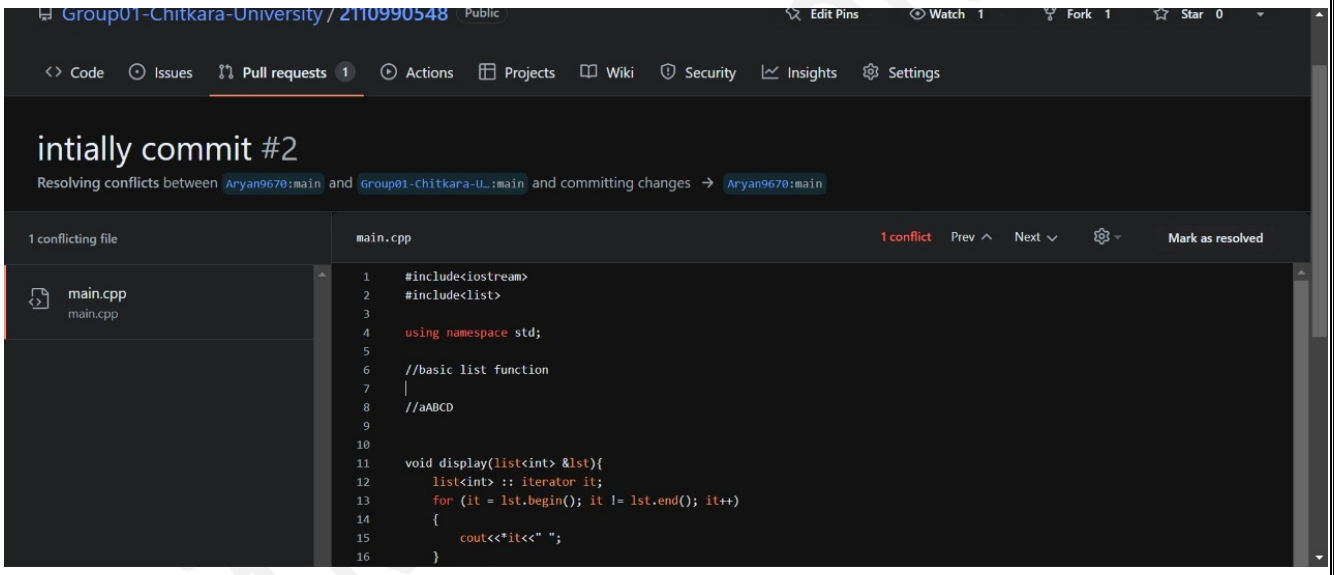

```
<<<<<<<< main



========

>>>>>>> main
```

Somebody has got merge conflicts in the file - those `<<<` and `===` and `>>>` lines are how git shows its merge conflicts. To make it valid, you need to get clean up those, and in each chunk, pick one of the bits inbetween them:

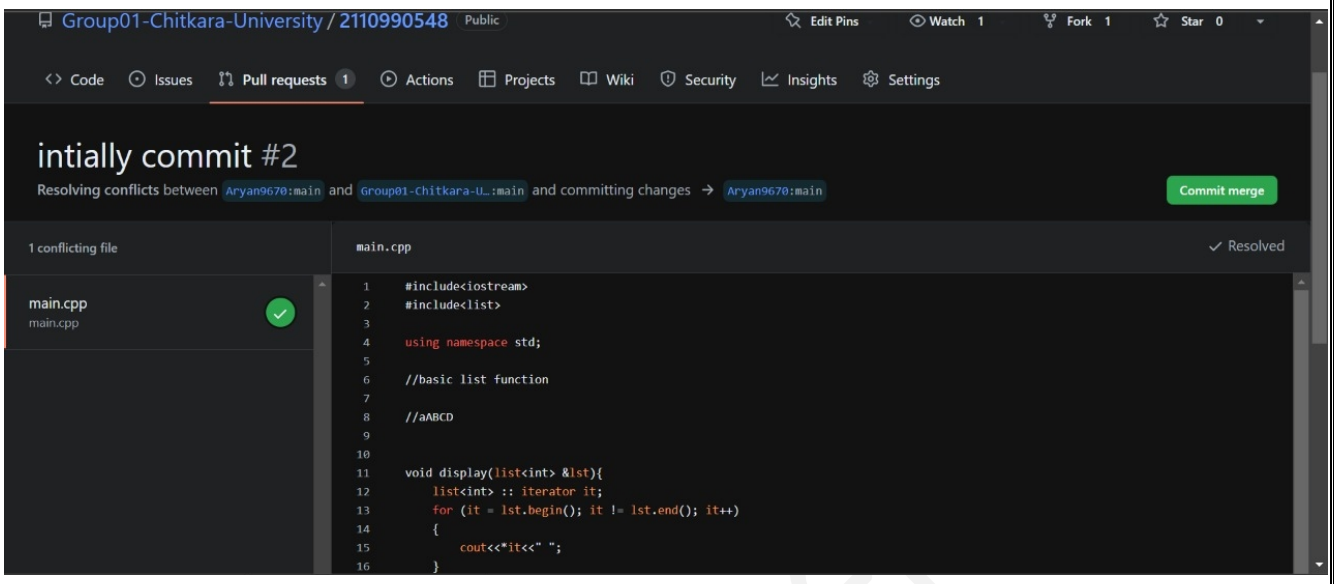
```
<<<<<< HEAD
   Either keep this line...
=======
   ... or this one. But not both!
>>>>>>> a1f5acbaa00bc1b1e5fc3532ba19a013a271542c
```

