**Assignment - 1**

**Student Name:** Preet **UID:** 22BDO10062

**Branch:** CSE (DevOps) **Section/Group**- 22BCD-1/B

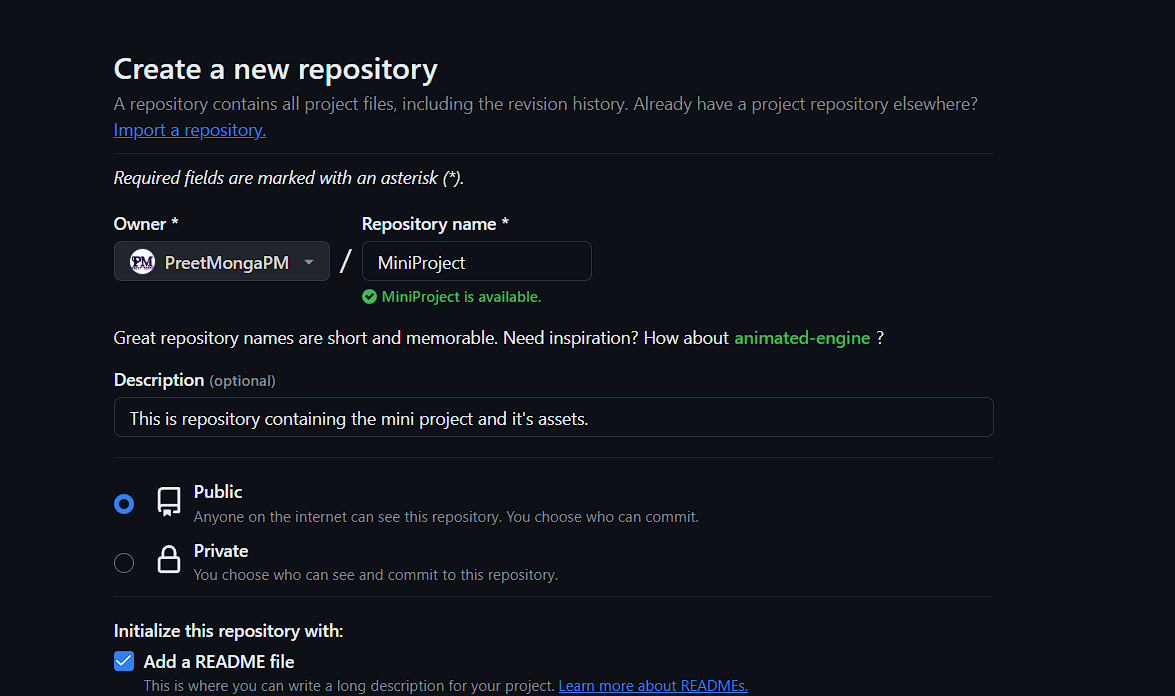
**Semester:** 4th **Date of Submission:** 26/02/2024

**Subject Name**: Git and hub **Subject Code:** 22CSH-293

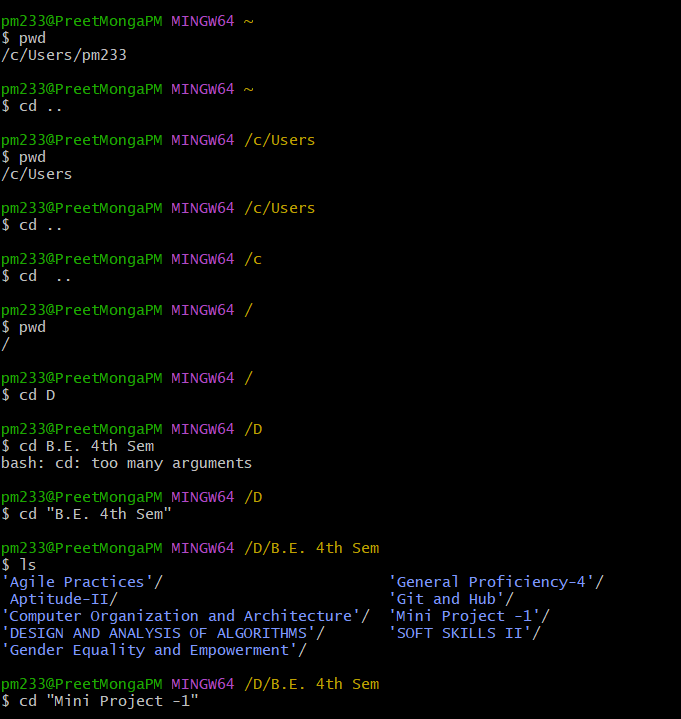
**Task 1: Create a project on GitHub and add collaborators.**

**Steps are following:**

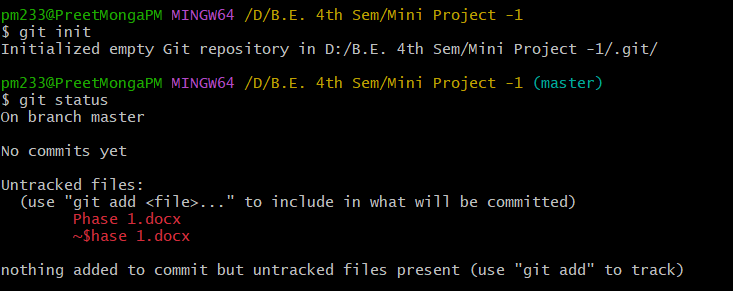
1. Create repository over GitHub.



1. Upload the files of Mini-Project over to this repository using the Git Bash.
2. First go to the directory of your project on your local machine.



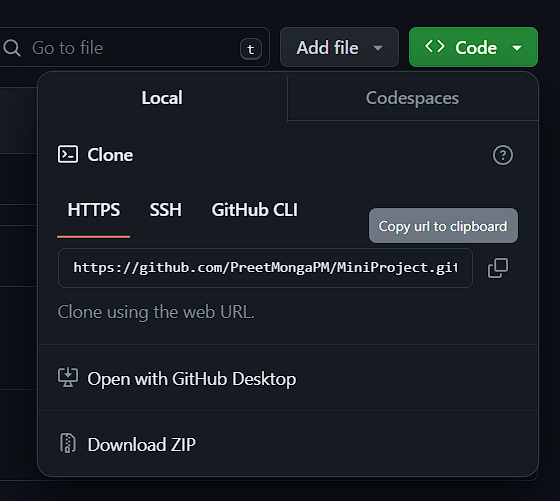
1. Initialize git in that and check status of that file.

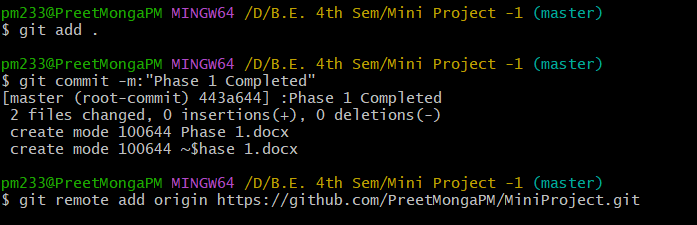


1. Now, add and commit those files, and push it over to your remote repository.

As the normal push wasn’t working due to conflict of files. I used Force push to overwrite the remote repository with the content of local repository.

Also, add origin of that remote repository using following commands.

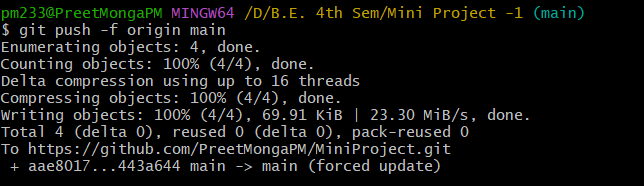




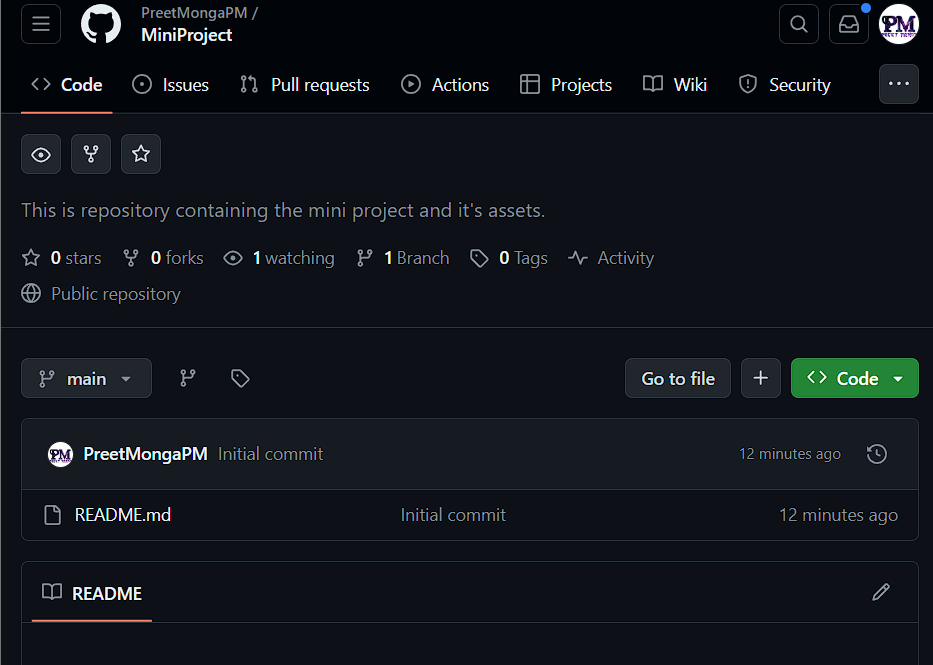
Rename the branch from master to main to push that.



