Subject Name: **Source Code Management**

Subject Code: **CS181**

Cluster: **Beta**

Department: **CSE**



**Submitted To-**

Dr. Monit Kapoor

**Submitted By-**

Abhijeet Sambharwal

2110990040

G01

# **INDEX**

1. Add collaborators on GitHub Repository.

2. Fork and Commit.

3. Merge and Resolve conflicts created due to own activity and collaborators activity.

4. Reset and Revert.

# **ADD COLLABORATORS ON GITHUB REPOSITORY.**

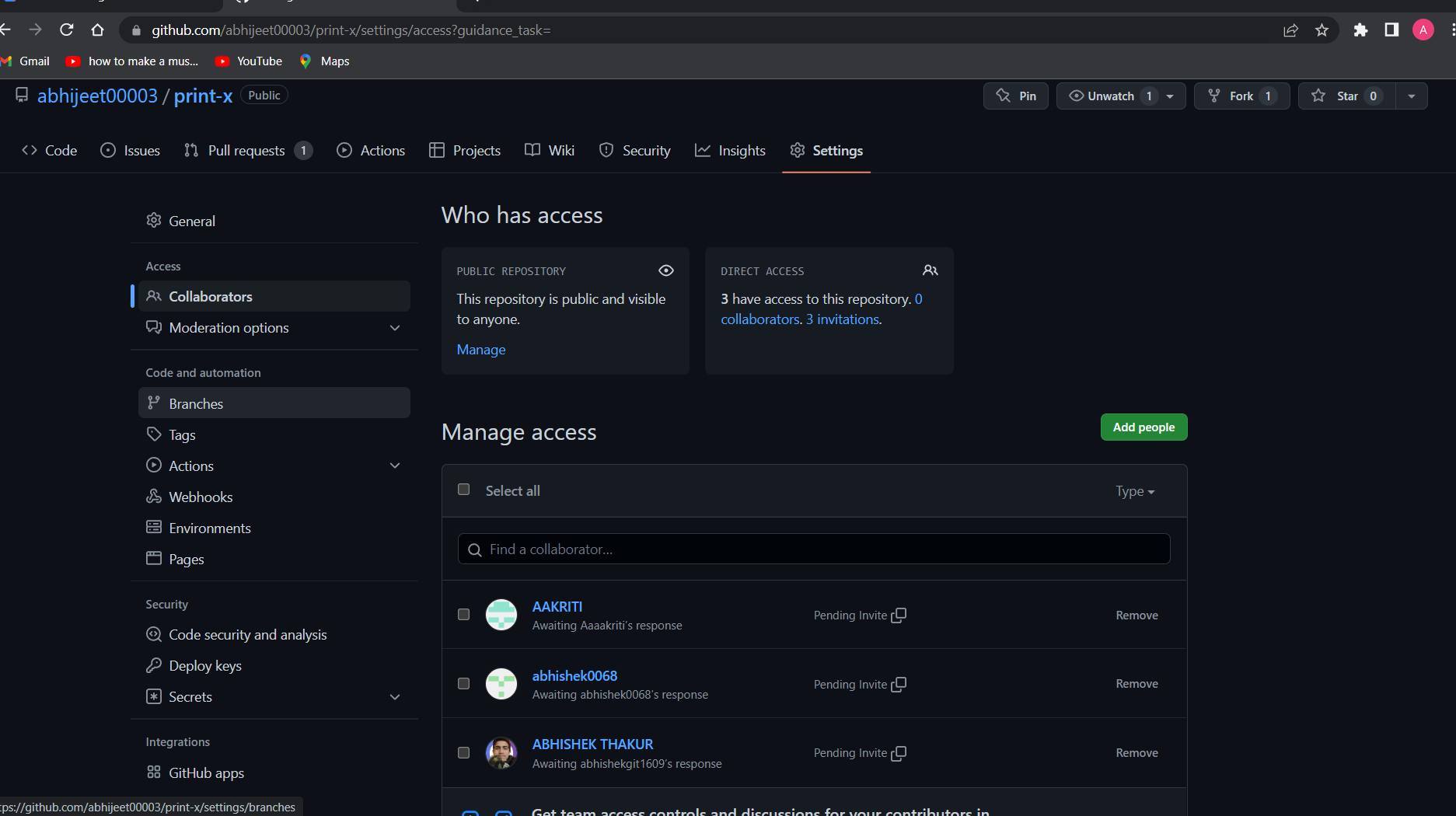
In GitHub, we can invite other GitHub users to become collaborators to our private repositories (which expires after 7 days if not accepted, restoring any unclaimed licenses). Being a collaborator, of a personal repository you can pull (read) the contents of the repository and push (write) changes to the repository. You can add unlimited collaborators on public and private repositories.

Collaborators can perform a number of actions into someone else’s personal repositories, they have gained access to. Some of them are,

1. Create, merge, and close pull requests in the repository
2. Publish, view, install the packages
3. Fork the repositories
4. Make the changes on the repositories as suggested by the Pull requests.
5. Mark issues or pull requests as duplicate
6. Create, edit, and delete any comments on commits, pull requests, and issues in the repository
7. Removing themselves as collaborators on the repositories.
8. Manage releases in the repositories.

## STEPS TO ADD COLLABORATORS:

1. Navigate to the repository on GitHub you wish to share with your collaborator.
2. Click on the "Settings" tab on the right side of the menu at the top of the screen.
3. On the new page, click the "Collaborators" menu item on the left side of the page.
4. Start typing the new collaborator's GitHub username into the text box.
5. Select the GitHub user from the list that appears below the text box.
6. Click the "Add" button.

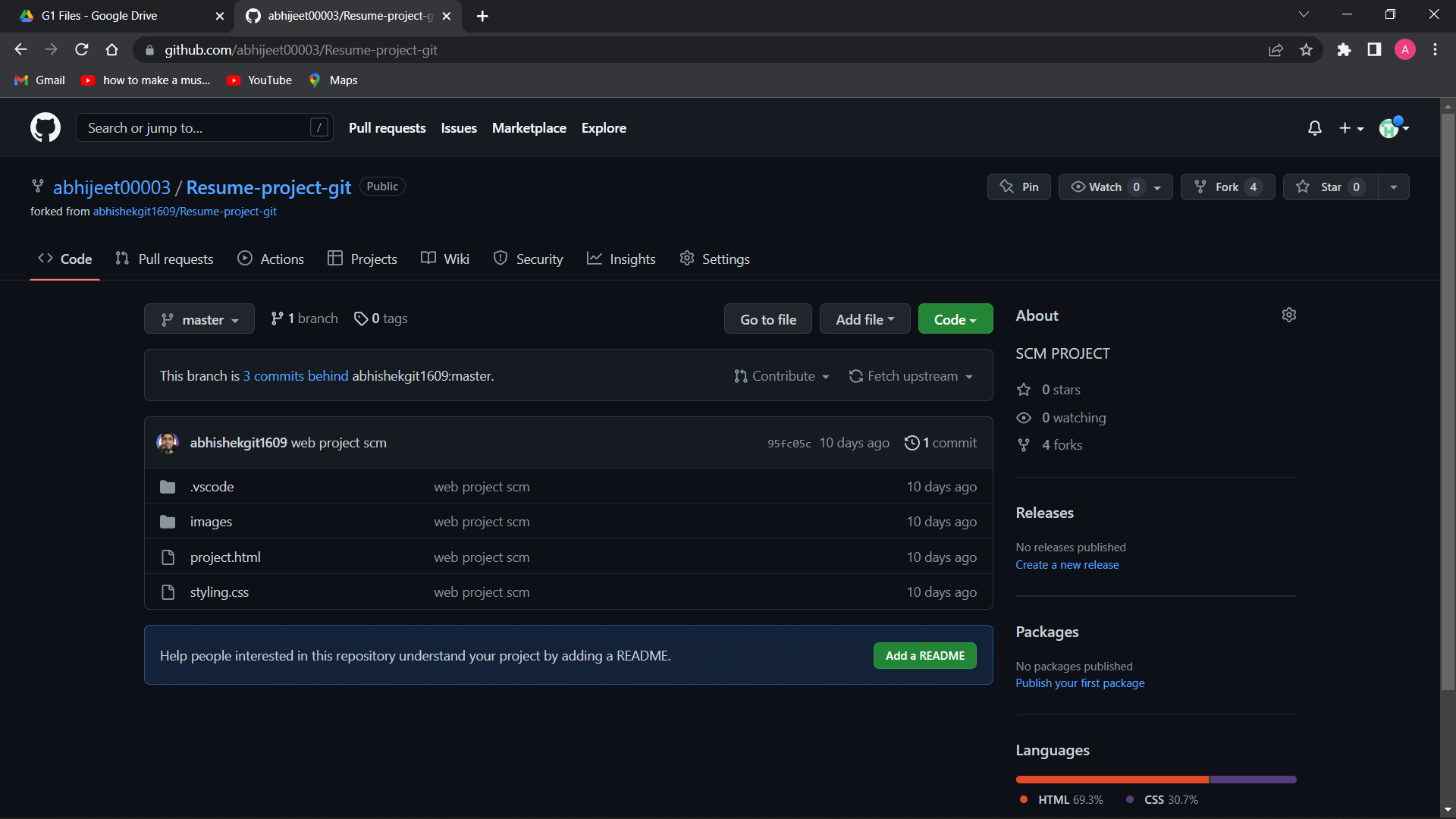


# **FORK AND COMMIT**

A fork is a copy of a repository. Forking a repository allows you to freely experiment with changes without affecting the original project. Most commonly, forks are used to either propose changes to someone else's project to which you do not have write access, or to use someone else's project as a starting point for your own idea.

## STEPS TO FORK A REPO

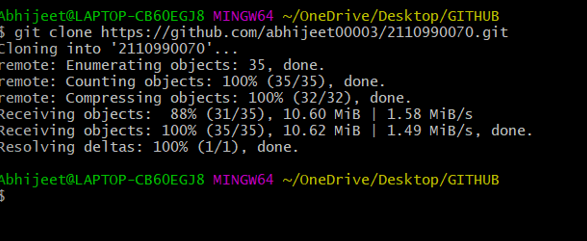
1. Go to the repository that you wish to fork.
2. Click on the option ‘Fork’ in the top right corner
3. Name your forked repository and click on ‘Create New Fork’.



1. You now have a forked repository.

## CLONING THE REPO INTO YOUR DEVICE

When you create a repository on GitHub.com, it exists as a remote repository. You can clone your repository to create a local copy on your computer and sync between the two locations.

1. Once you have forked the repository, you can clone it into your computer using directly the option given on GitHub or through running git clone command in git bash.
2. Copy the URL of the forked repository
3. Open git bash and type the command “git clone <URL of the forked repository>”

## 

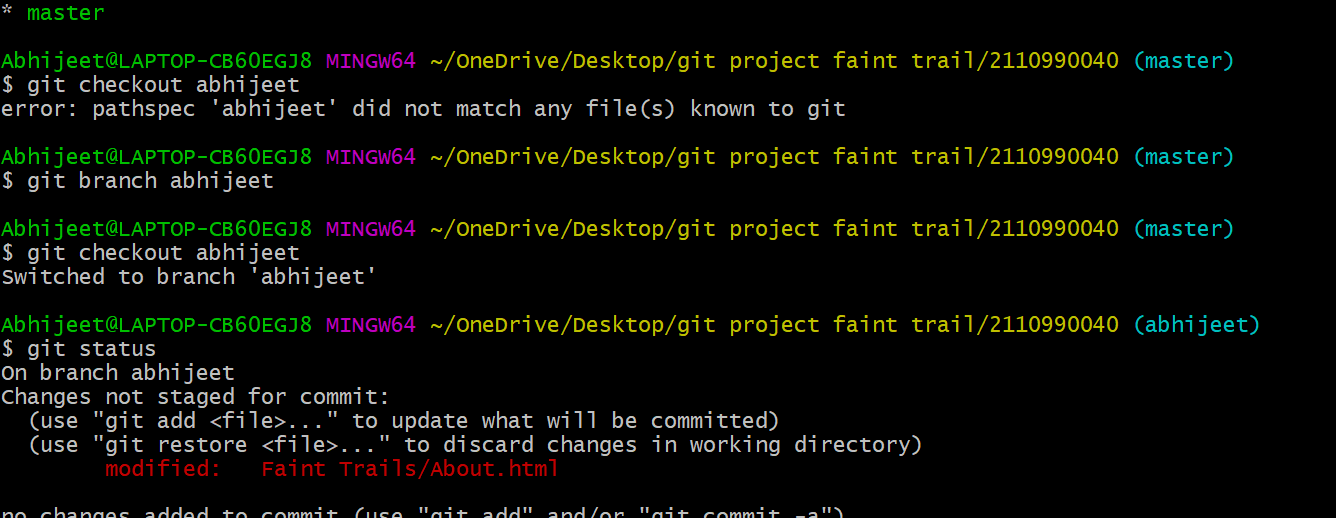
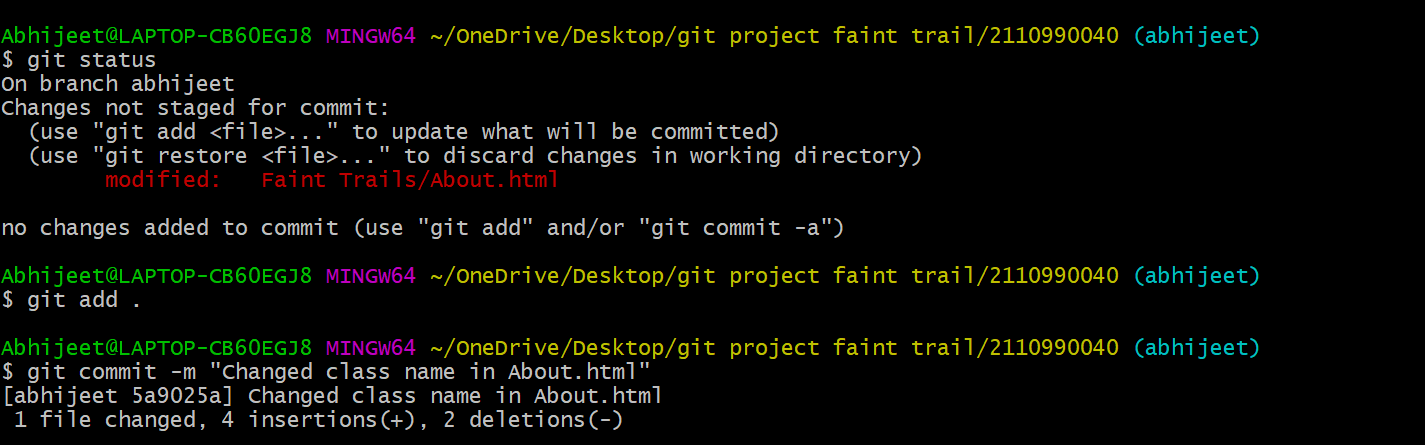
## COMMITING CHANGES TO THE FORKED REPOSITORY