referenced image from GitHub gist

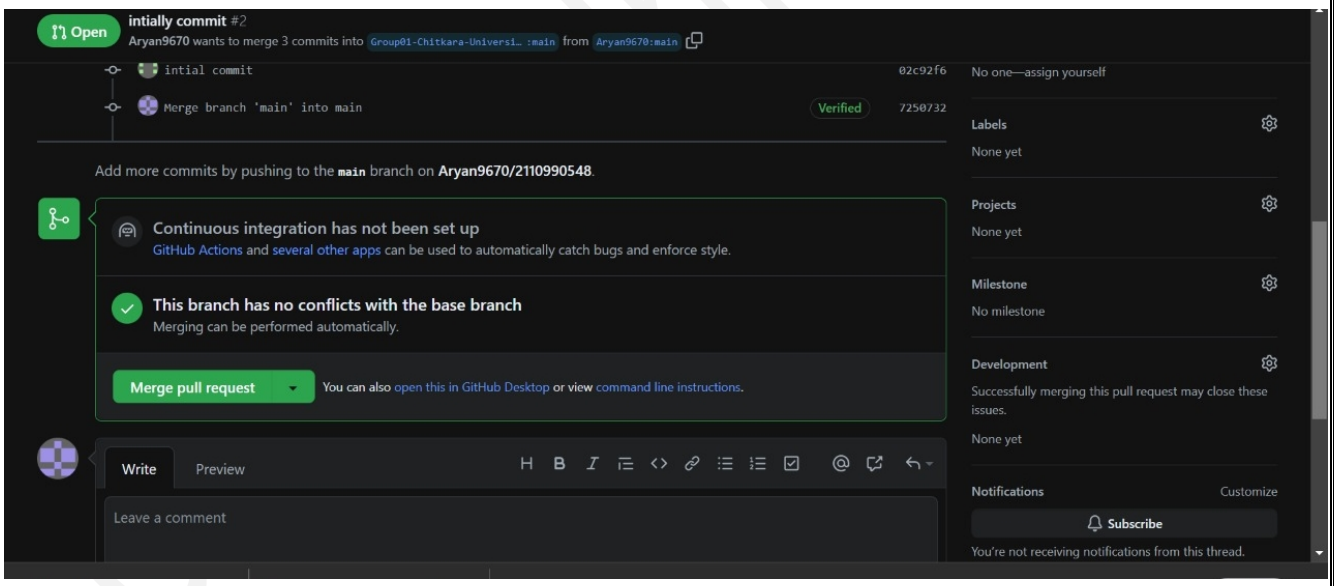
Step 4: Once all the conflicts has been removed, you can see the button Mark as resolved became active.