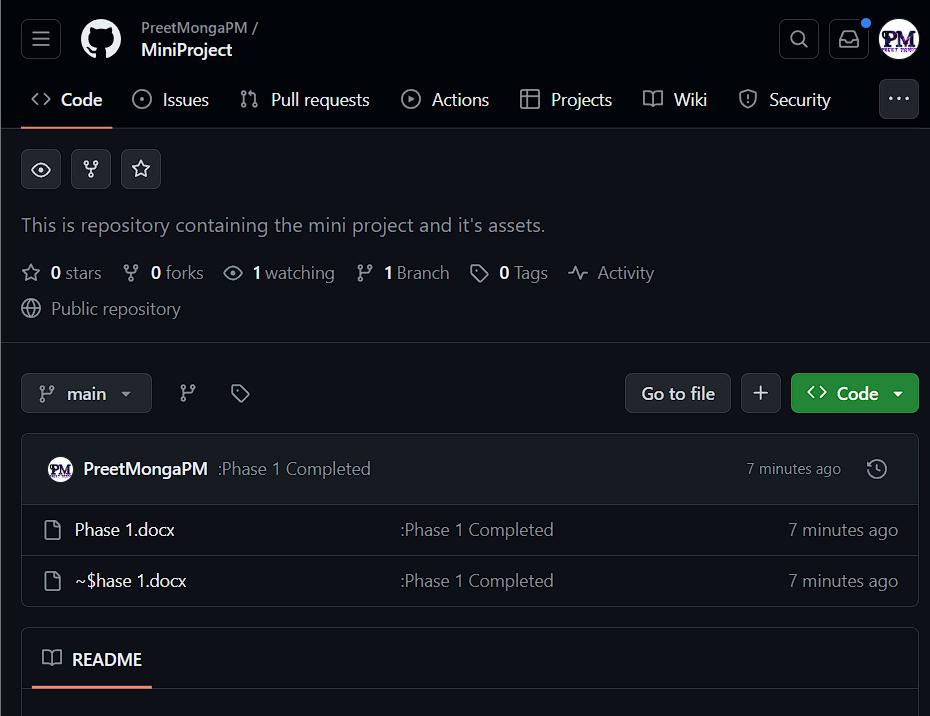
Force Push:



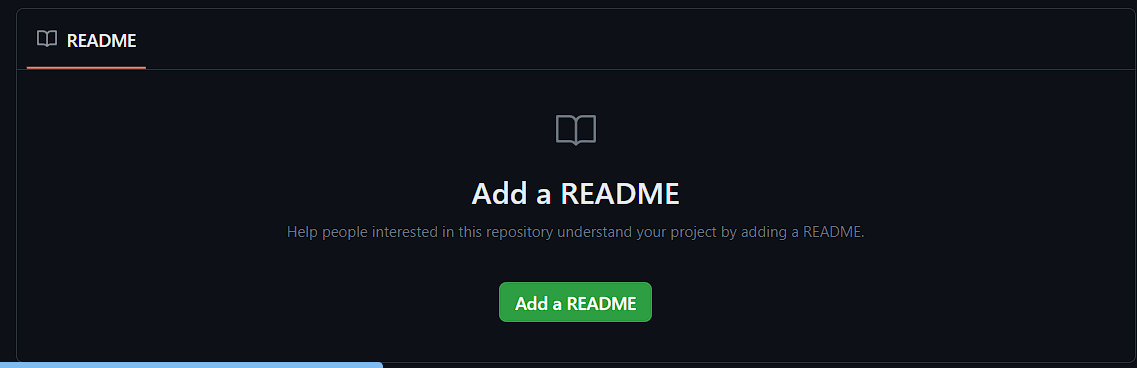
Before Push:

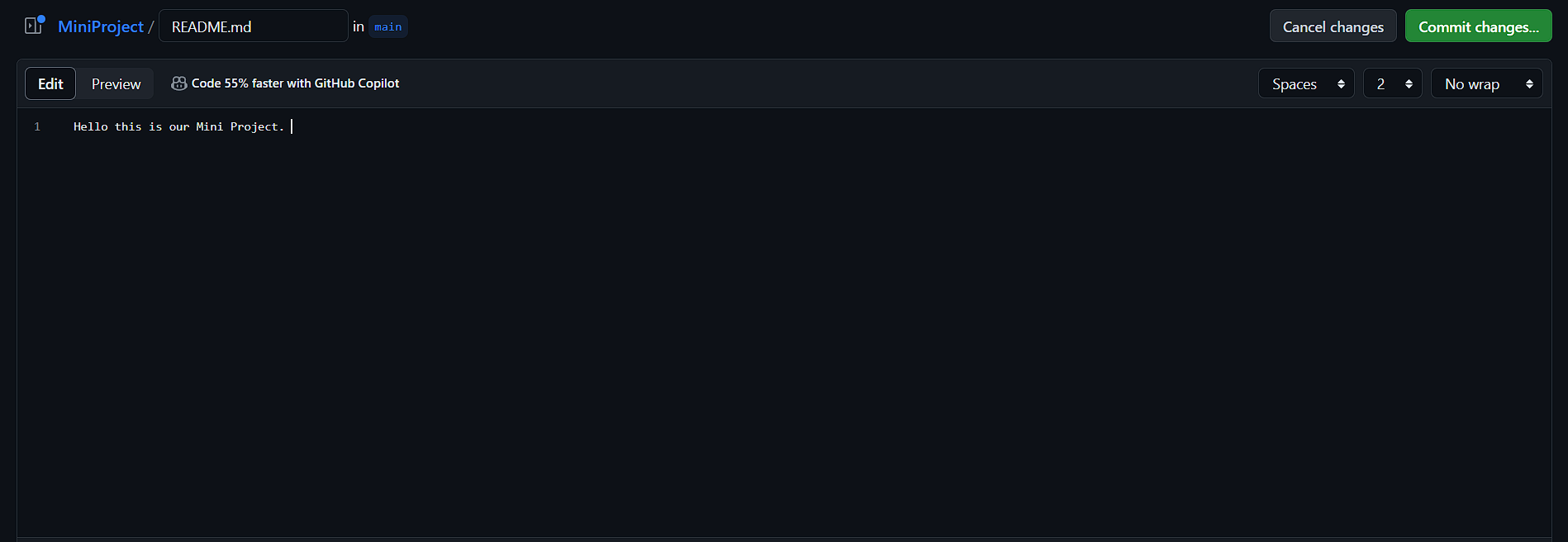


After Push:



1. As the README.md was overwitten by local let’s add it again.



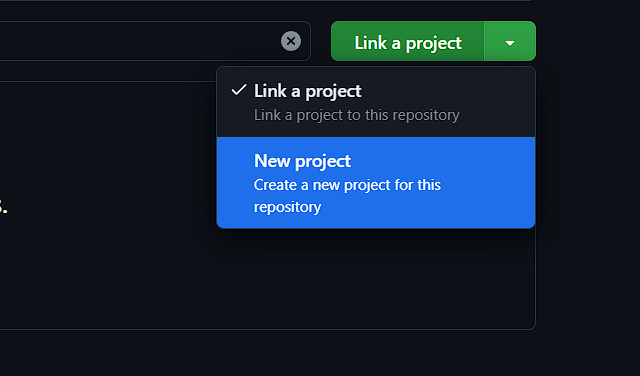


1. Let’s head over to Projects section to create project for this repository.

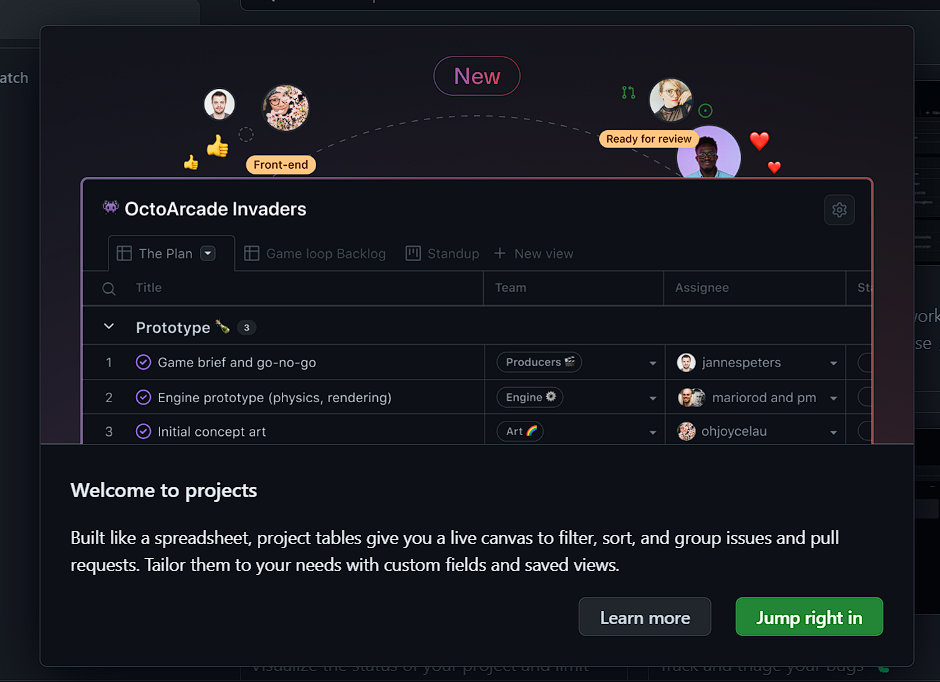


1. There are two options whether you wanna create a new project or link a existing project.

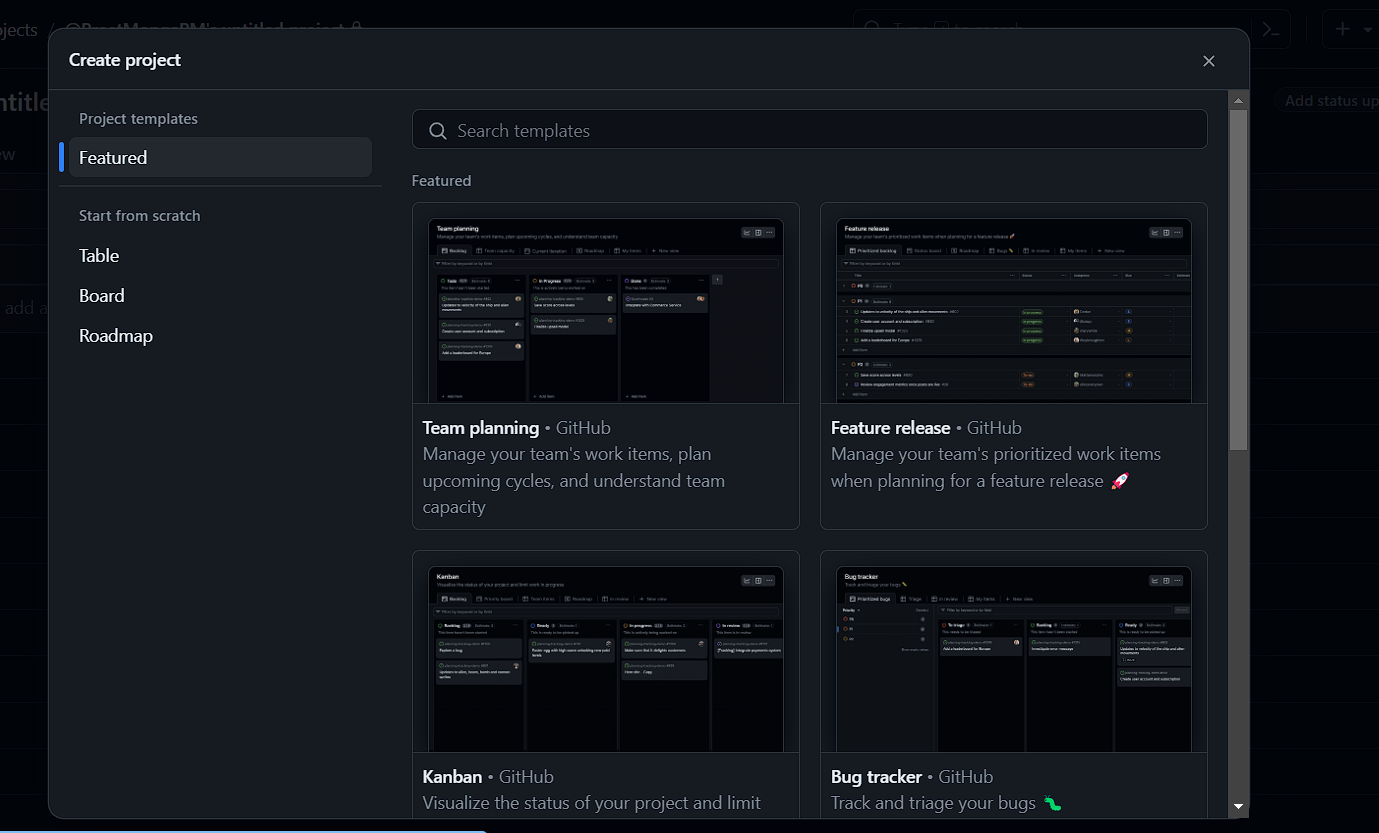
Click on New Project.



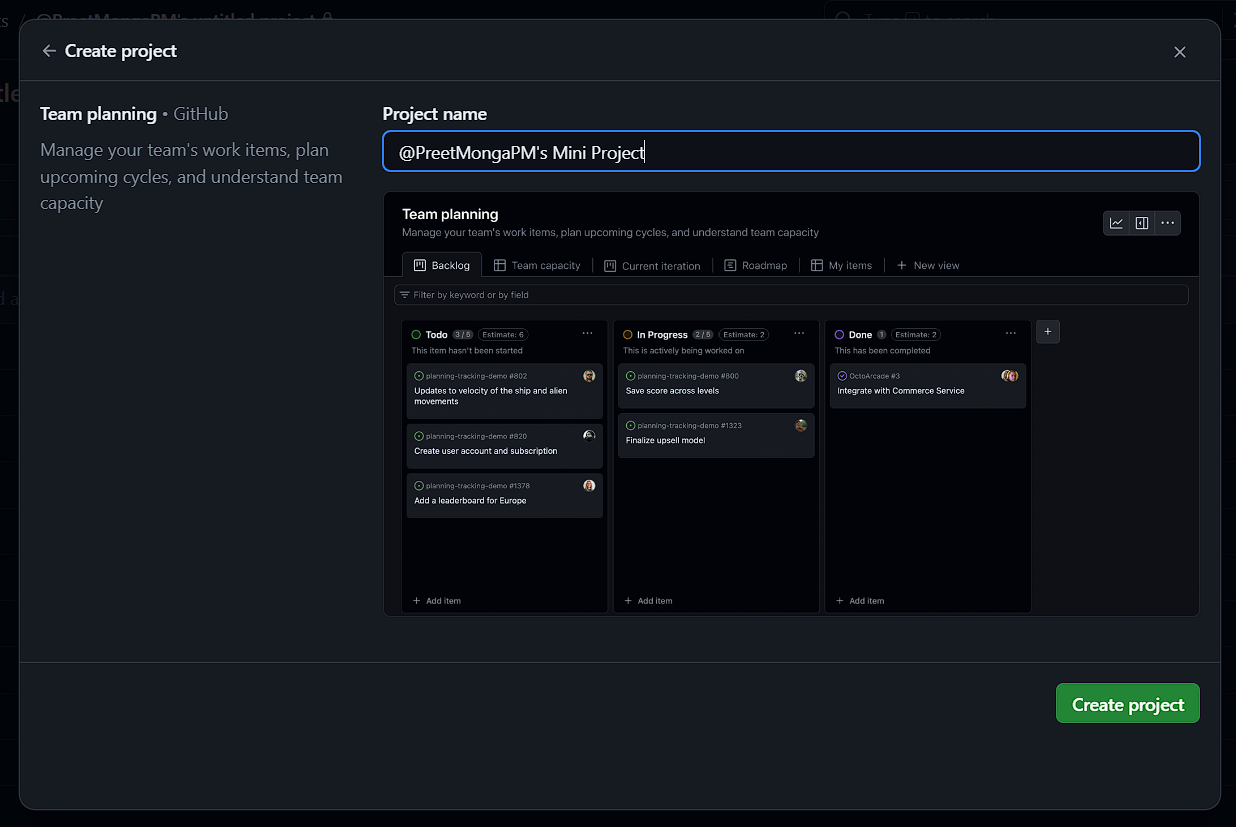
1. If you are using projects for the first time this window will popup.



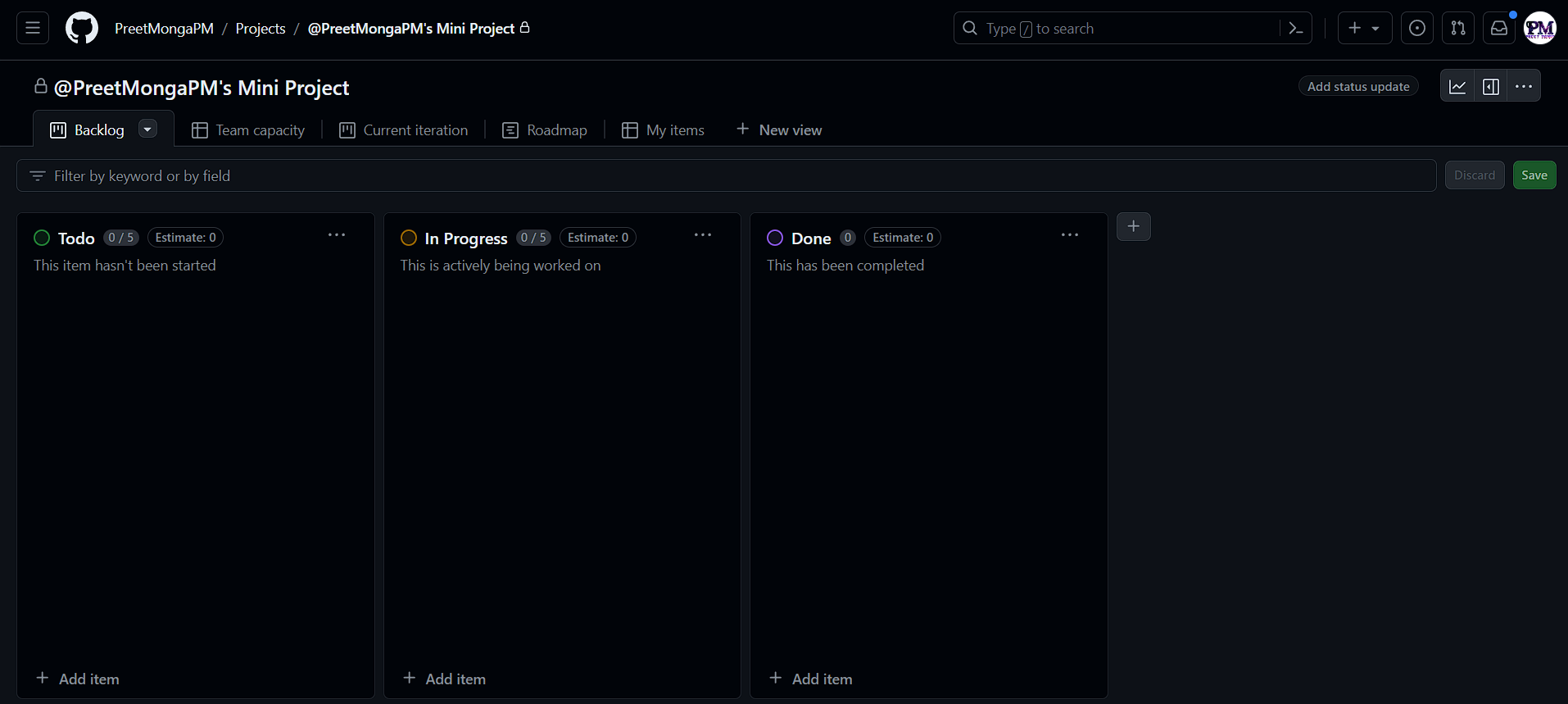
1. There are various templates which we can use, I am selecting the team planning.



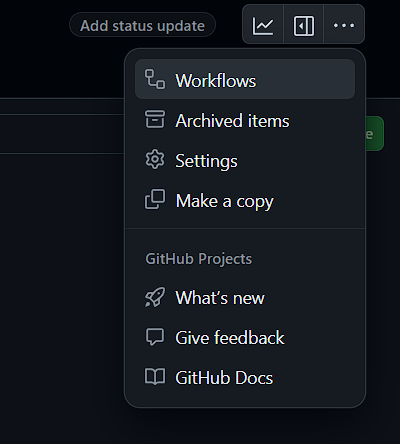
1. Write Project’s Name and click on “Create Project”.