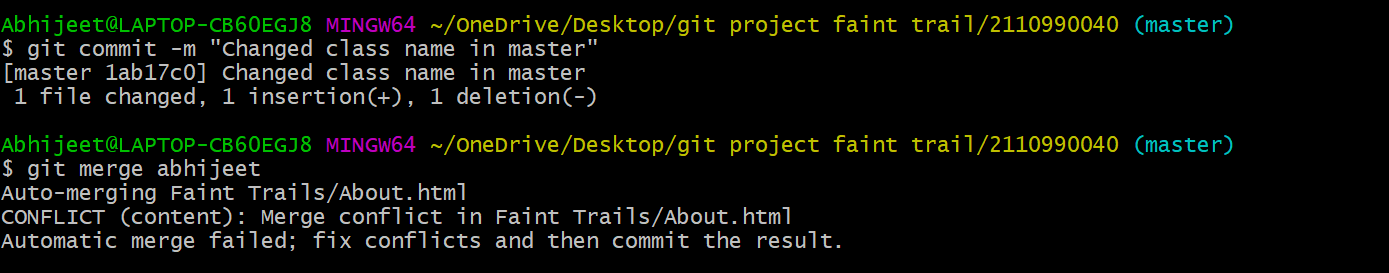
1. Once you have cloned the repository you can introduce changes to it as per your wish.
2. After changing it you have to stage the file and then commit it.
3. After committing changes push it to your remote repository.

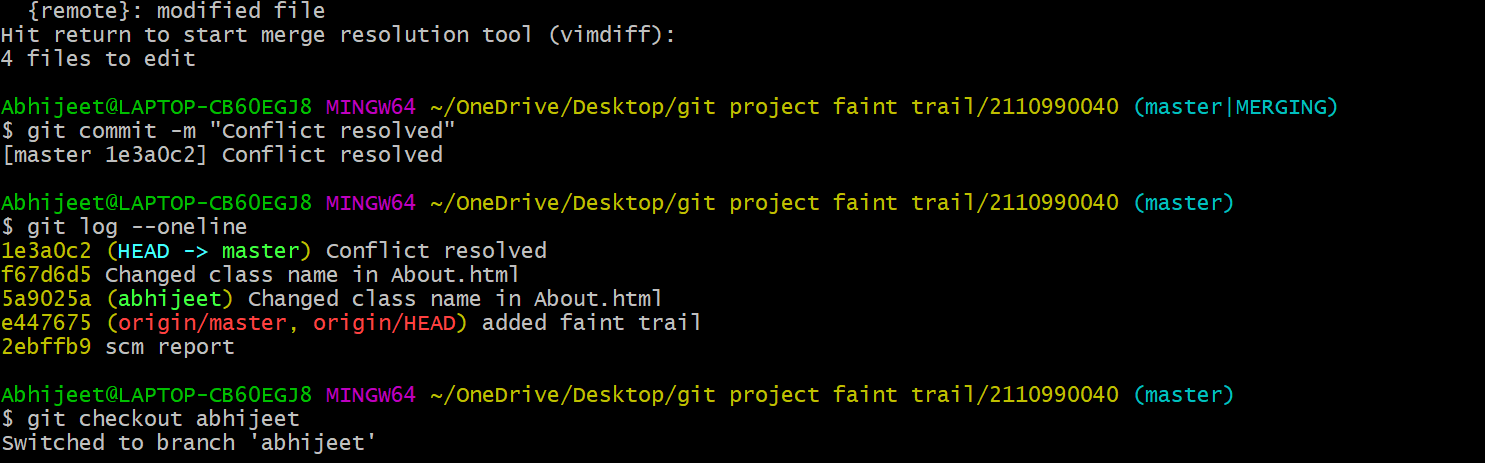
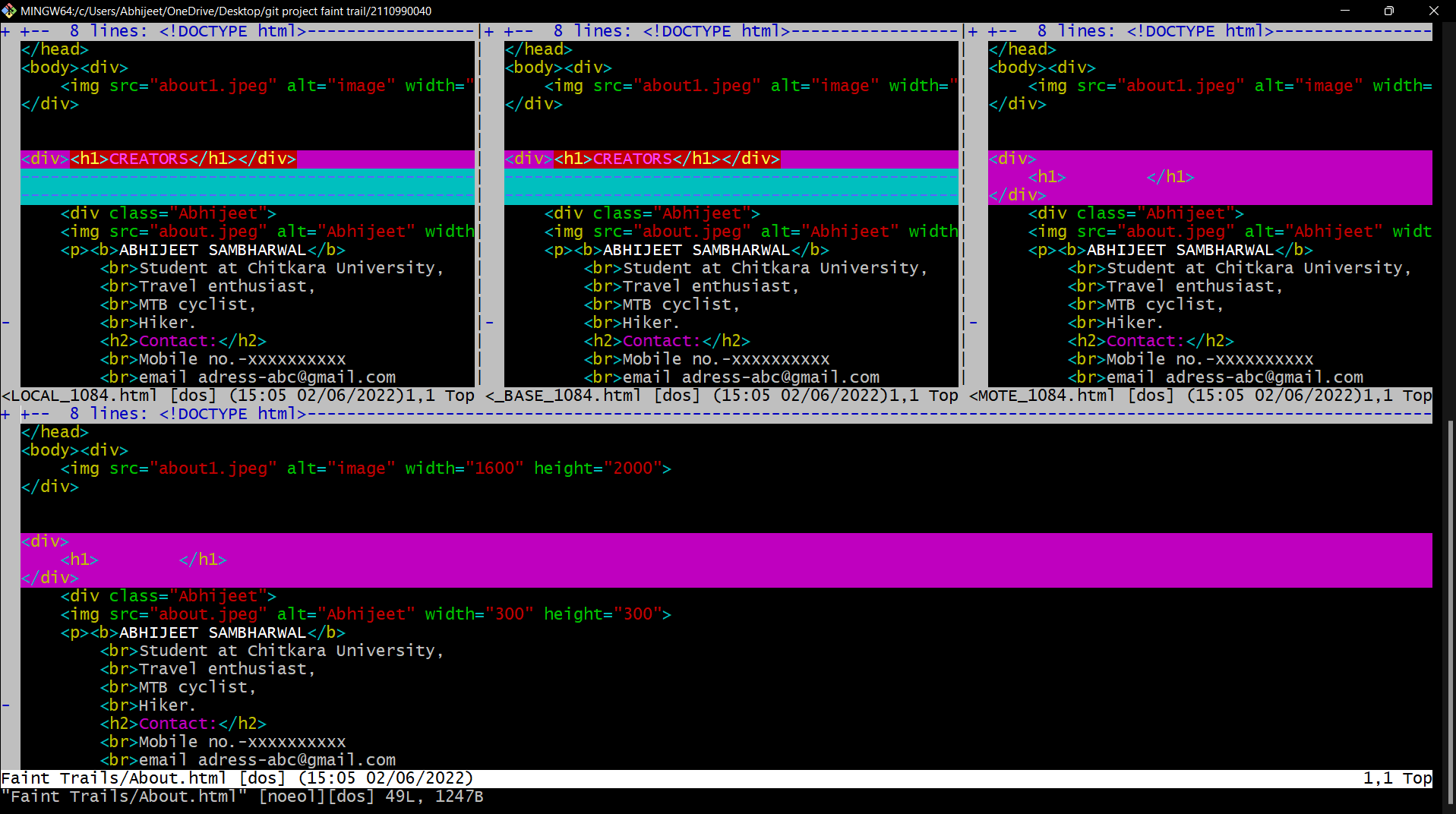
# **MERGE AND RESOLVE CONFLICTS CREATED DUE TO OWN ACTIVITY AND COLLABORATORS ACTIVITY**

Merging and conflicts are a common part of the Git experience. Conflicts generally arise when two people have changed the same lines in a file, or if one developer deleted a file while another developer was modifying it. In these cases, Git cannot automatically determine what is correct. Conflicts only affect the developer conducting the merge, the rest of the team is unaware of the conflict. Git will mark the file as being conflicted and halt the merging process. It is then the developers' responsibility to resolve the conflict.

1. To understand the merging concept of branches, create a branch named ‘feature’ in your repository.
2. Here, there is a file called ‘first.cpp’. Make changes to it, add and commit them.



1. Similarly, change the same lines of first.cpp file in the master branch.
2. If you are not already on the branch that you want the other one to merged in (in this example master branch), then switch to it.
3. Using the command try merging feature branch into master branch using the “git merge <branch name>”
4. Auto merging fails and conflict arises. In order to resolve it we make use of the mergetool by running the command “git mergetool”. The mergetool editor will open.
5. Make changes as per requirement in order to resolve the conflicts and exit the editor. 



# **RESET AND REVERT**

While Working with Git in certain situations we want to undo changes in the working area or index area, sometimes remove commits locally or remotely and we need to reverse those changes. We can do it by using the git reset, git revert, git checkout commands.

RESET-

git reset is used when we want to unstage a file and bring our changes back to the working directory. git reset can also be used to remove commits from the local repository.

Suppose we make edits to a file, stage it and commit it.

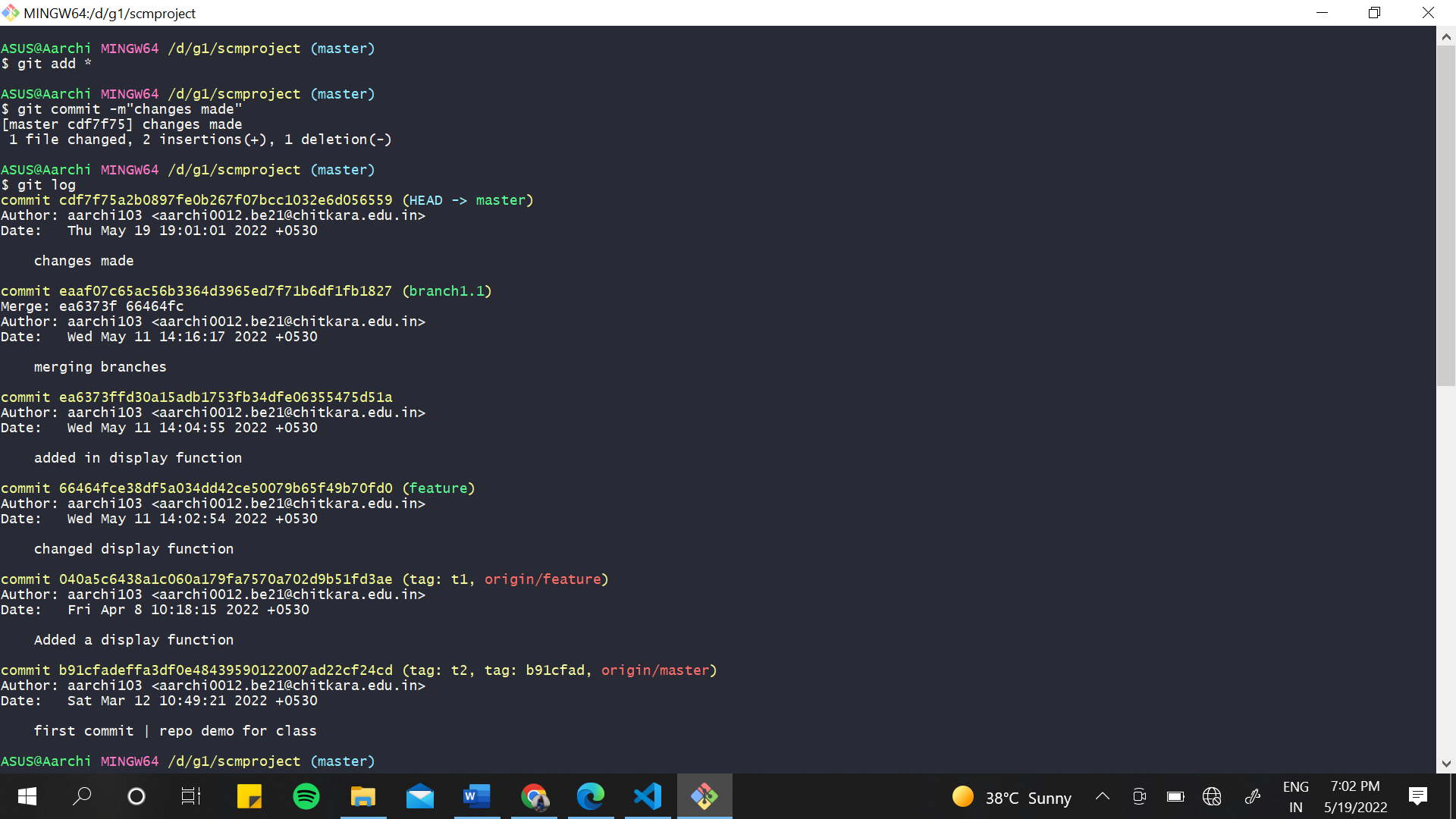
In order to reset the changes made in the recent commit, run the “git reset --hard HEAD~1” command.