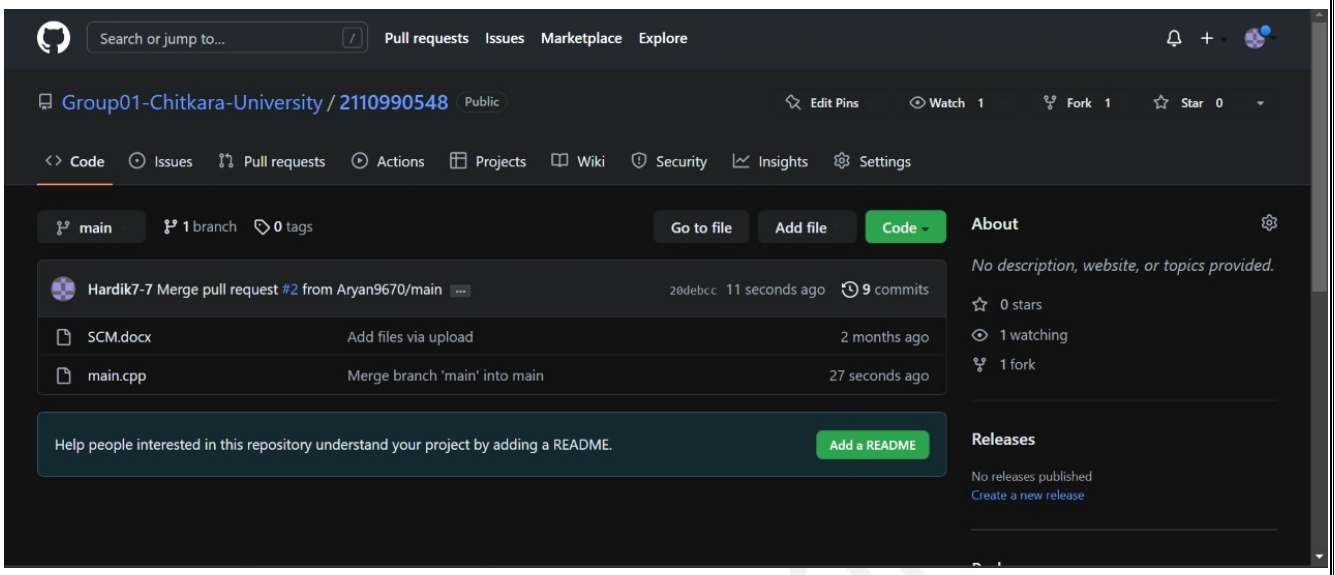
Step 5: Commit Merge to do this merging with the main branch



Step 6: In the next page you can see the merge pull request option, you also have an option to save this draft, here we will proceed with merging the pull request.



Now you can see the branch Sanjay has been merged to the main branch. You have successfully resolved git conflicts and merged changes in branch to main branch.



Experiment No. 04

Aim: Reset and revert

How to reset a Git commit

Let's start with the Git command reset. Practically, you can think of it as a "rollback"—it points your local environment back to a previous commit. By "local environment," we mean your local repository, staging area, and working directory.

Take a look at Figure 1. Here we have a representation of a series of commits in Git. A branch in Git is simply a named, movable pointer to a specific commit. In this case, our branch master is a pointer to the latest commit in the chain.

```
$ git log --oneline b764644
File with three lines 7c709f0
File with two lines
9ef9173 File with one line
```

What happens if we want to roll back to a previous commit. Simple—we can just move the branch pointer. Git supplies the reset command to do this for us. For example, if we want to

reset master to point to the commit two back from the current commit, we could use either of the following methods:

$ git reset 9ef9173 (using an absolute commit SHA1 value 9ef9173) or

$ git reset current~2 (using a relative value -2 before the "current" tag)

Figure 2 shows the results of this operation. After this, if we execute a git log command on the current branch (master), we'll see just the one commit.

```
$ git log --oneline
9ef9173 File with one line
```

The git reset command also includes options to update the other parts of your local environment with the contents of the commit where you end up. These options include: hard to reset the commit being pointed to in the repository, populate the working directory with the contents of the commit, and reset the staging area; soft to only reset the pointer in the repository; and mixed (the default) to reset the pointer and the staging area.

Using these options can be useful in targeted circumstances such as git reset -hard <commit sha1 | reference>. This overwrites any local changes you haven't committed. In effect, it resets (clears out) the staging area and overwrites content in the working directory with the content from the commit you reset to. Before you use the hard option, be sure that's what you really want to do, since the command overwrites any uncommitted changes.