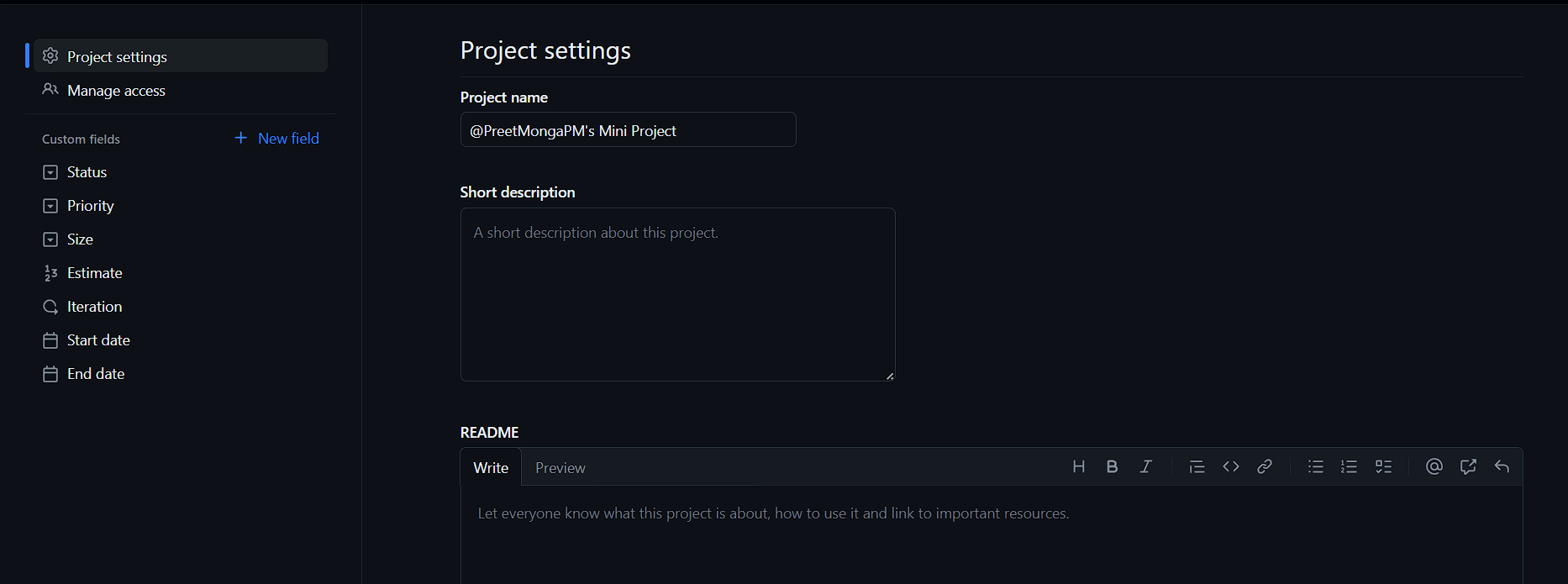
1. Our project is created.



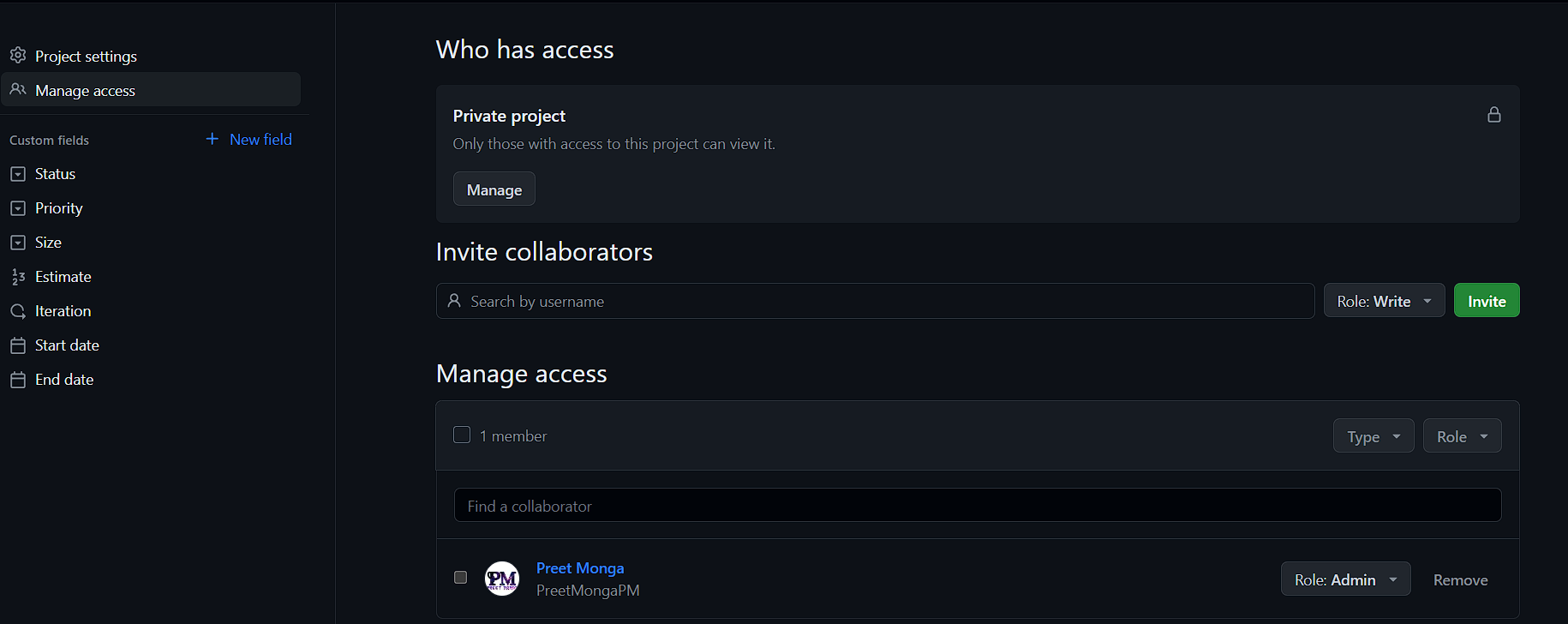
1. Go to three horizontal dots and go to settings.