The HEAD returns to the previous commit and the changes made are reset.

REVERT-

git revert is used to remove the commits from the remote repository. git revert removes the commit that we have done but adds one more commit which tells us that the revert has been done.

In order to understand it add changes to a file, stage and commit it.

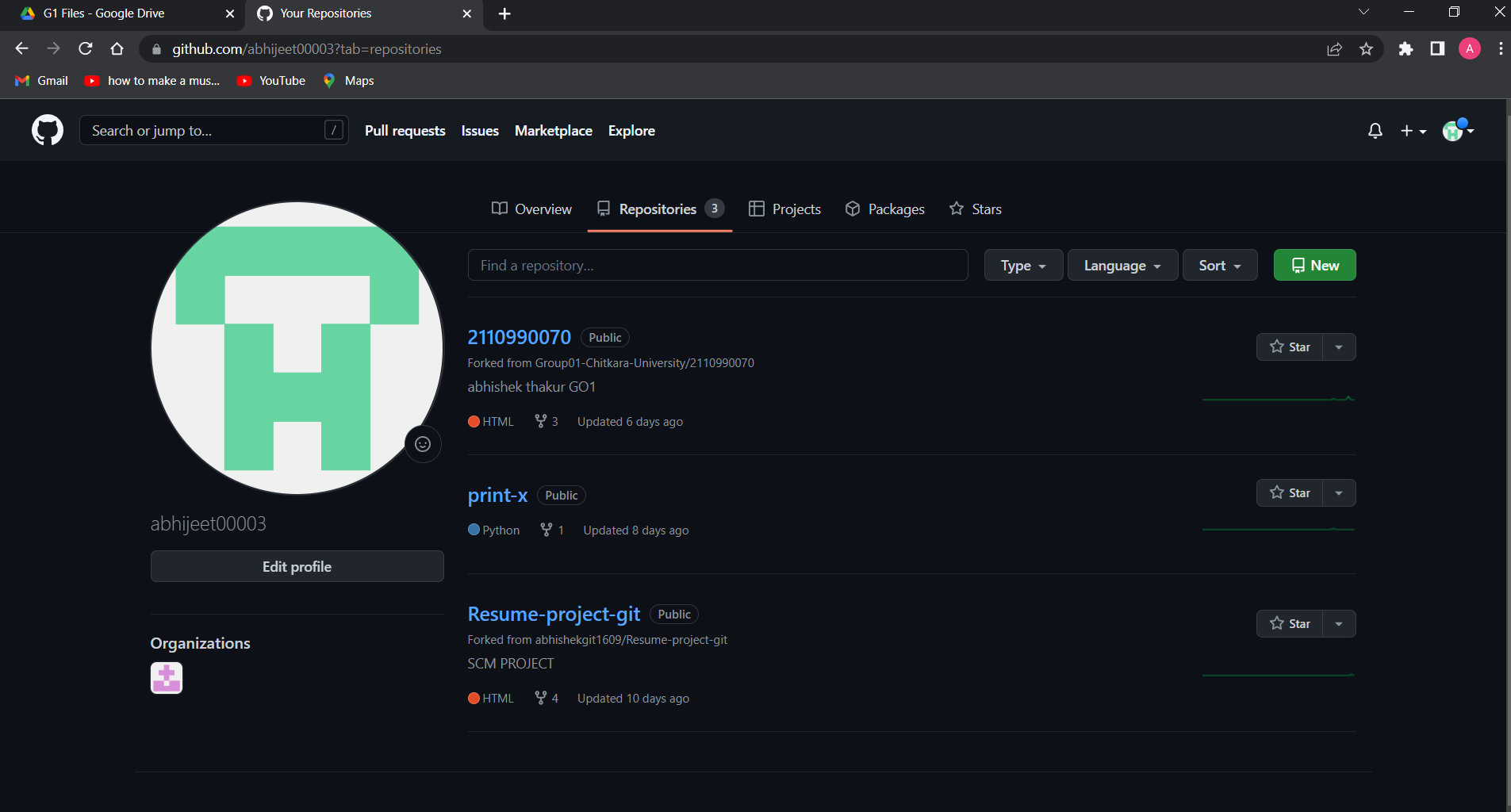


Now to revert the changes made in the commit run the “git revert <commit id>” command.

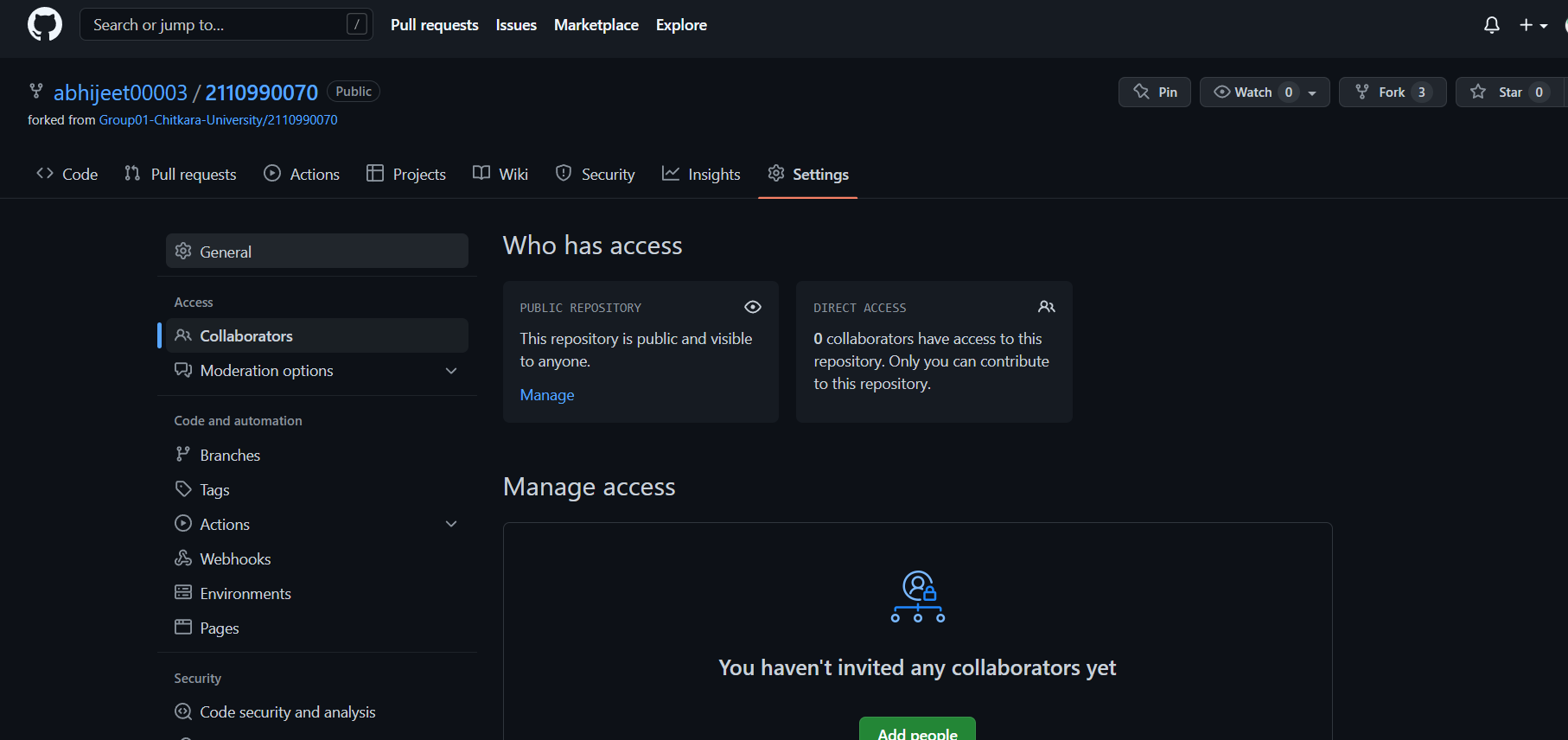
You can see that a new commit as ‘revert “changes made”’ is there and the file has returned to its previous state.

# **CREATE A DISTRIBUTED REPOSITORY AND ADD MEMBERS IN PROJECT TEAM**

1. On the homepage of your GitHub account, click on Repositories option in the menu bar.
2. Click on the ‘New’ button in the top right corner.



1. Enter the Repository name and add the description of the repository.
2. To add members to your repository, open your repository and select settings option in the navigation bar.
3. Click on Collaborators option under the access tab.

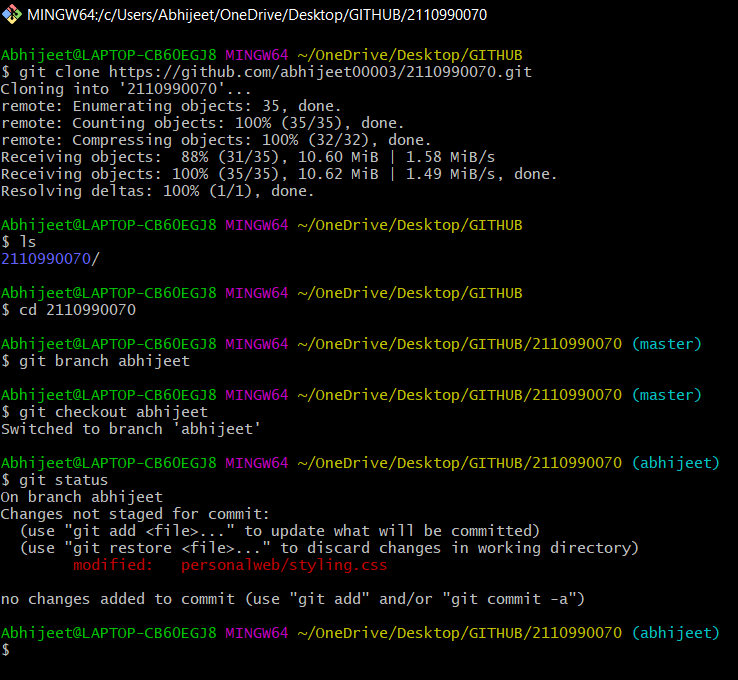


1. You can manage access and add/remove team members to your project.
2. To add members, click on the add people option and search the id of your respective team member.
3. To accept the invitation from your team member, open your email registered with GitHub.
4. You will receive an invitation mail from the repository owner. Open the email and click on accept invitation.
5. You will be redirected to GitHub where you can either select to accept or decline the invitation.

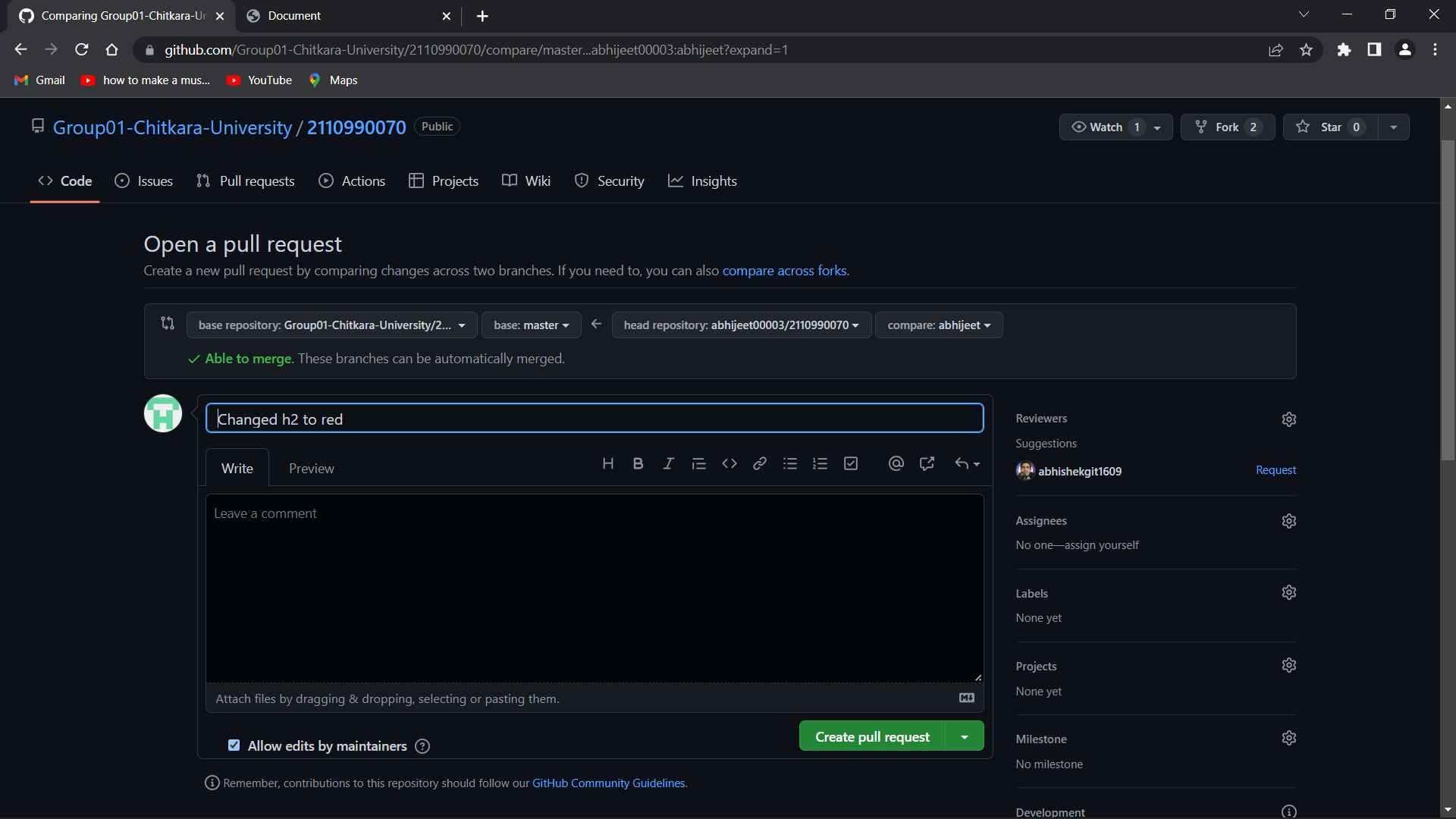
Similarly, you can add more collaborators to your project.