How to revert a Git commit
The net effect of the git revert command is similar to reset, but its approach is different. Where the reset command moves the branch pointer back in the chain (typically) to "undo" changes, the revert command adds a new commit at the end of the chain to "cancel" changes. The effect is most easily seen by looking at Figure 1 again. If we add a line to a file in each commit in the chain, one way to get back to the version with only two lines is to reset to that commit, i.e., git reset HEAD~1.

Another way to end up with the two-line version is to add a new commit that has the third line removed—effectively cancelling out that change. This can be done with a git revert command, such as:

```
$ git revert HEAD
```

Because this adds a new commit, Git will prompt for the commit message:

```
Revert "File with three lines"

This reverts commit
b764644bad524b804577684bf74e7bca3117f554.

# Please enter the commit message for your changes.
Lines starting
# with '#' will be ignored, and an empty message aborts
the commit.
# On branch master
# Changes to be committed:
#       modified:   file1.txt #
```

Figure 3 (below) shows the result after the revert operation is completed.

If we do a git log now, we'll see a new commit that reflects the contents before the previous commit.

```
$ git log --oneline
11b7712 Revert "File with three lines"
b764644 File with three lines 7c709f0
File with two lines 9ef9173 File with
one line
```

Here are the current contents of the file in the working directory:

```
$ cat <filename>
Line 1
Line 2
```

TASK 2

S. No.	Title
1	Introduction
2	Creating a repo
3	Open and Close Pull request
4	Fork ,clone and creating pull request

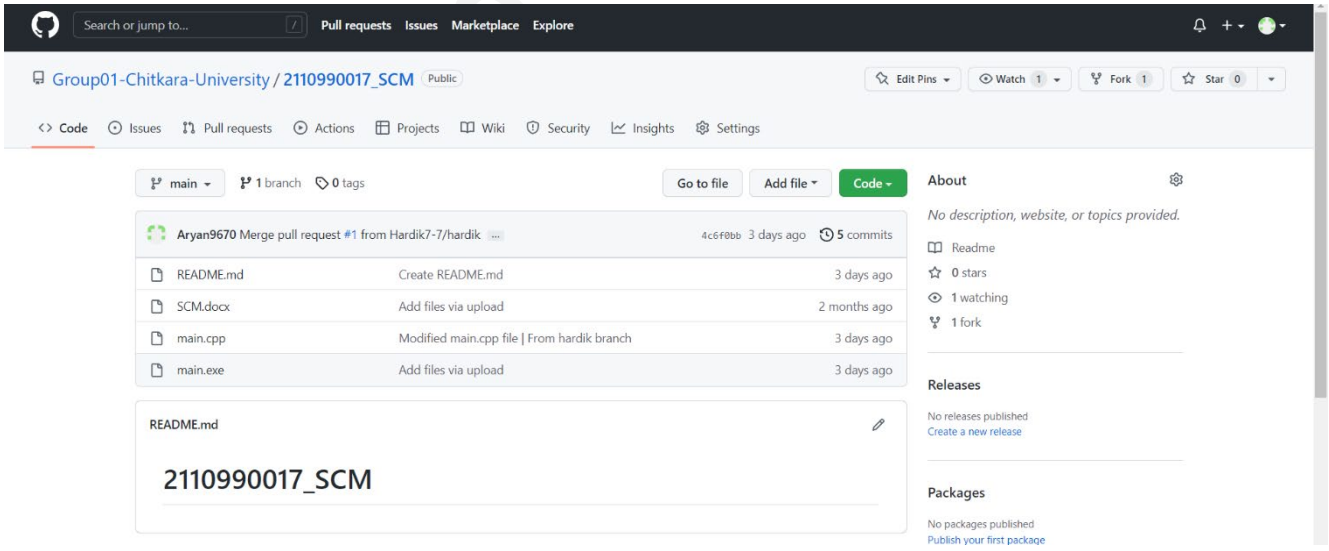
INTRODUCTION:

This task is performed in the group of four. Each one of us made it possible to work on this project as if we are doing an open source contribution.

Each one of us create his/her repo and rest of the three contributers in the repo , firstly forked that repo and then clone it in our local machine and then make a new branch and made some changes in the existing file in master branch in the repo and then push it from your local system.