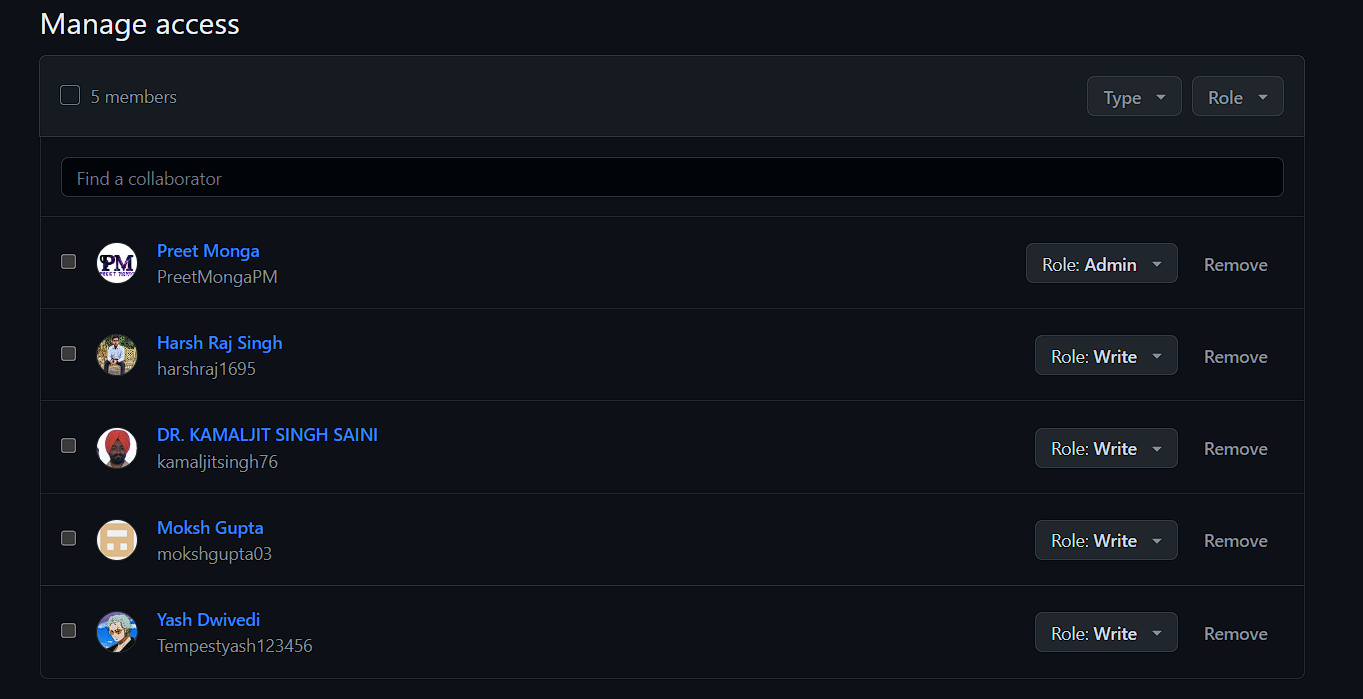


1. Here, we can write the short description of our project or can change the name of project.



1. Go to manage access and invite collaborators by using :   
   @user\_name

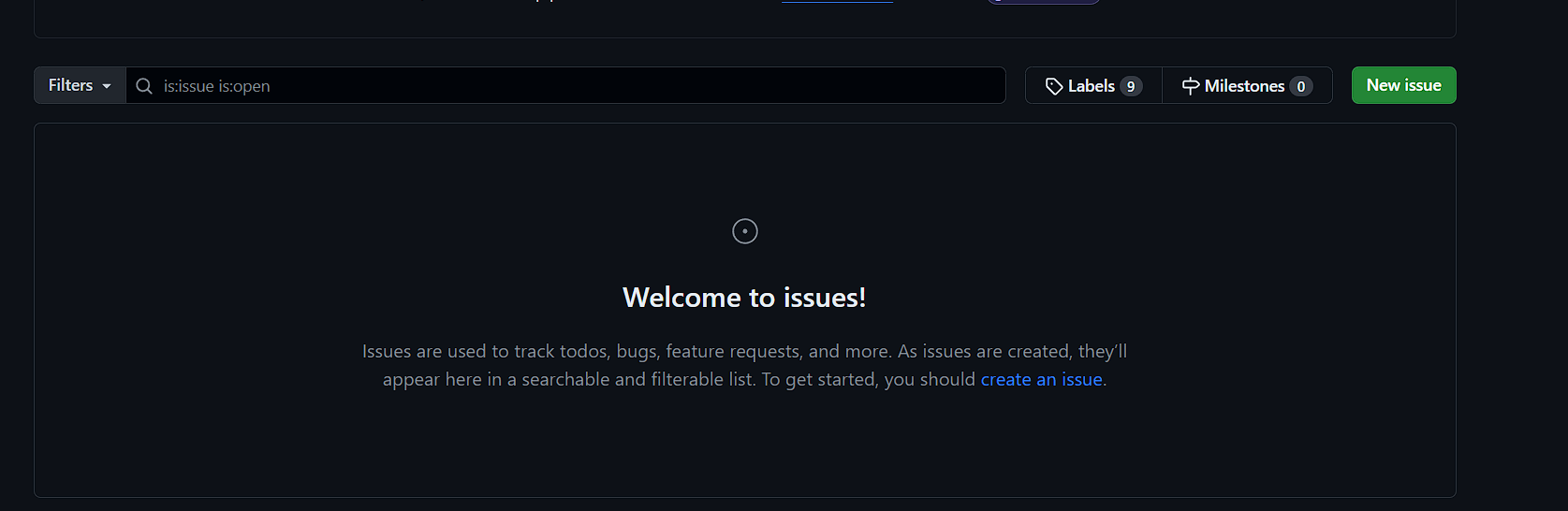




1. Let’s create a issue by heading over to issue tab.



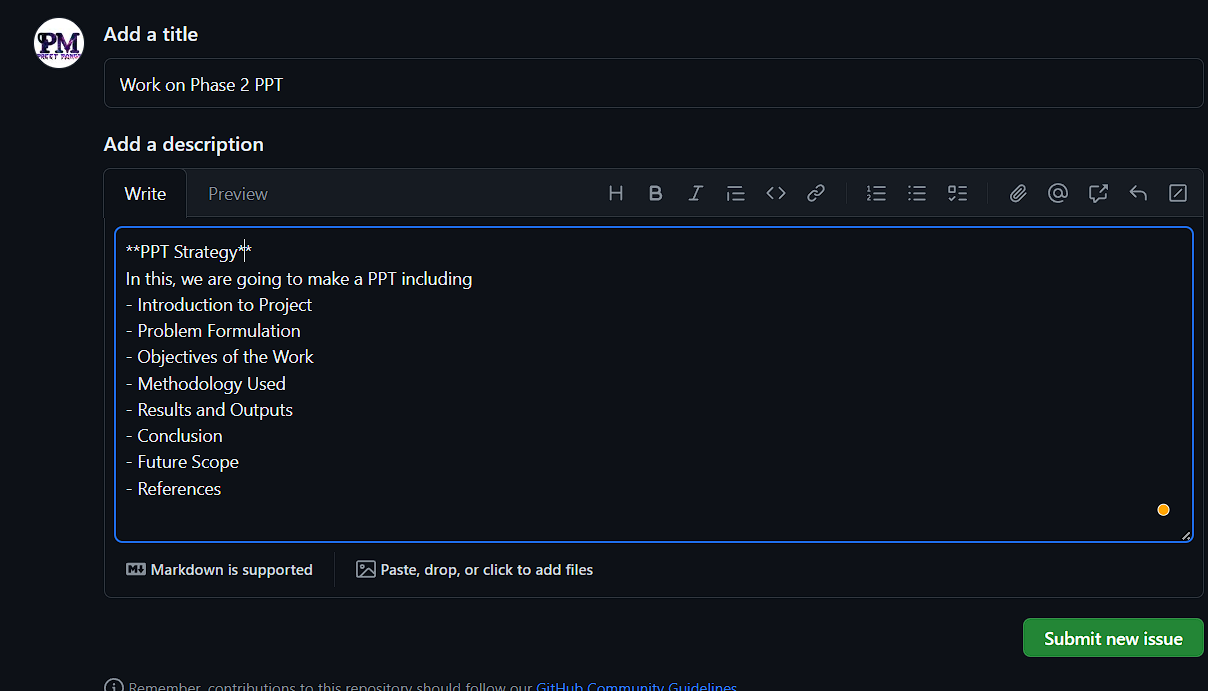
1. Click on new issue.



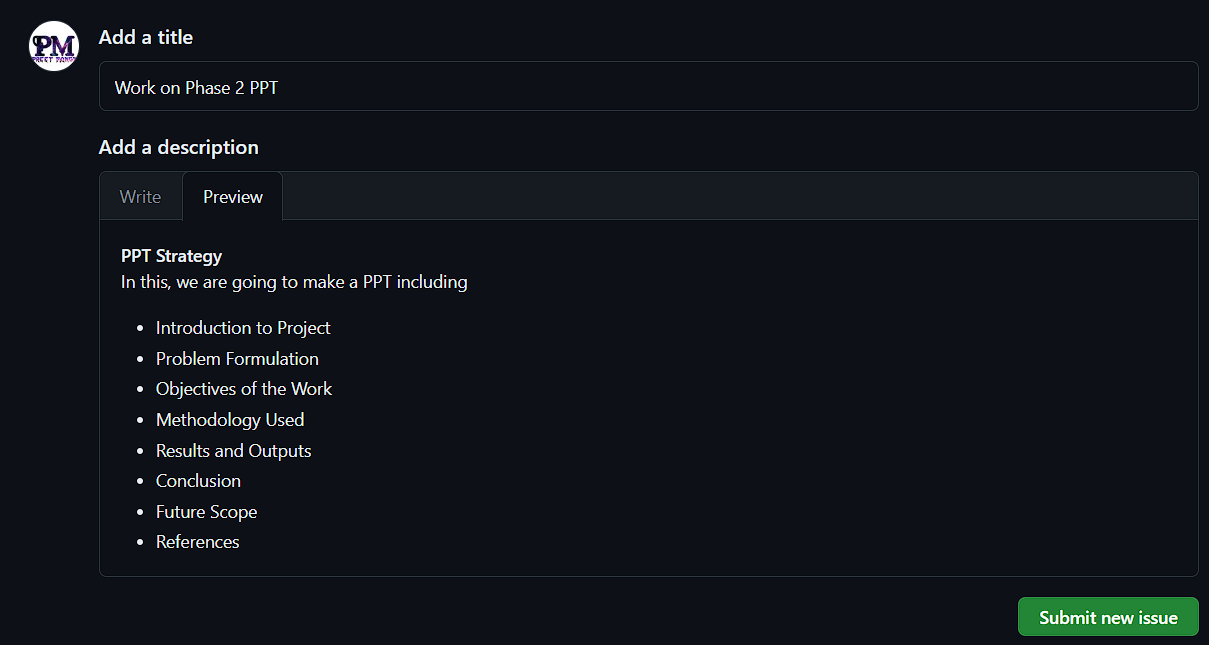
1. Here, give title and description using text or markdown as per your requirement.

\*\* \*\* - Bold letters

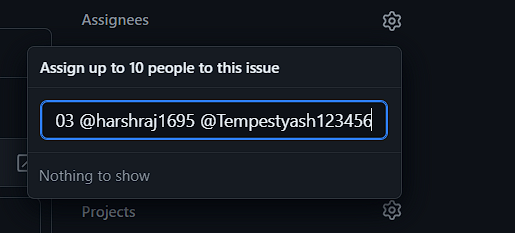
* For ordered list

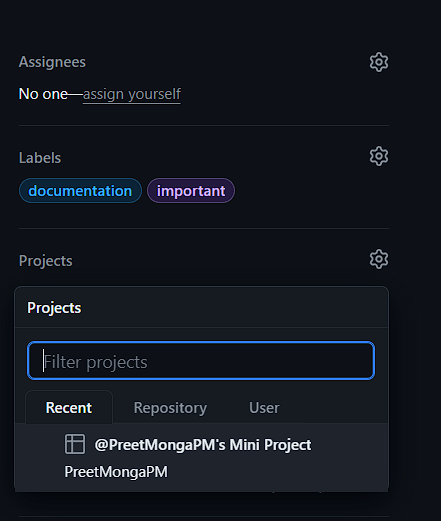
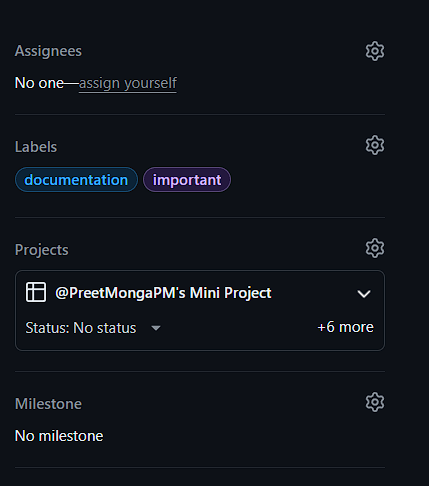