# **OPEN AND CLOSE A PULL REQUEST**

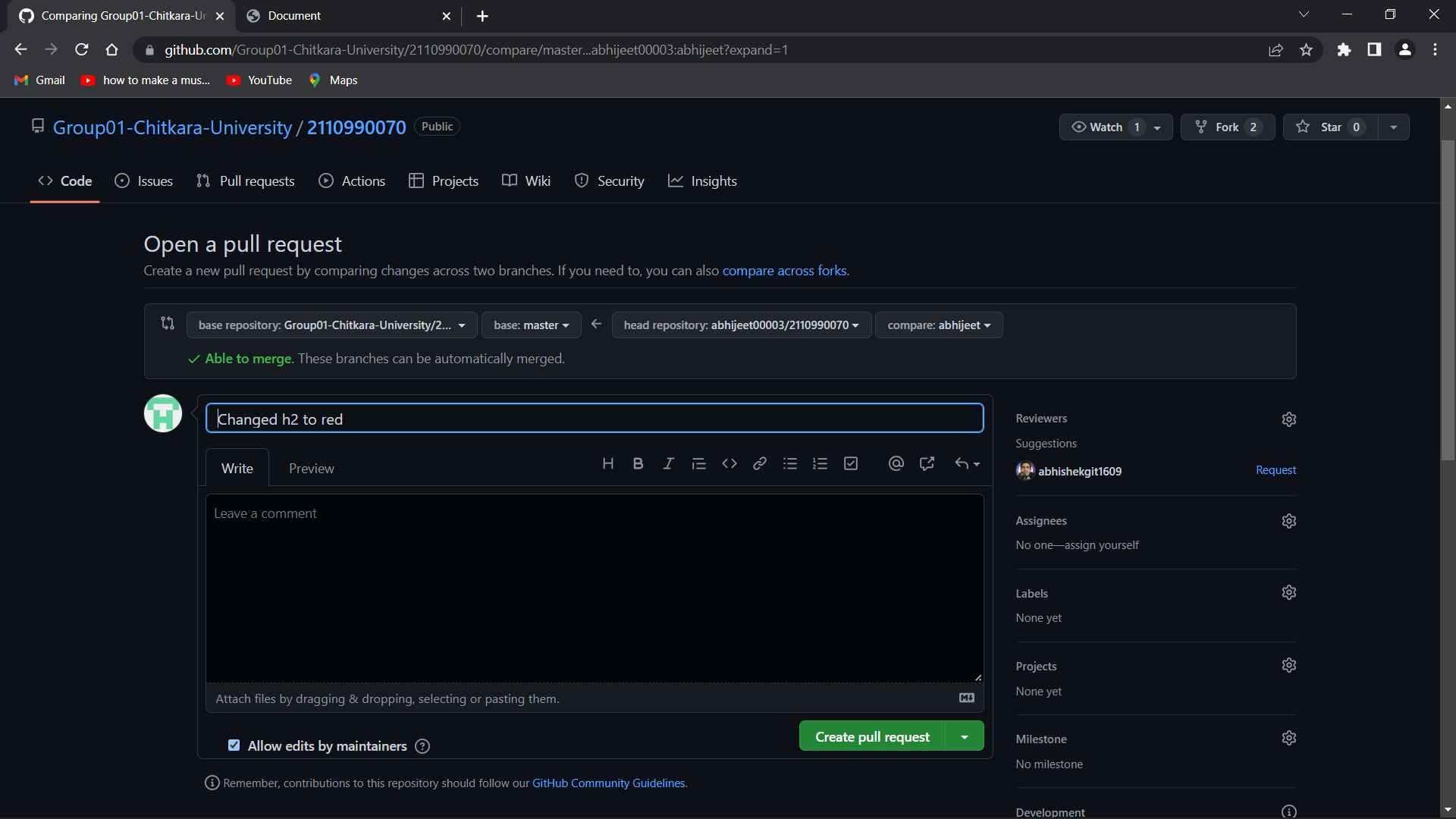
1. First, select a repository of the other person in which you want to make changes and create a pull request.
2. Clone it into your local storage.
3. To open a pull request we first have to make a new branch, by using git checkout -b *branch name* option.
4. After making new branch we add a file to the branch or make changes in the existing file.
5. Add and commit the changes to the local repository.



1. Use git push origin *branch name* option to push the new branch to the main repository.
2. After pushing new branch GitHub will either automatically ask you to create a pull request or you can create your own pull request.
3. To create your own pull request, click on pull request option.



1. GitHub will detect any conflicts and ask you to enter a description of your pull request.
2. After opening a pull request the owner of the original repository will be sent the request if they want to merge or close the request.

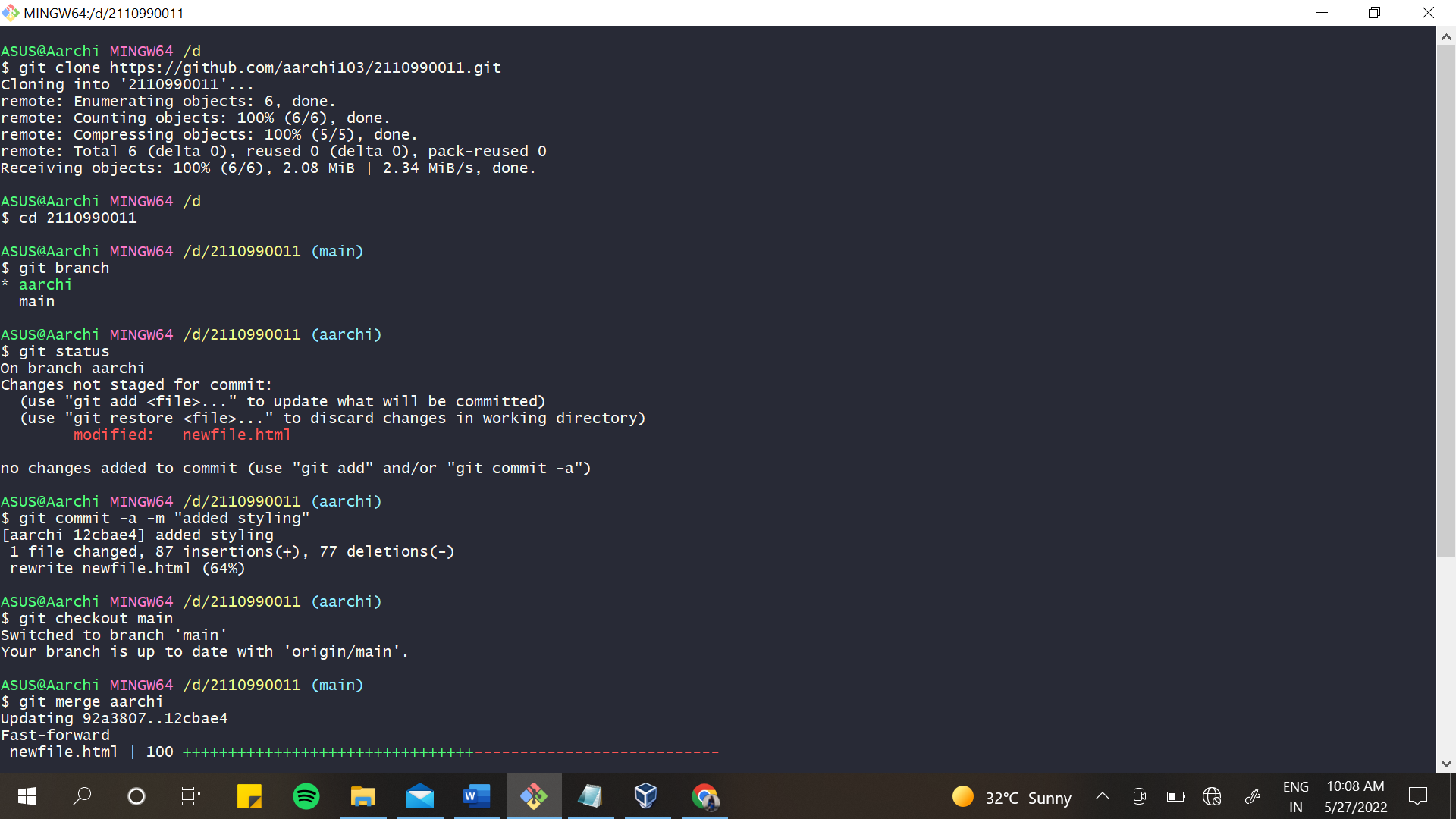


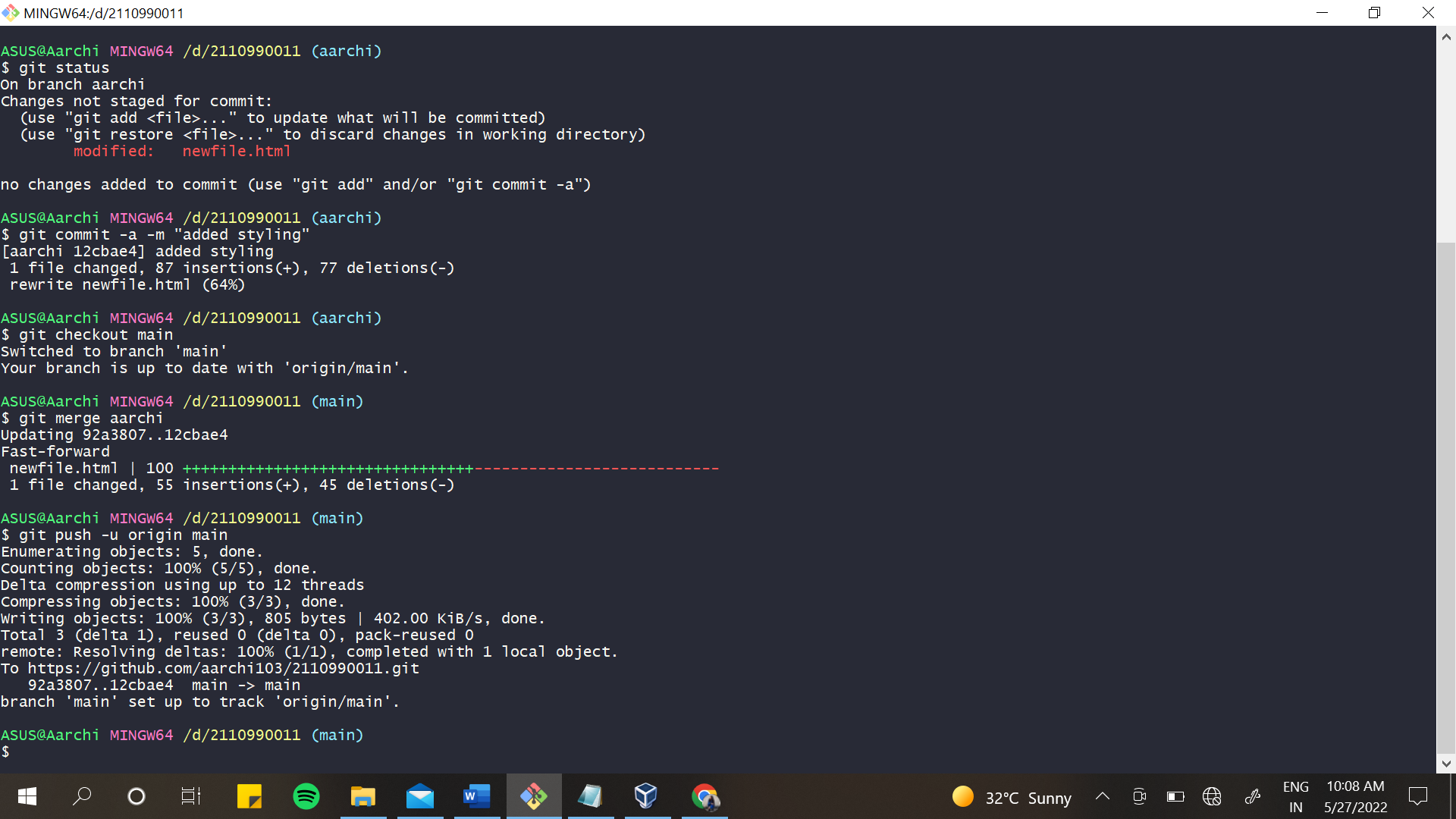
1. If the owner chooses not to merge your pull request, they will close it.
2. To close the pull request simply click on close pull request and add comment/ reason why you closed the pull request.
3. If you want to merge it into the original, click on merge pull request.