And finally make pull request to the owner of the repo in whose repo we want to make the changes.

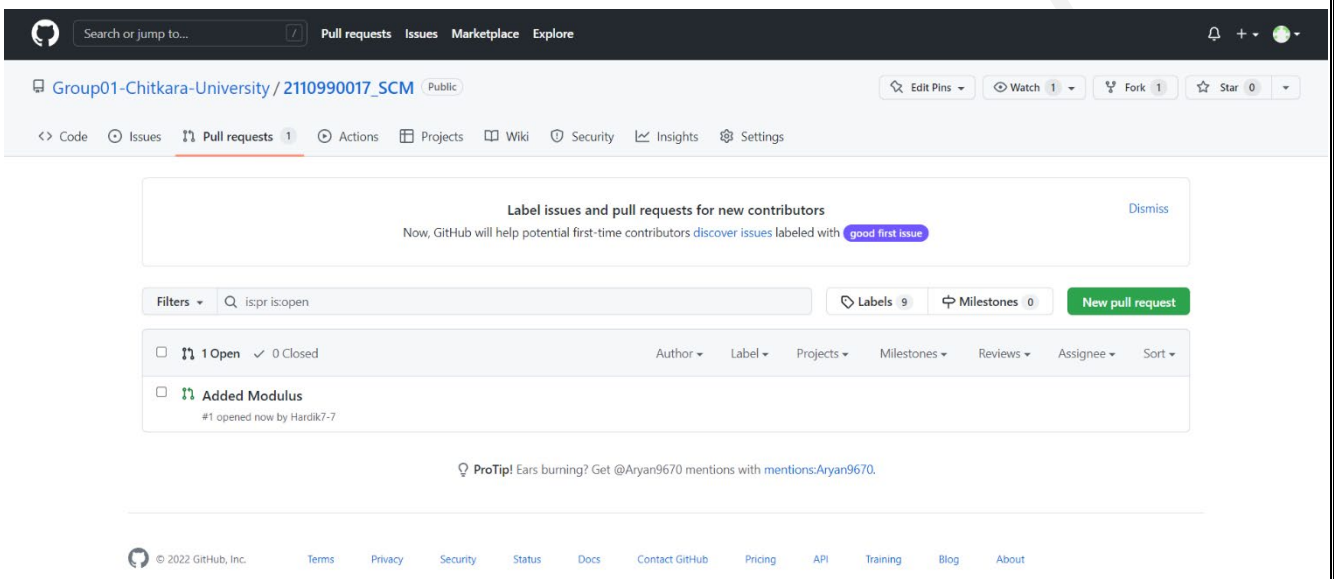
CREATING A REPO :