1. We can see preview also.

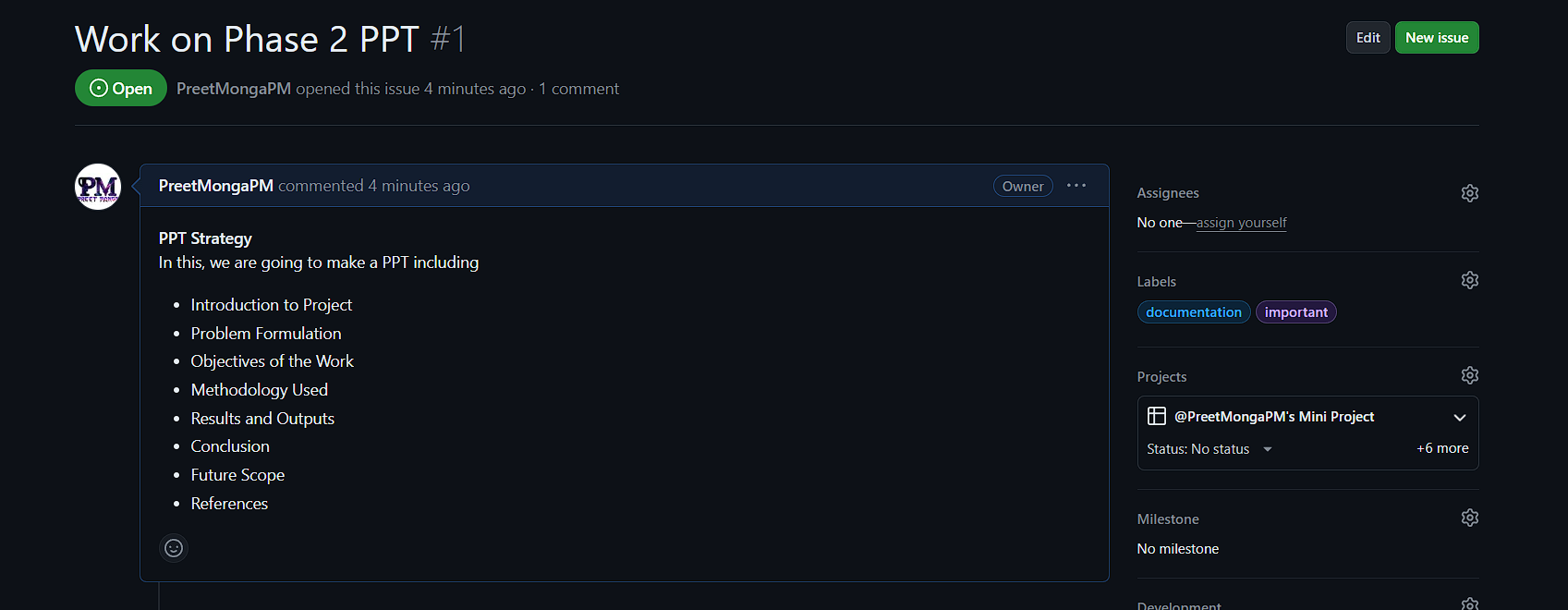


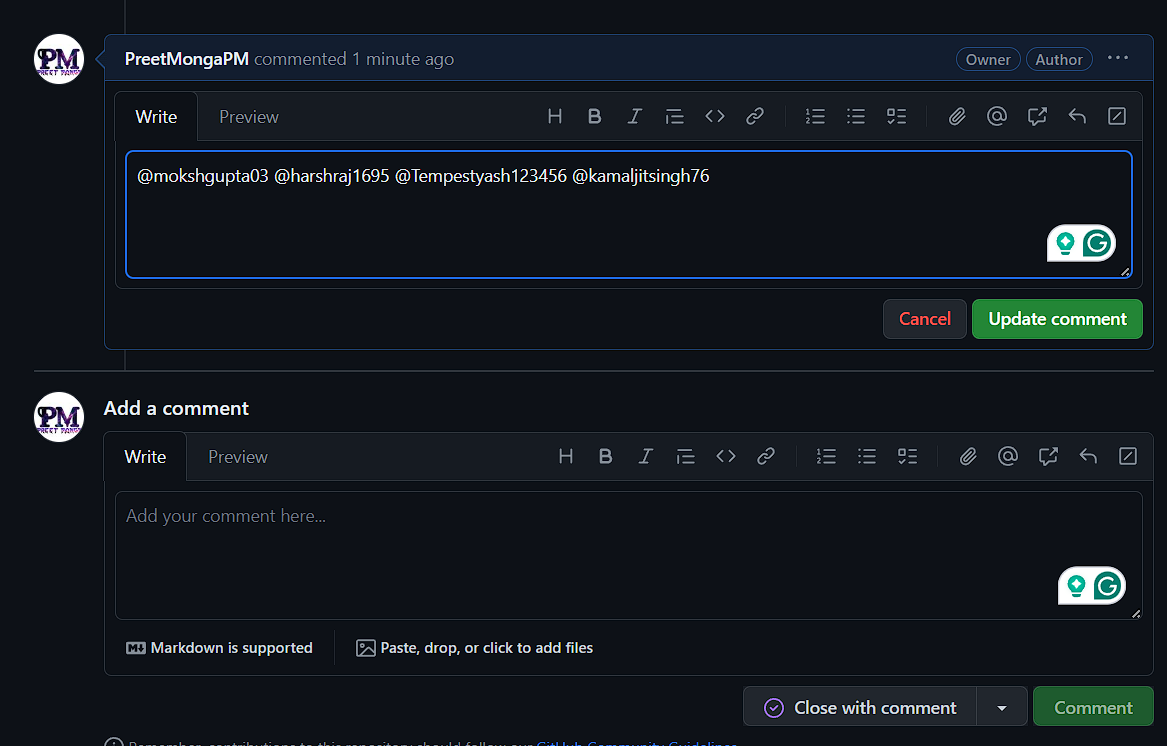
1. We can add assignees, labels, milestones and projects.



1. Also, we can assign the issue using the comment section as follows.



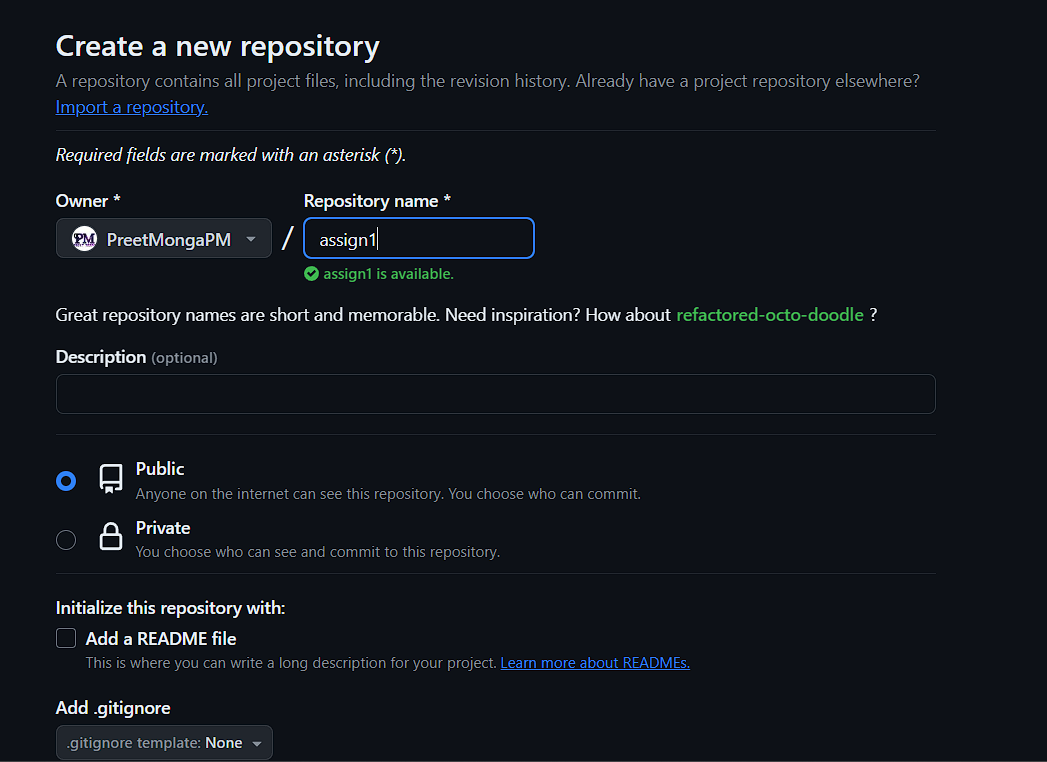


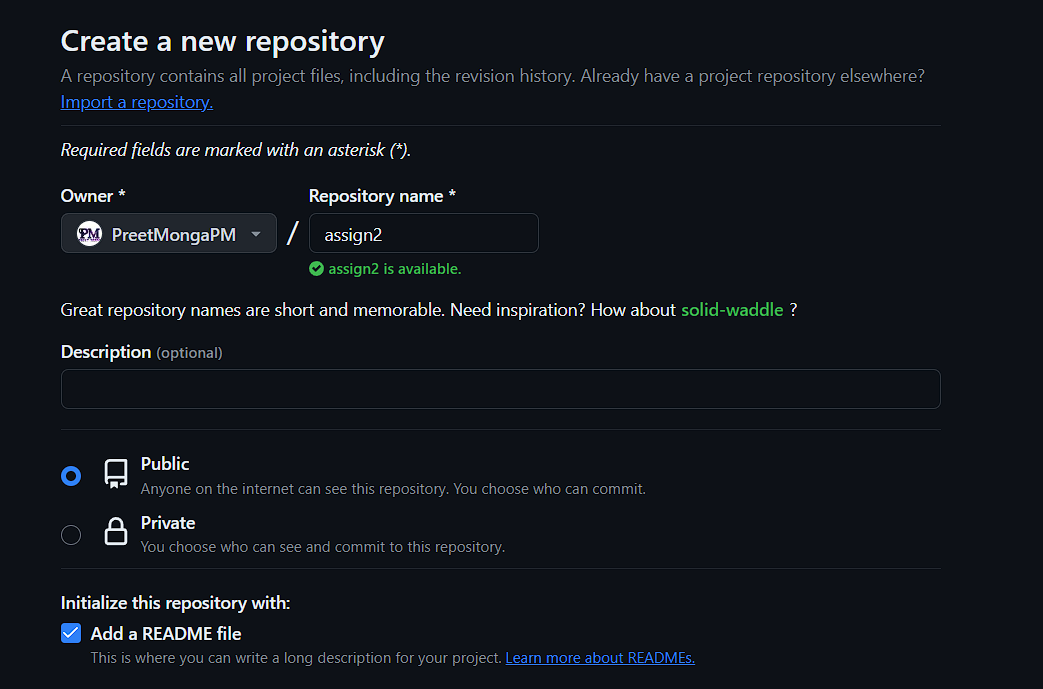
1. So our repository with project and issue is successfully created.