# **CREATE A PULL REQUEST ON A TEAM MEMBER’S REPO AND CLOSE PULL REQUESTS GENERATED BY TEAM MEMBERS ON OWN REPOSITORY AS A MAINTAINER**

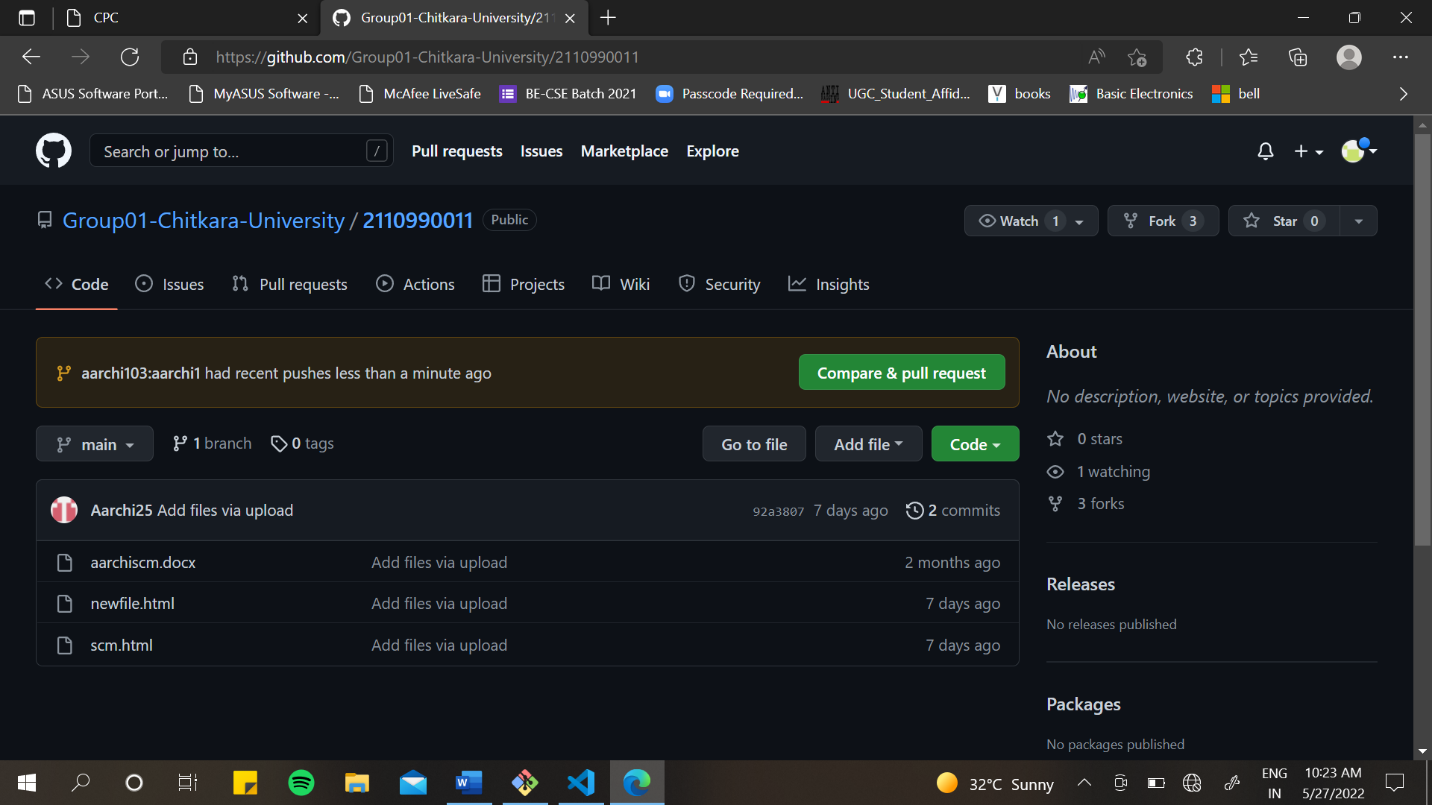
#OPENING PULL REQUESTS ON TEAM MEMBER’S REPOSITORY

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 branchname.

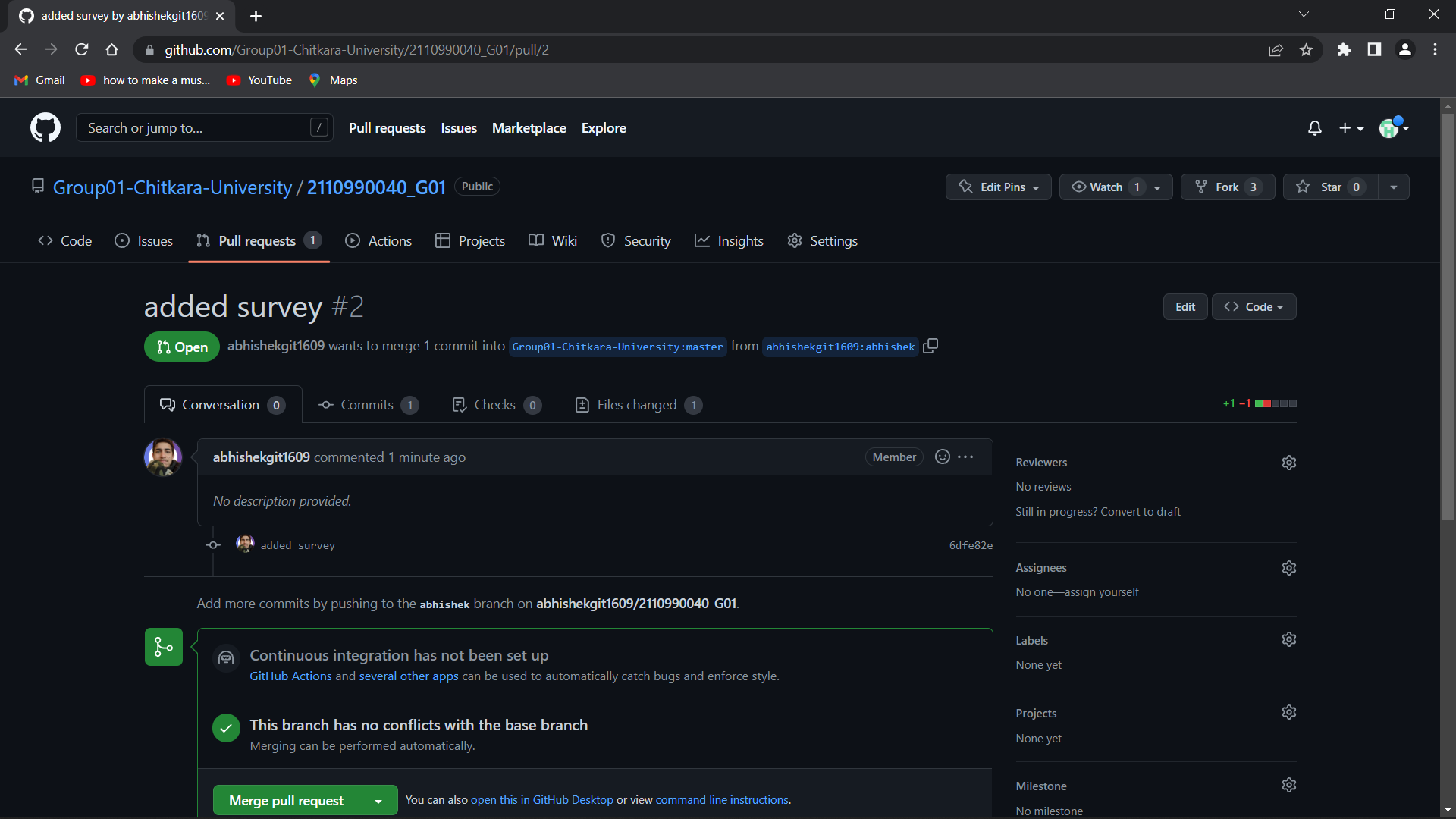




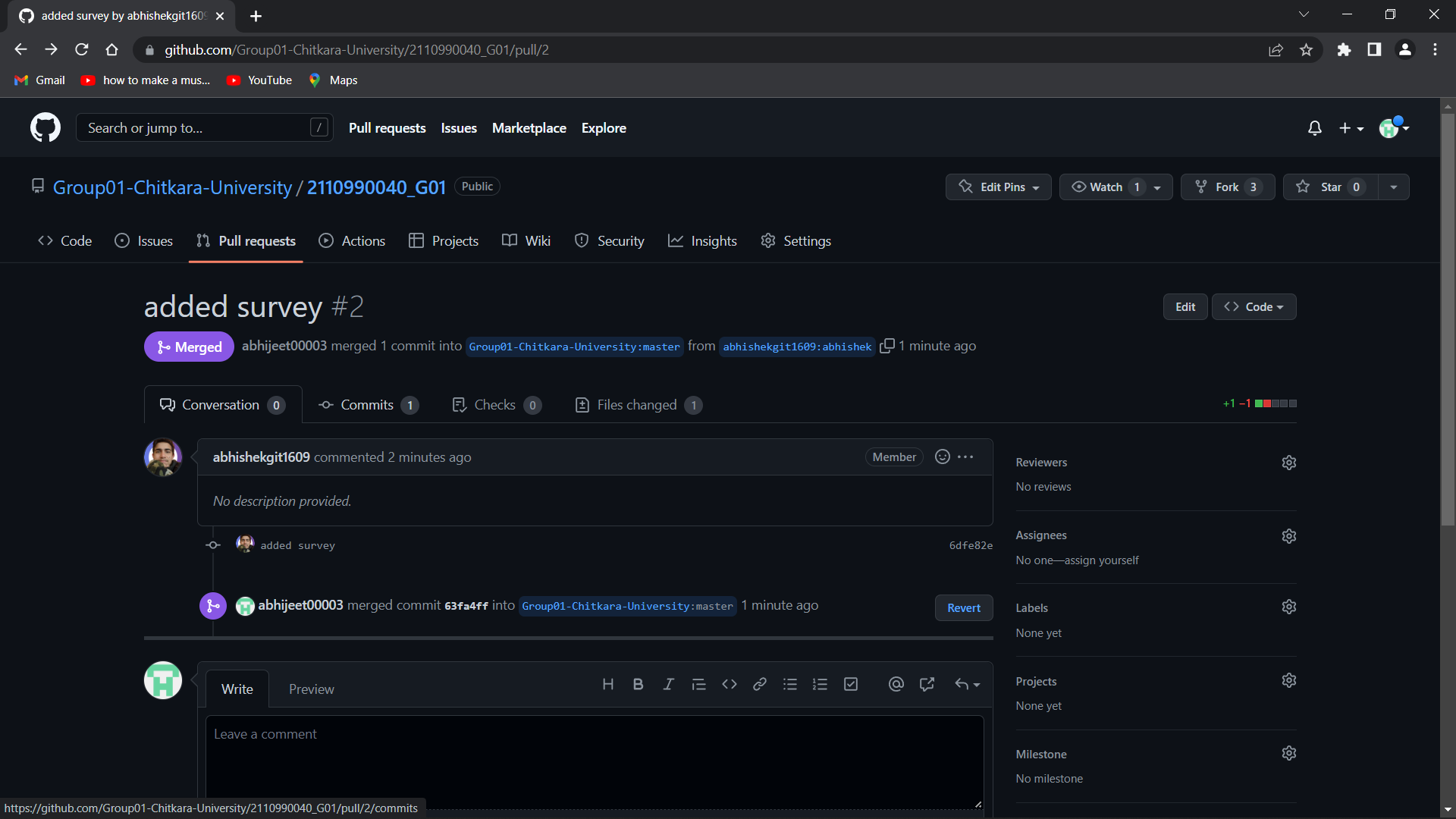
1. Open a pull request by following the procedure from the above experiment.
2. The pull request will be created and will be visible to all the team members.



1. Click on compare & pull request option appearing at the top.
2. After you enter a description, click on the open request button.