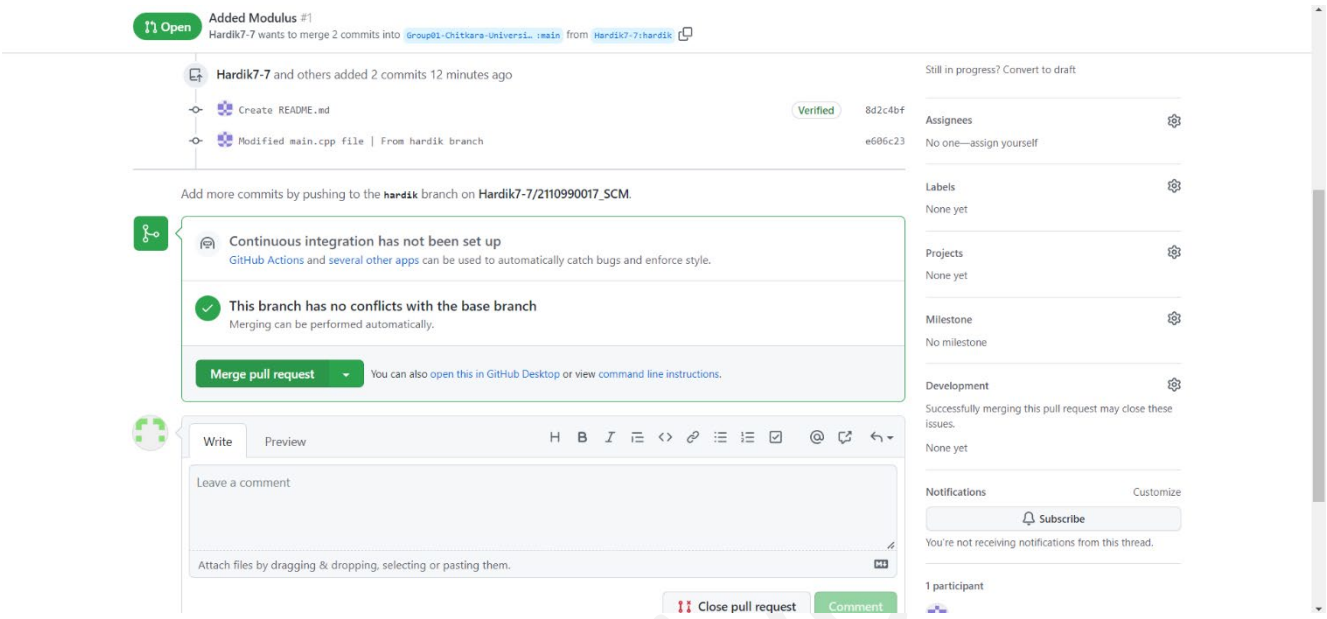


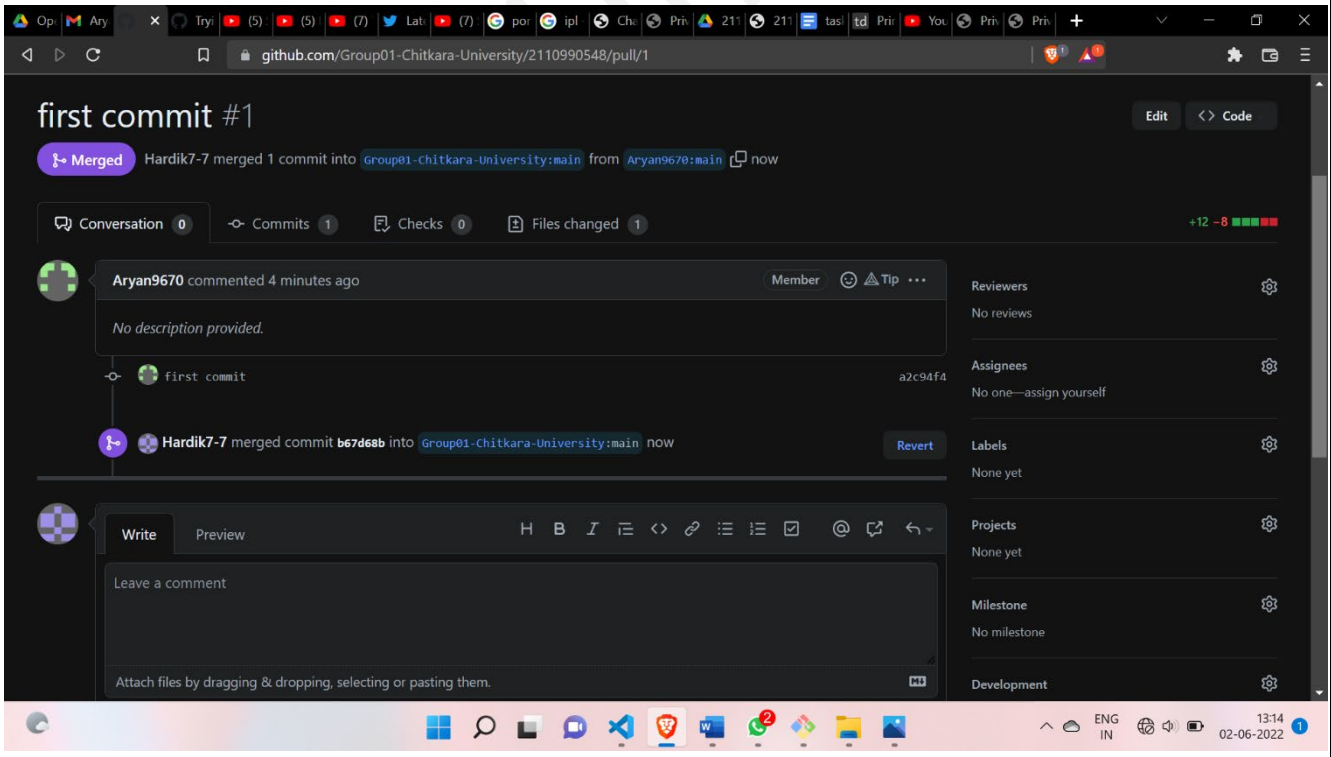
OPEN AND CLOSE PULL REQUEST:

Overview of the pull request sent by the contributers.