**TASK 2: Create two remote repositories, then make two files with same name but different content then merge them on local machine and push the merged version over to both remote repositories.**

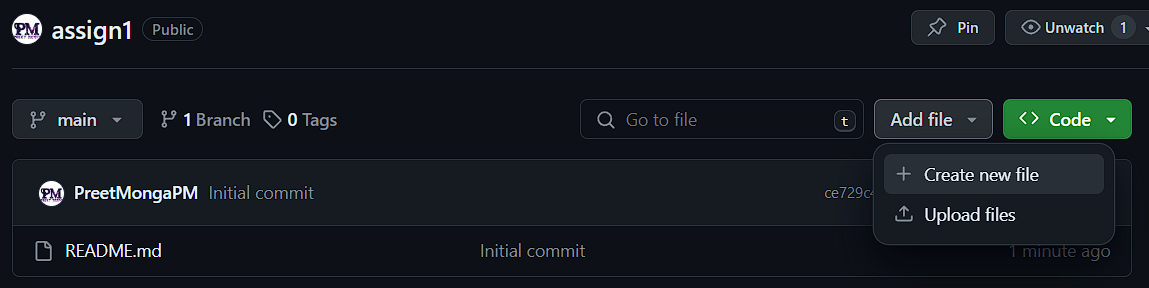
Steps:

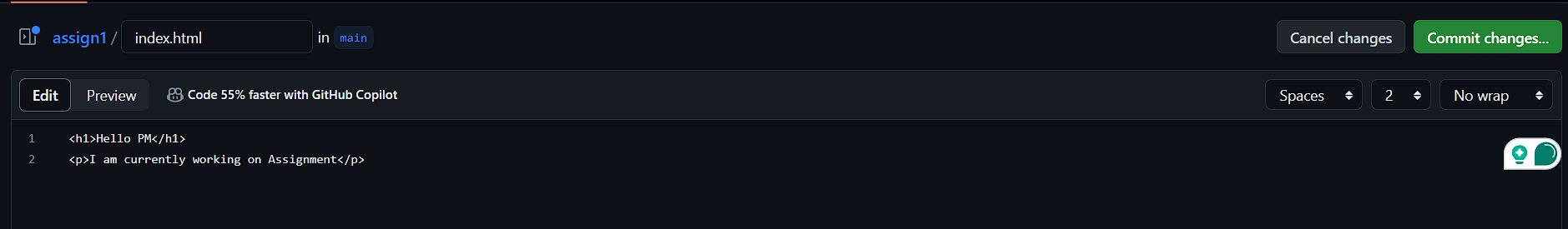
1. Go to GitHub and create 2 repositories, let’s take assign1 and assign2.

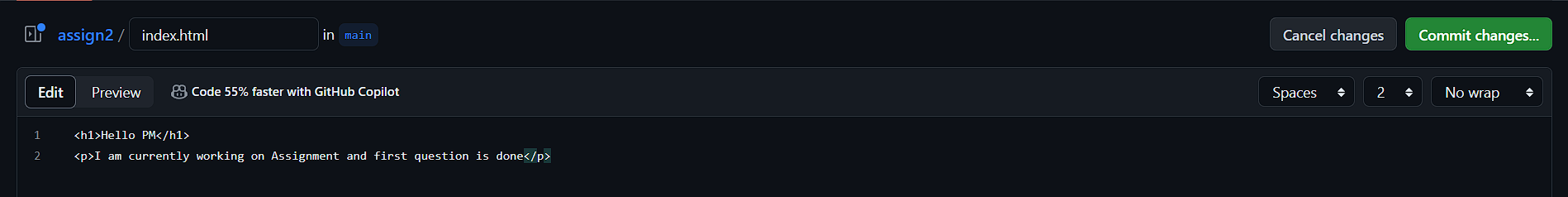
****

****

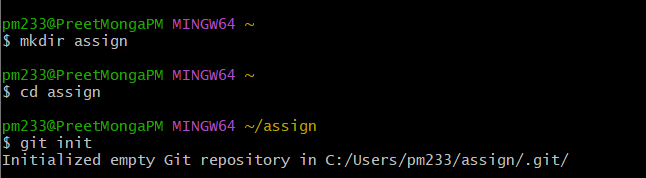
1. Create file in each repo, index.html with different content.

****

****

****

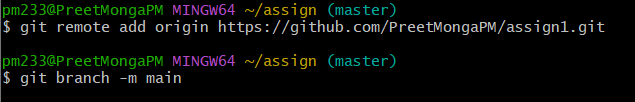
1. Open Git Bash and create a local repository using mkdir and cd to change directrory, then initialize the git in it.



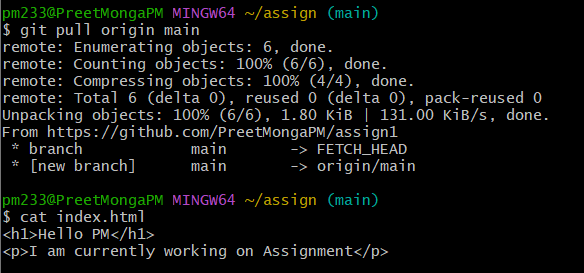
1. Then add remote of “assign1” remote repository.

(also changed the branch name using:

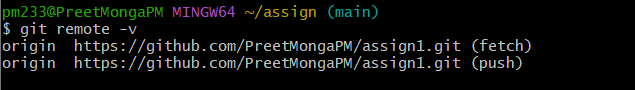
*branch -m)*



1. Pull the changes of assign1 and display file.

****

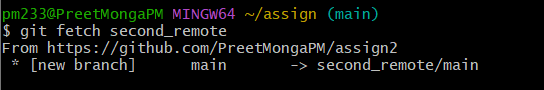
1. Also, can check the current remote.

****

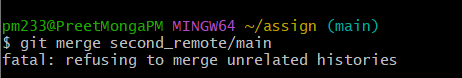
1. Add second remote of assign2 repository.

****

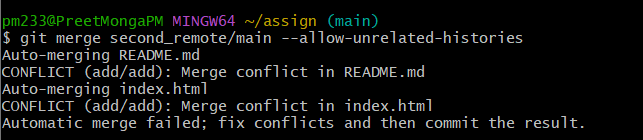
1. Fetch the changes of assign2 repository.

****

1. If we merge simply both the index.html it will show the following error.

****

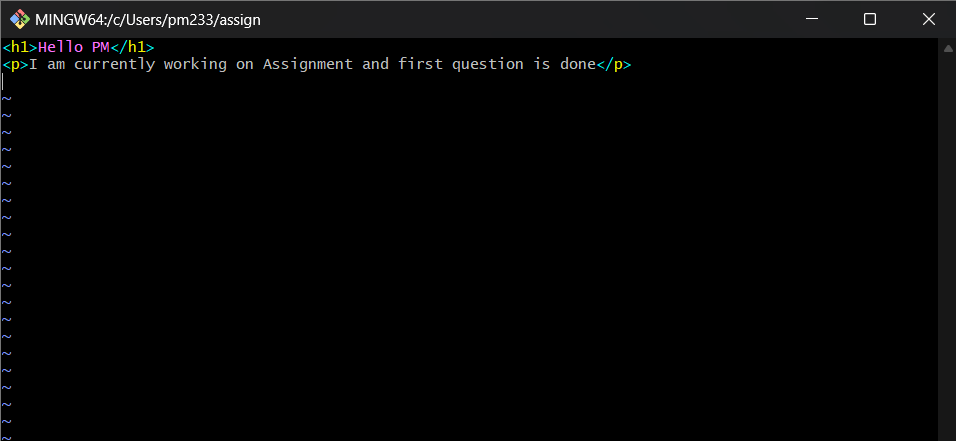
1. Use the following command to fix this error because both repo don’t have a common ancestor.

****

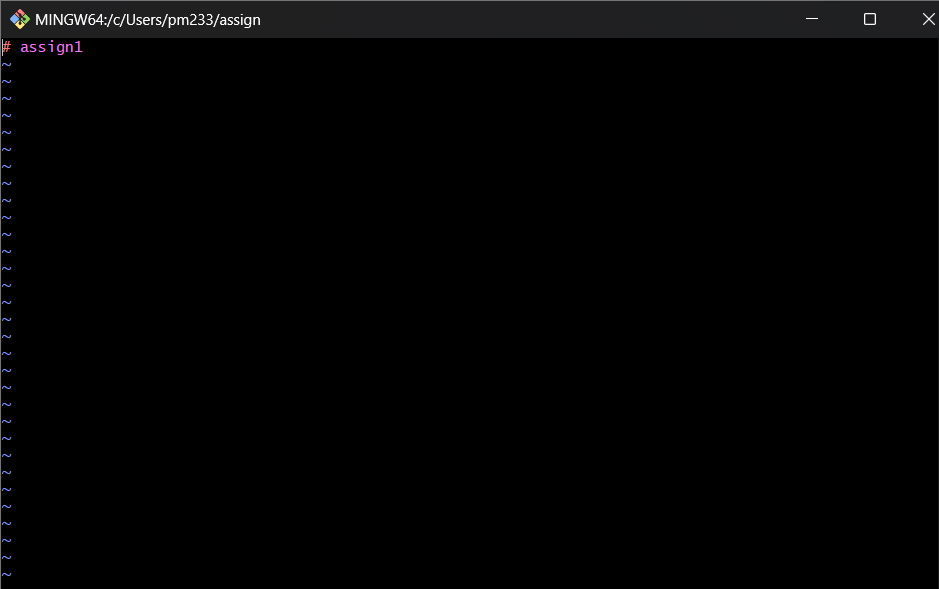
1. Now, we have to resolve the conflicts manually using vim editor,
2. Using vim:

****

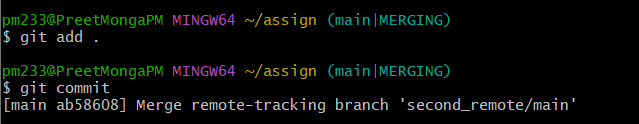
1. Delete first both remote names and keep the content you want to keep.
2. Index.html