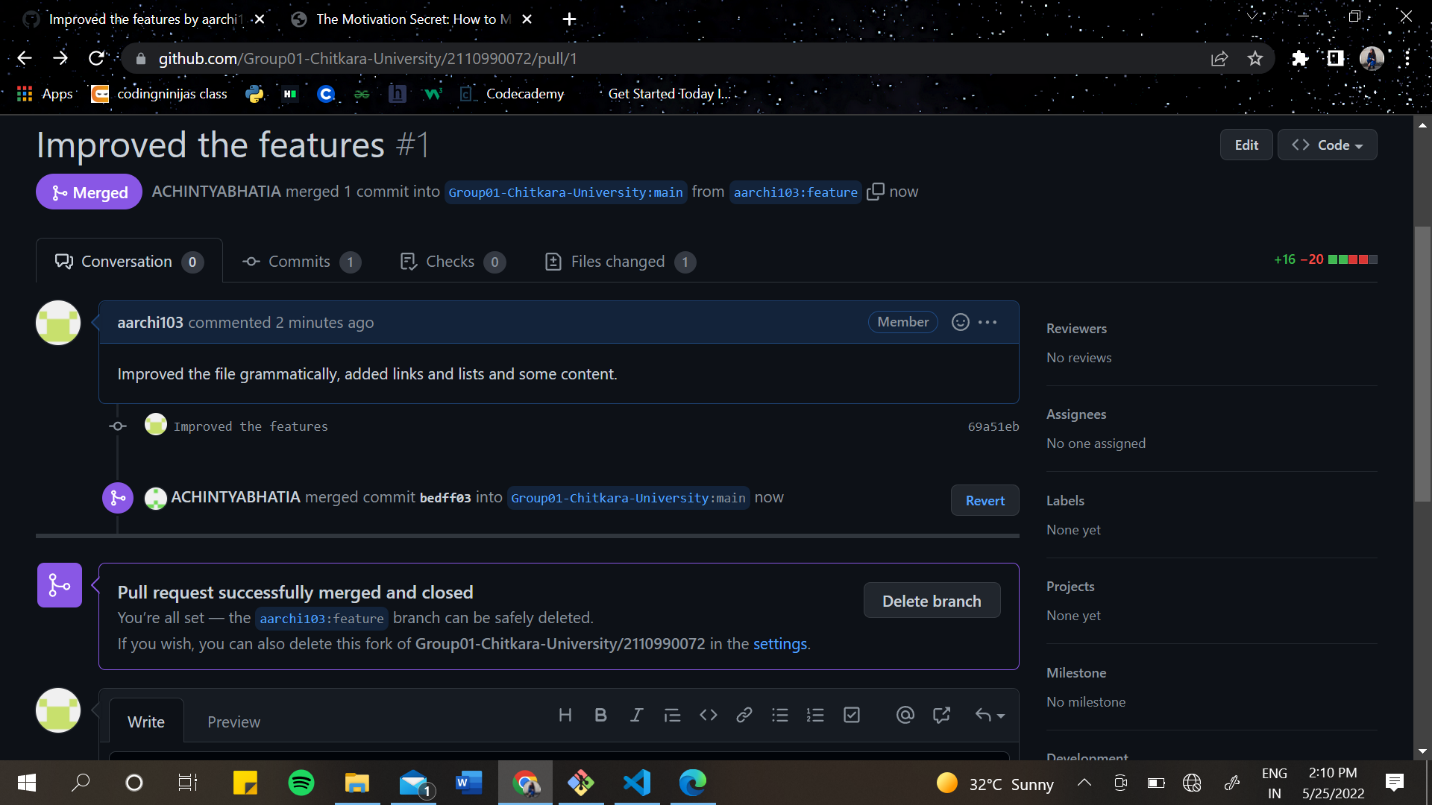


1. Pull request has been sent to your team member. They can choose to either close the pull request or merge it.
2. Suppose they merged it, then the changes you made to the forked repository will be introduced into the owner’s original repository and you will be notified about merging.

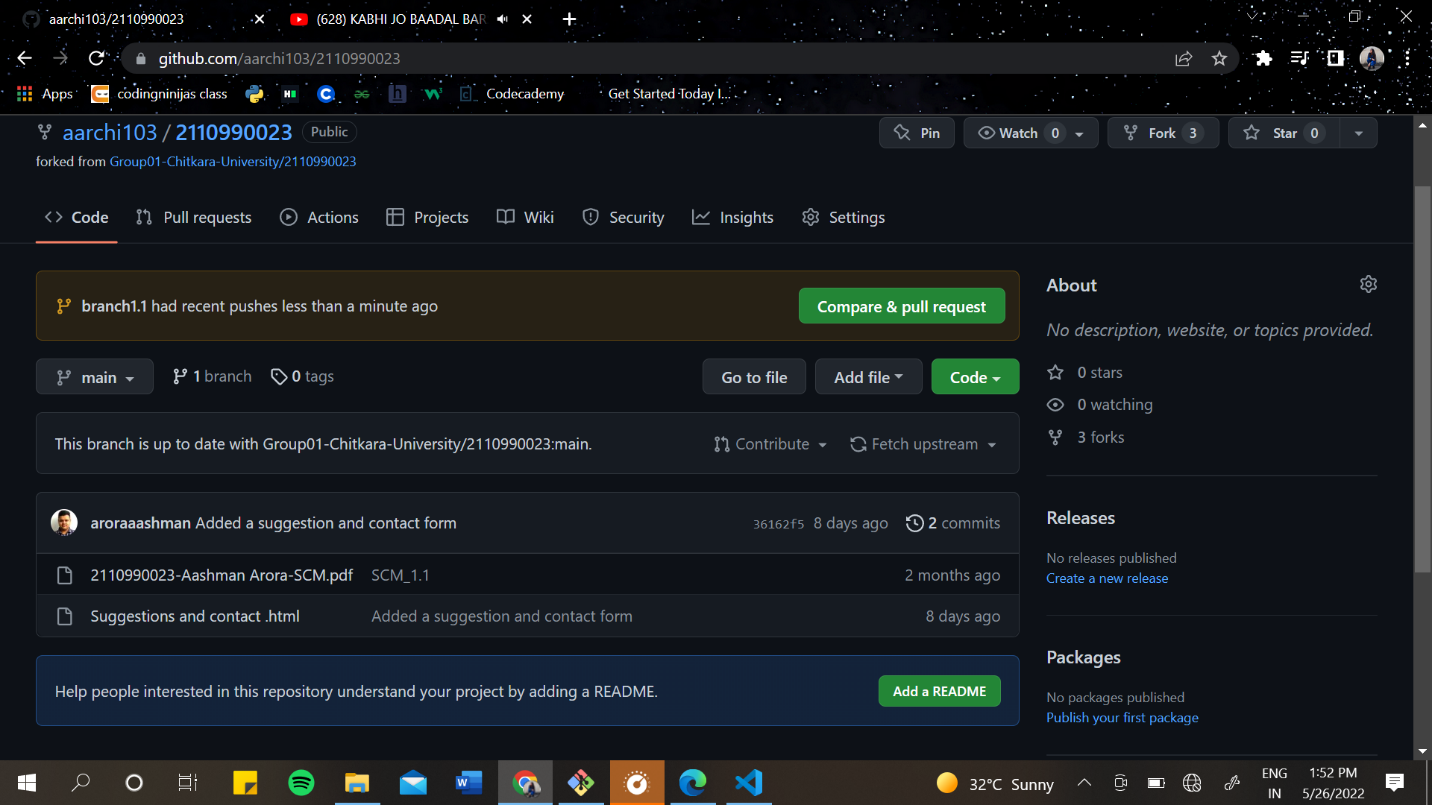


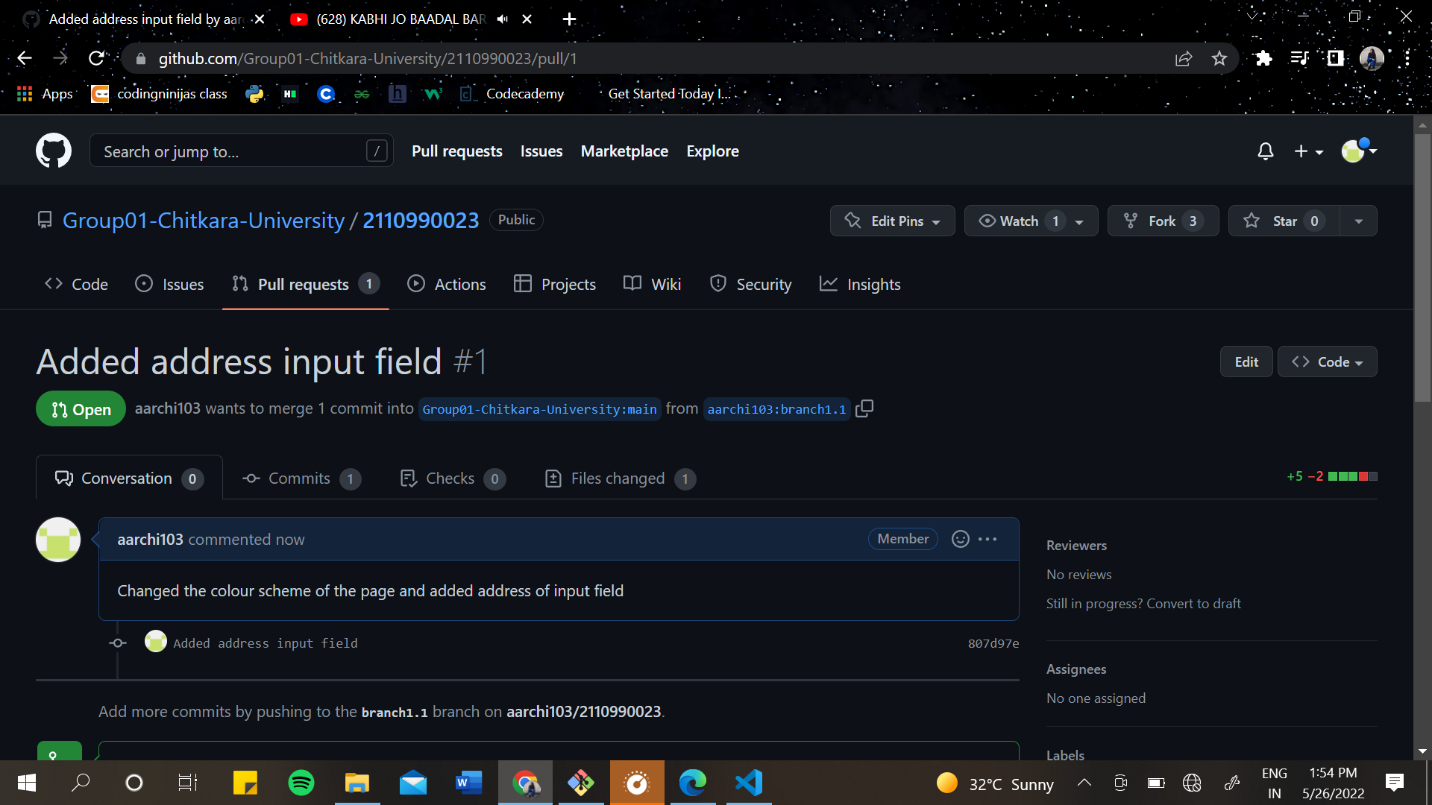
Similarly, create pull requests on other members repositories.

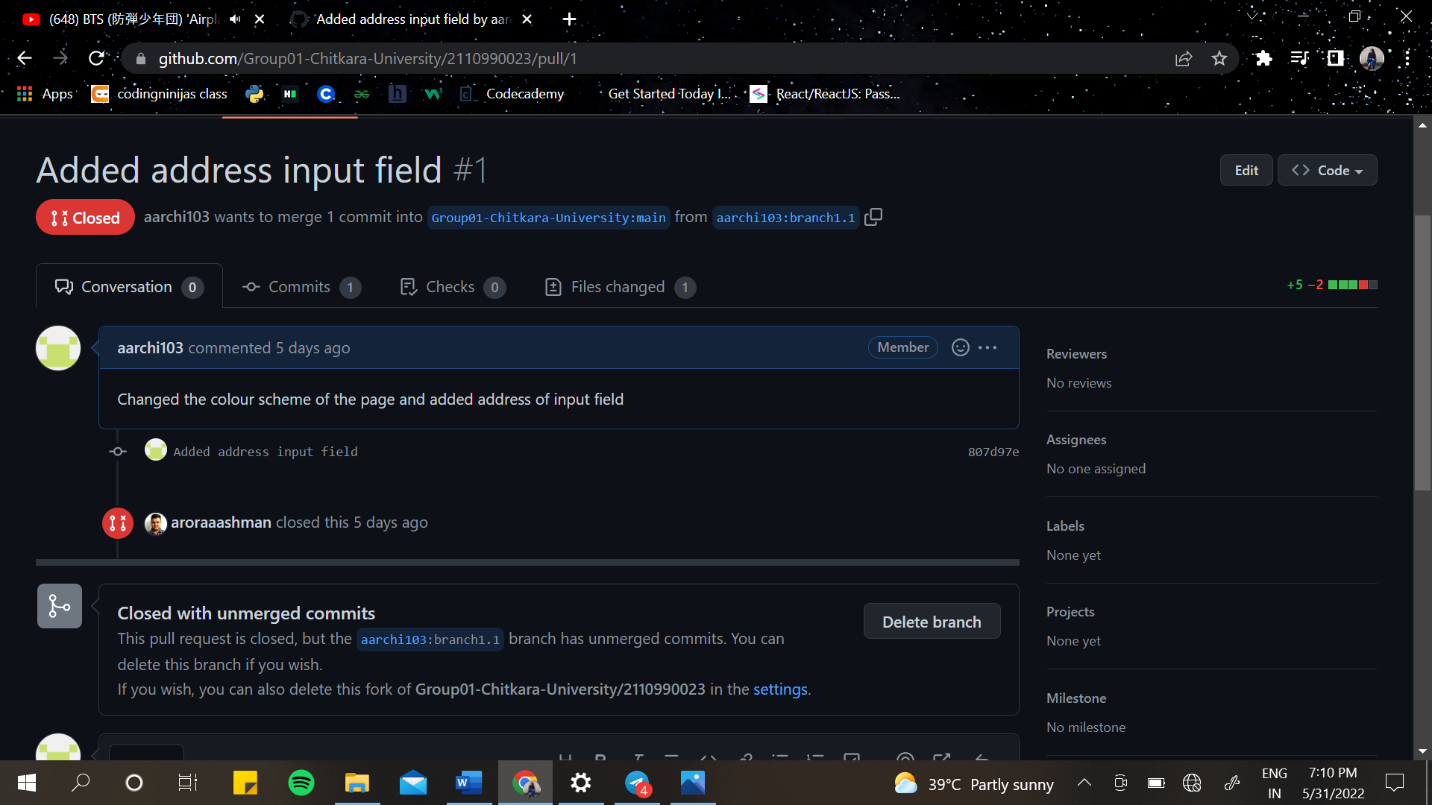
On 2nd team member’s repository:



On 3rd team member’s repository



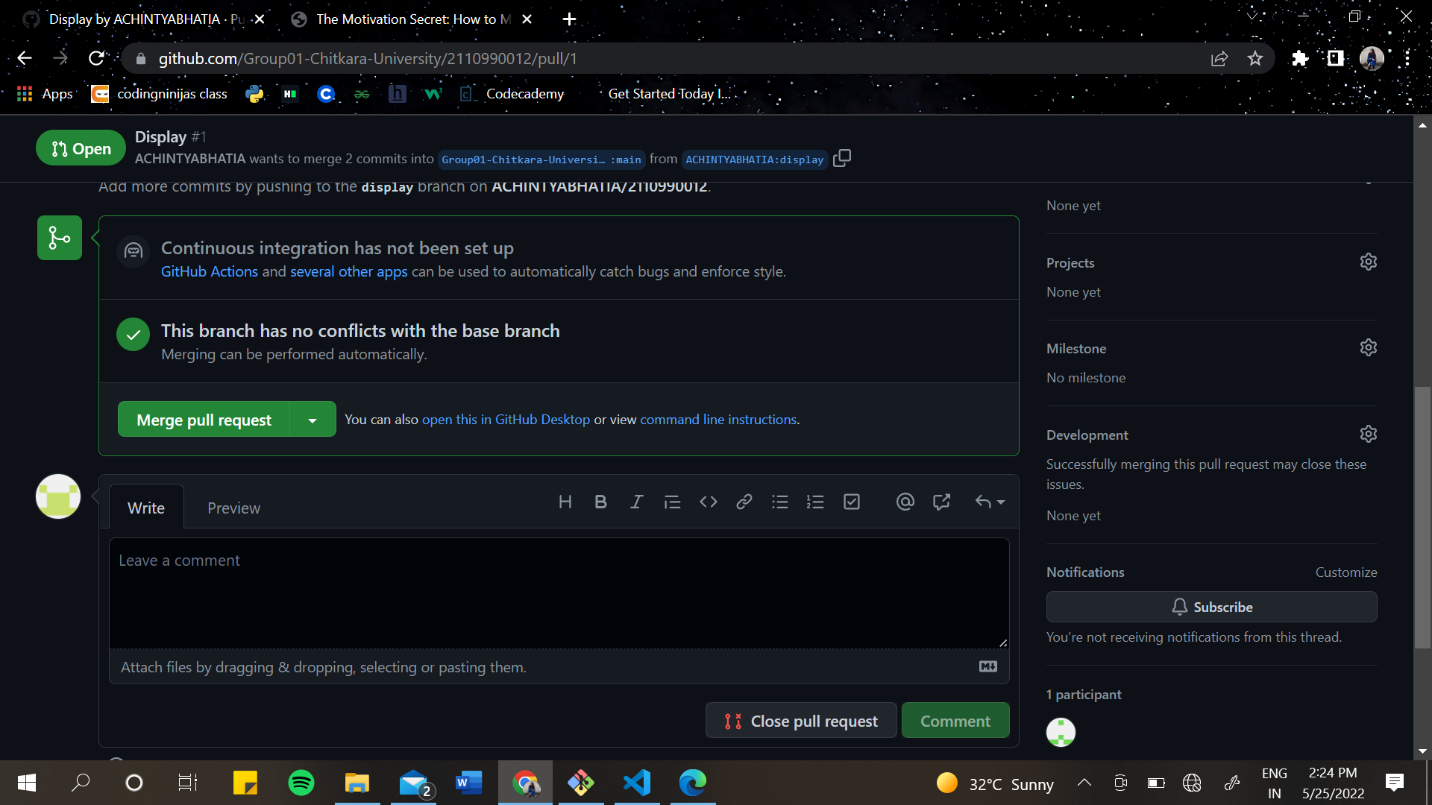




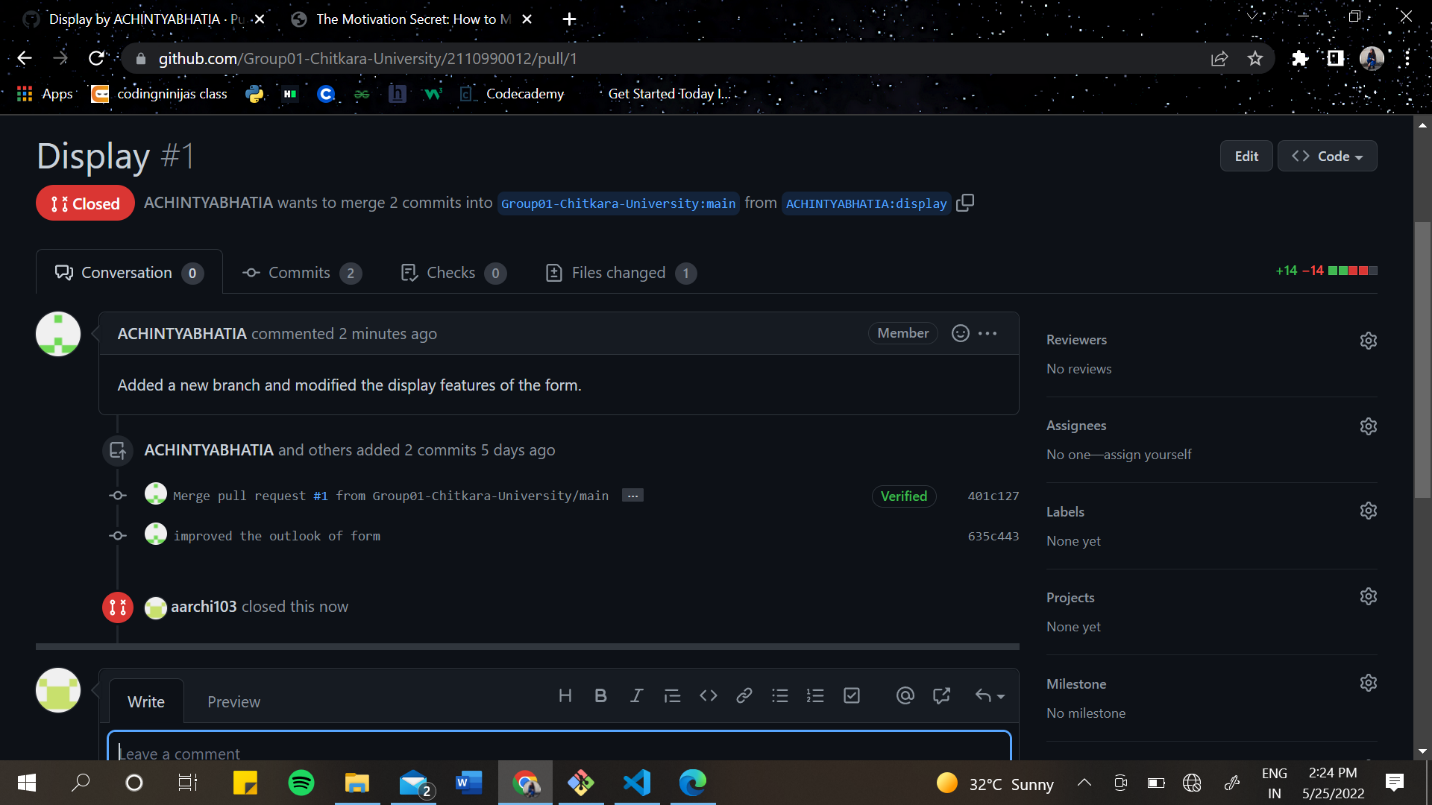
Here, the owner of the repository has closed the pull request without merging the commits that we made.

#CLOSING/MERGING PULL REQUESTS GENERATED BY TEAM MEMBERS

1. In the pull request menu of your repository their will be a notification. Open it.
2. You can either choose to close it without merging the commits or you can merge it.