And if in case there is a confict in merging , solved it by mergetool in the git bash.

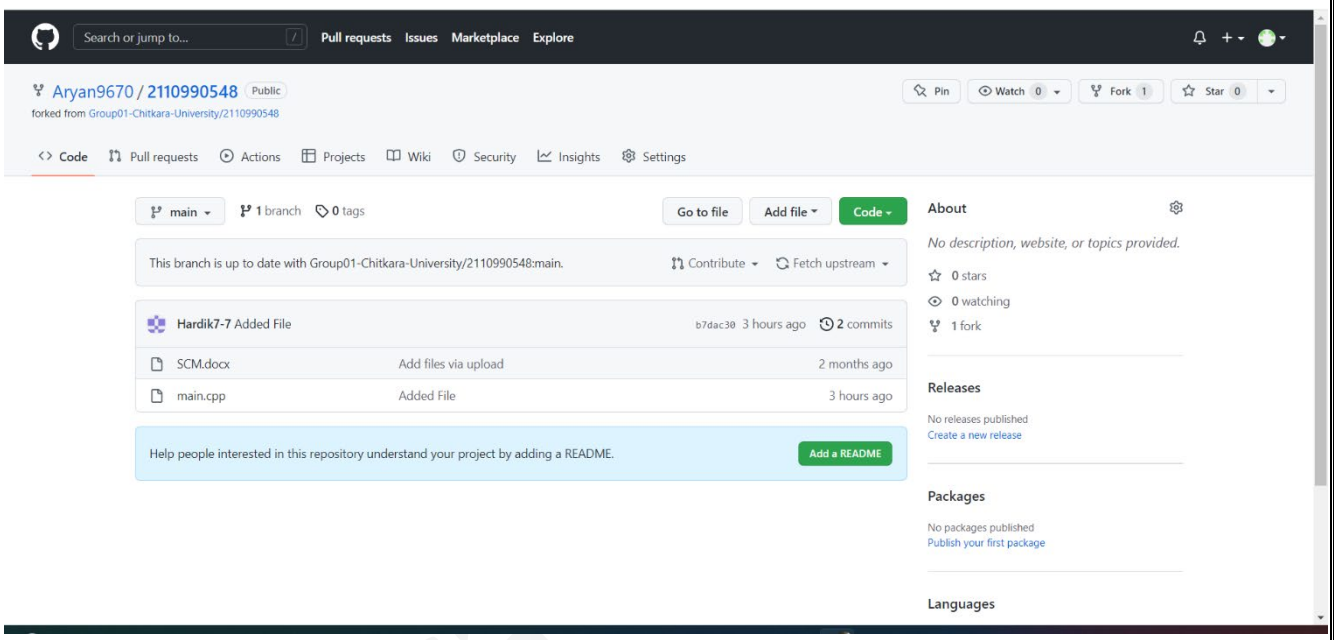



Open
Added Modulus #1
Hardik7-7 wants to merge 2 commits into Group01-Chitkara-Universi… :main from Hardik7-7:hardik
Hardik7-7 and others added 2 commits 12 minutes ago
Create README.md
Verified
8d2c4bf
Modified main.cpp file | From hardik branch
e606c23
Add more commits by pushing to the hardik branch on Hardik7-7/2110990017_SCM.
Continuous integration has not been set up
GitHub Actions and several other apps can be used to automatically catch bugs and enforce style.
This branch has no conflicts with the base branch
Merging can be performed automatically.
Merge pull request
You can also open this in GitHub Desktop or view command line instructions.
Write
Preview
Leave a comment
Attach files by dragging & dropping, selecting or pasting them.
Close pull request
Comment
Still in progress? Convert to draft
Assignees
No one—assign yourself
Labels
None yet
Projects
None yet
Milestone
No milestone
Development
Successfully merging this pull request may close these issues.
None yet
Notifications
Customize
Subscribe
You're not receiving notifications from this thread.
1 participant

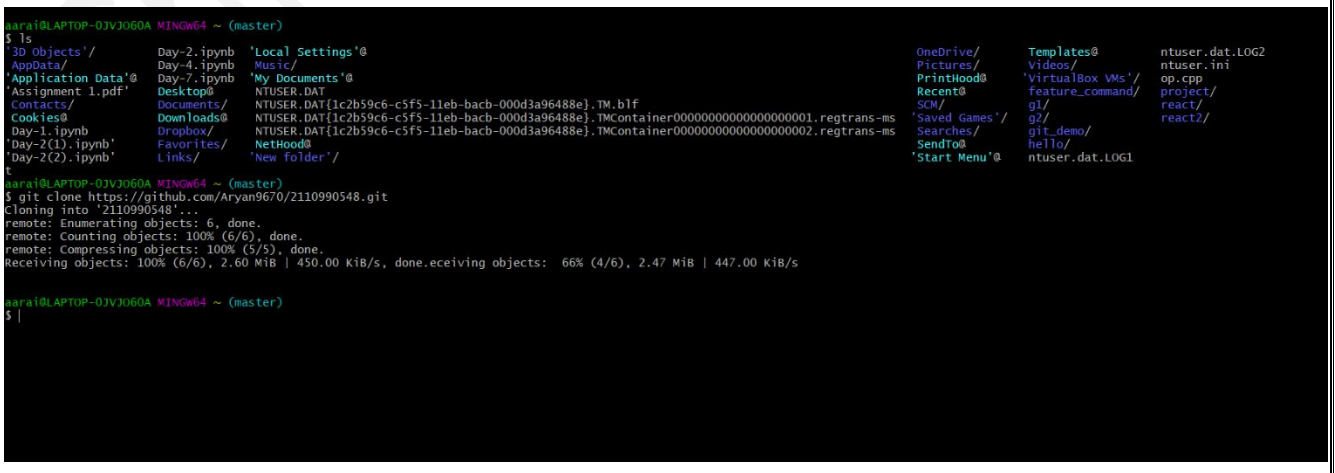

github.com/Group01-Chitkara-University/2110990548/pull/1
first commit #1
Edit
Code
Merged
Hardik7-7 merged 1 commit into Group01-Chitkara-University:main from Aryan9670:main now
Conversation 0
Commits 1
Checks 0
Files changed 1
+12 −8
Aryan9670 commented 4 minutes ago
Member
Tip
No description provided.
first commit
a2c94f4
Hardik7-7 merged commit b67d68b into Group01-Chitkara-University:main now
Revert
Write
Preview
Leave a comment
Attach files by dragging & dropping, selecting or pasting them.
Reviewers
No reviews
Assignees
No one—assign yourself
Labels
None yet
Projects
None yet
Milestone
No milestone
Development
ENG
IN
13:14
02-06-2022

FORK AND CLONING THE REPO'S :

1. Create fork by selecting yourself as a owner , and click on create fork.



2. Cloning in your remote system and making a branch to add your changes . and finally pulling a pull request .



```
aarai@LAPTOP-OJVJO6OA MINGW64 ~/2110990548 (main)
$ ls
SCM.docx  main.cpp

aarai@LAPTOP-OJVJO6OA MINGW64 ~/2110990548 (main)
$ nano main.cpp

aarai@LAPTOP-OJVJO6OA MINGW64 ~/2110990548 (main)
$ git add .

aarai@LAPTOP-OJVJO6OA MINGW64 ~/2110990548 (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)
        modified:   main.cpp


aarai@LAPTOP-OJVJO6OA MINGW64 ~/2110990548 (main)
$ git commit -m "first commit"
[main a2c94f4] first commit
 1 file changed, 12 insertions(+), 8 deletions(-)

aarai@LAPTOP-OJVJO6OA MINGW64 ~/2110990548 (main)
$ git push origin main
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 16 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 454 bytes | 454.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/Aryan9670/2110990548.git
   b7dac30..a2c94f4  main -> main

aarai@LAPTOP-OJVJO6OA MINGW64 ~/2110990548 (main)
$
```

3. Create a pull request by clicking on compare and pull request.