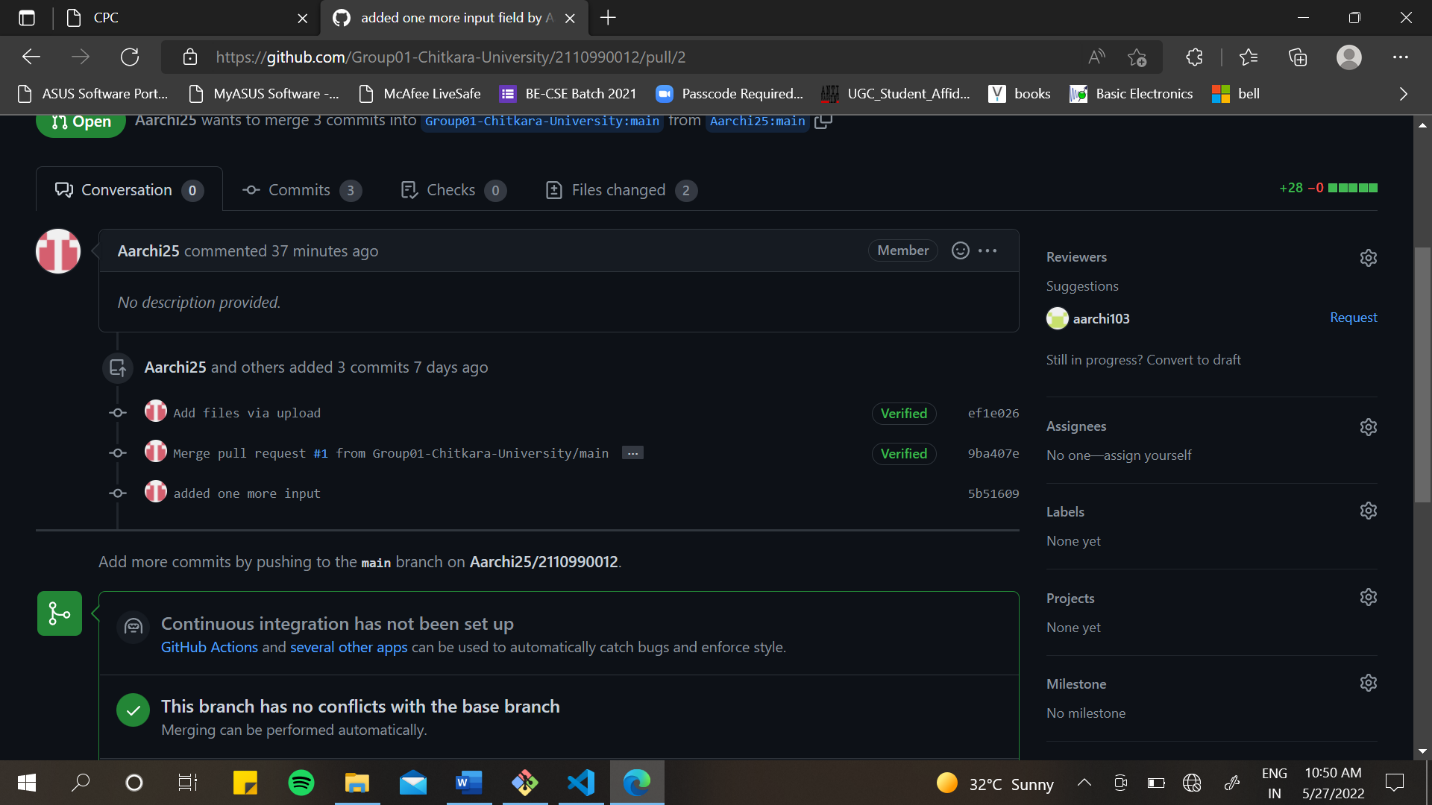


1. After closing it, this screen will appear.

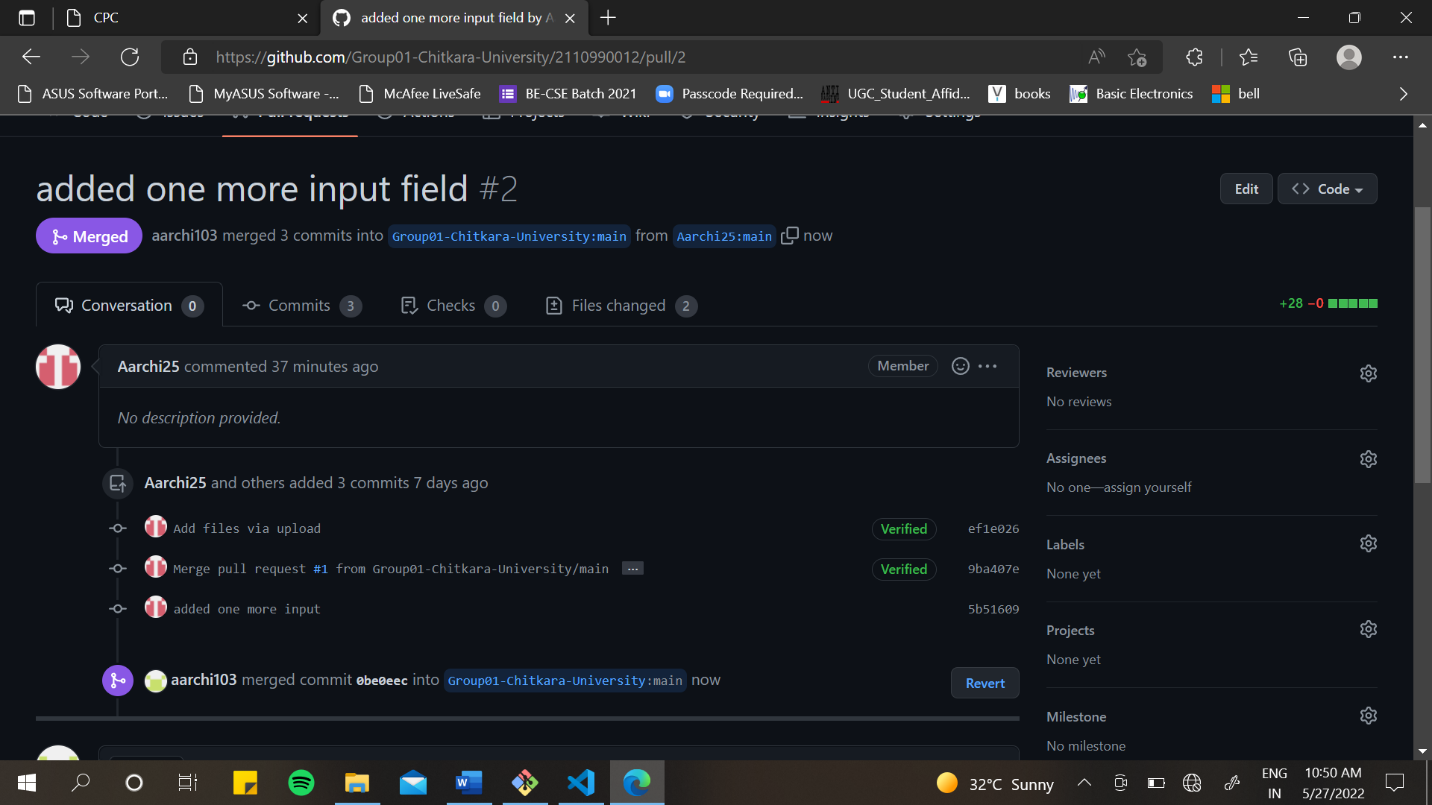


Similarly, close/merging the pull requests of other team members,

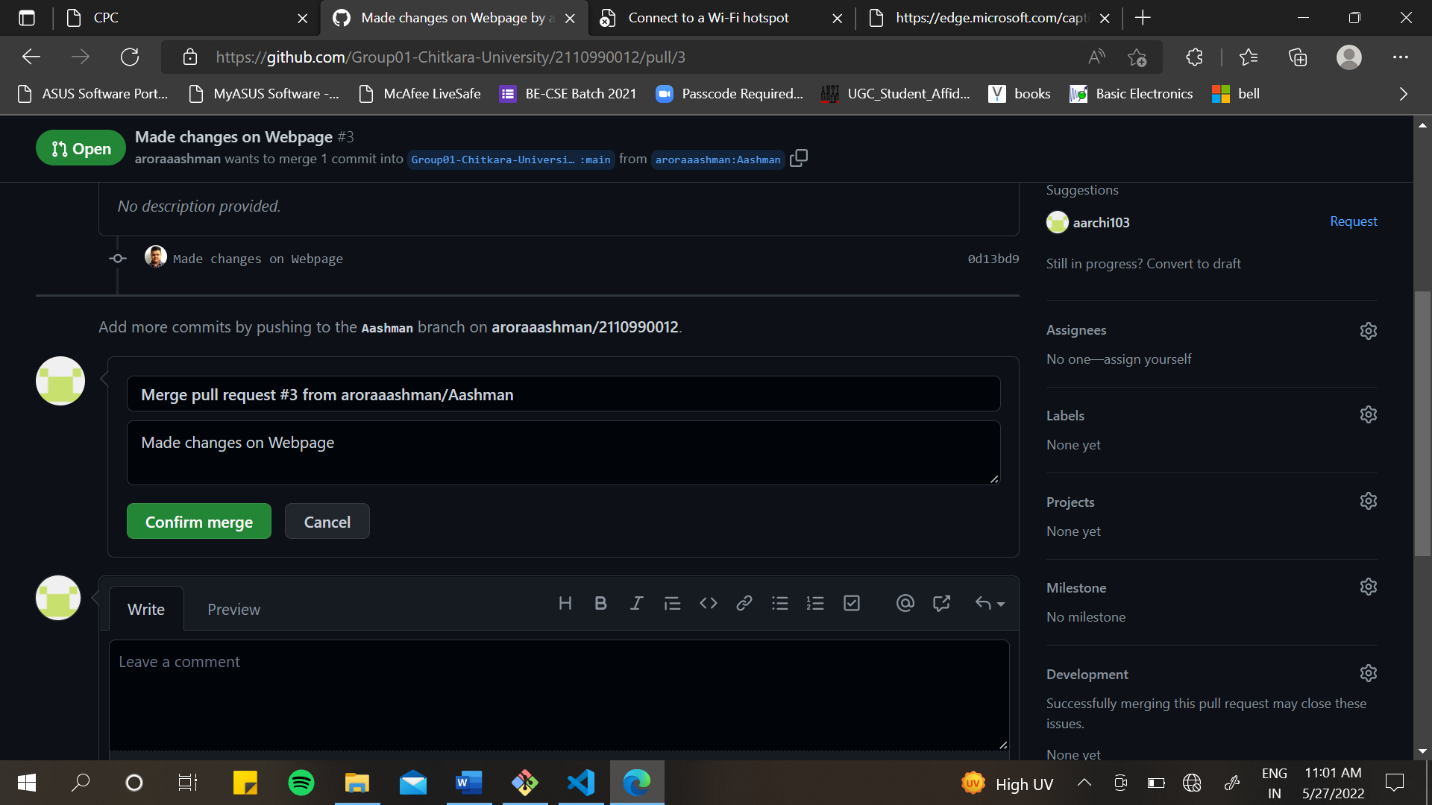
2nd team member:

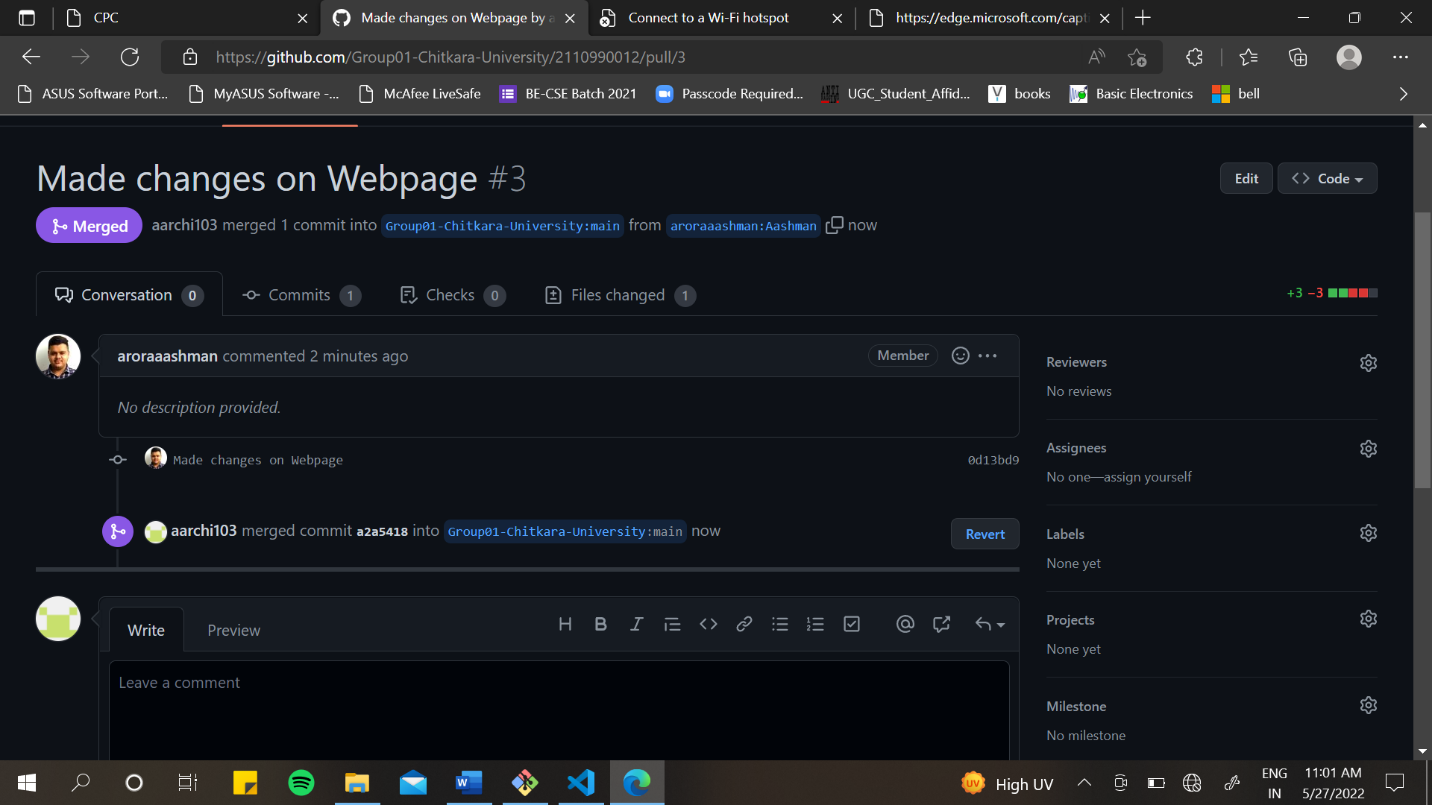


Here, I have chosen to merge the pull request which will introduce the changes made by them into my own repository.



3rd team member:





1. To view the pull requests history of your repository, click on the pull request button in the menu.
2. You will be shown by default the open requests screen.
3. Click on Closed option to see the history.

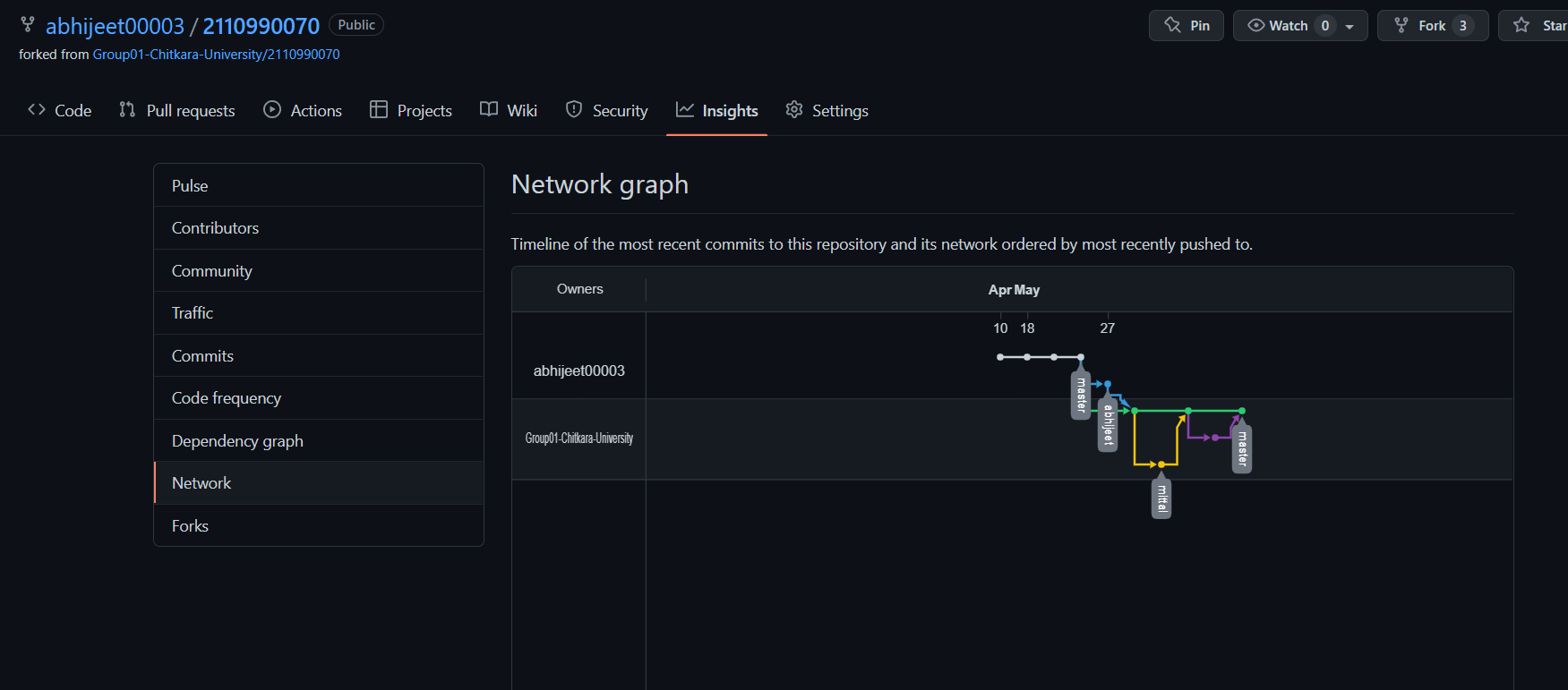


# **NETWORK GRAPHS**

To view the network graphs of your repository, follow the steps:

1. Go to the repository of which you want the graph/details.
2. Click on the ‘Insights’ option it the menu bar.
3. In the right menu list click on network.
4. You can see the network graph there.

It shows the timeline of the most recent commits to this repository and its network ordered by most recently pushed to.



The points in the network graph represents the commits. By hovering over the points, you can see the information about the commit such as author, checksum, message of commit.