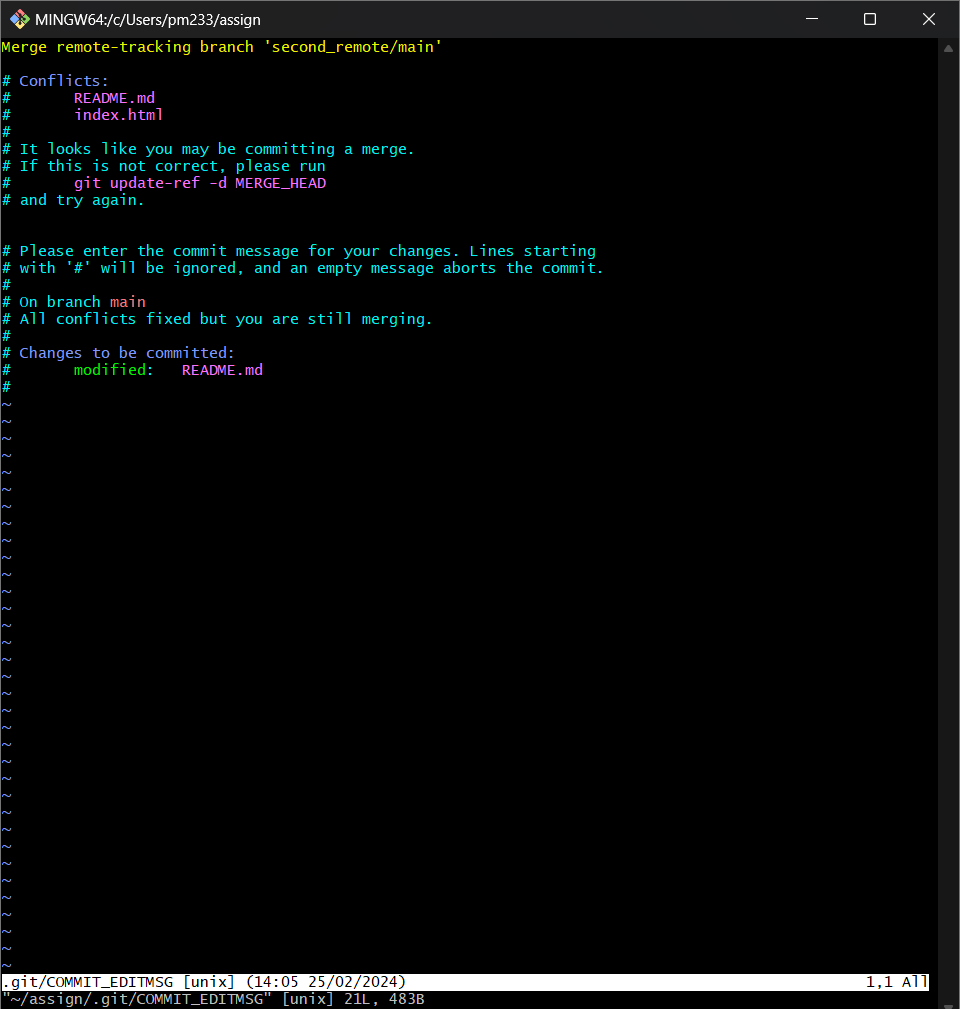
1. README.md

****

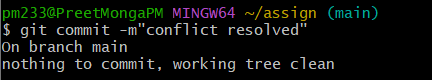
1. Now, add the changes and commit them.

****

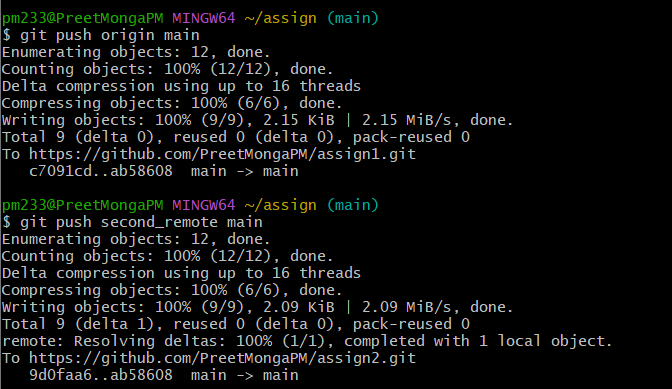
As, we committed without message it will show following screen in vim editor.

****

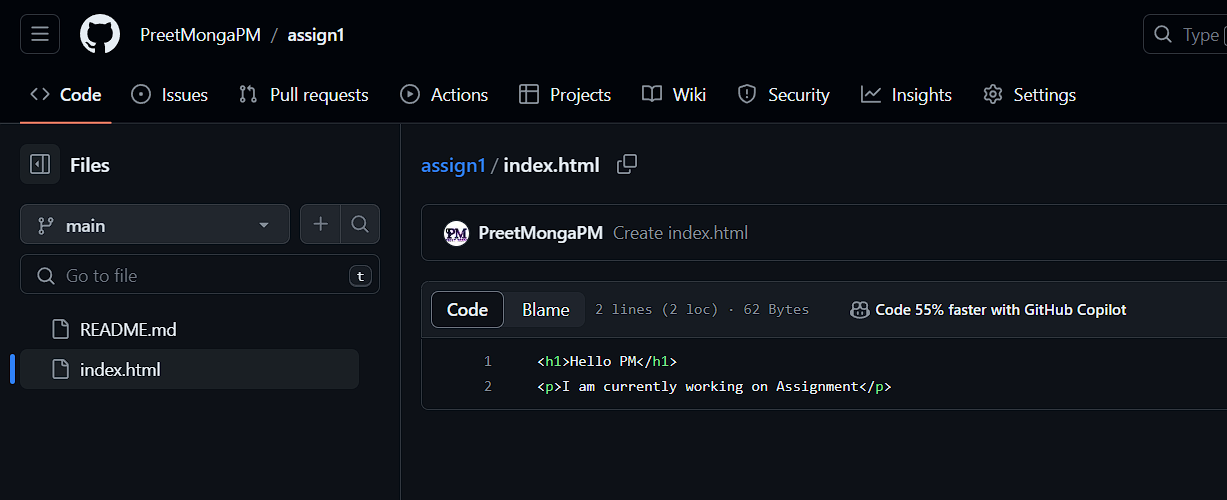
1. Now, close the vim editor using :wq and commit using -m together.

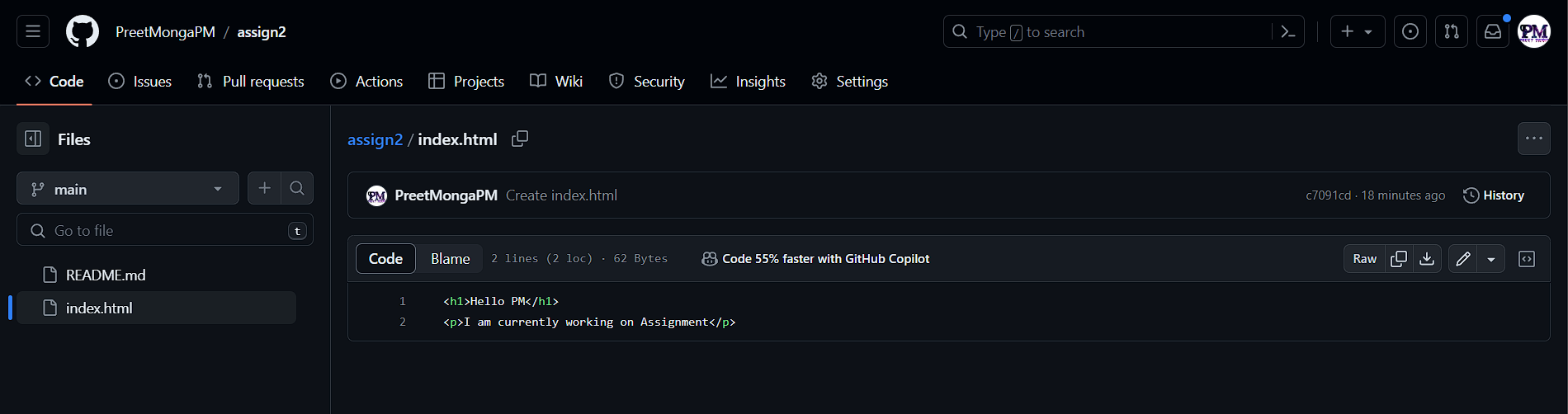


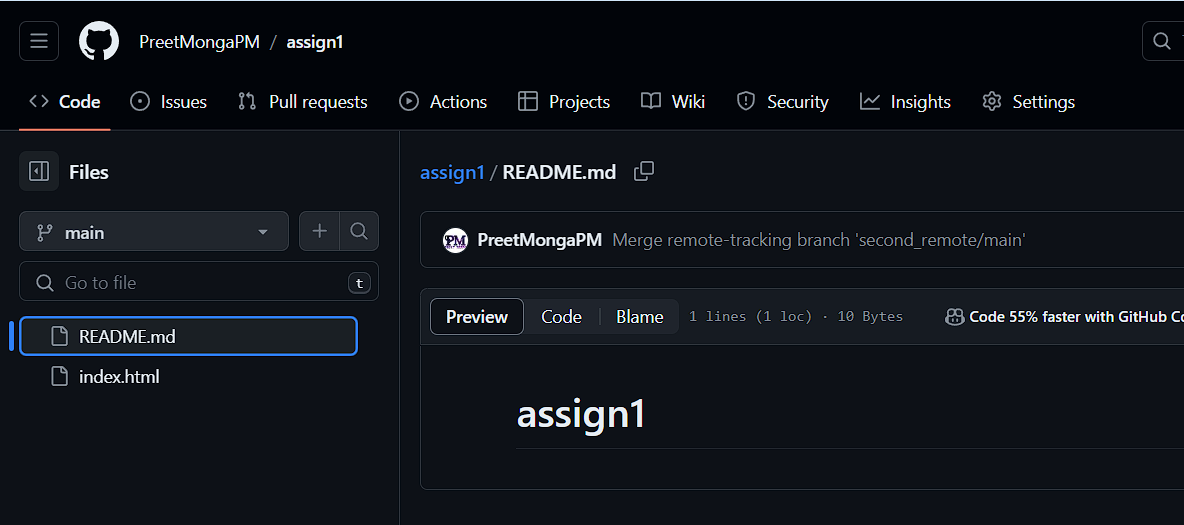
1. Push the changes to both repositories.

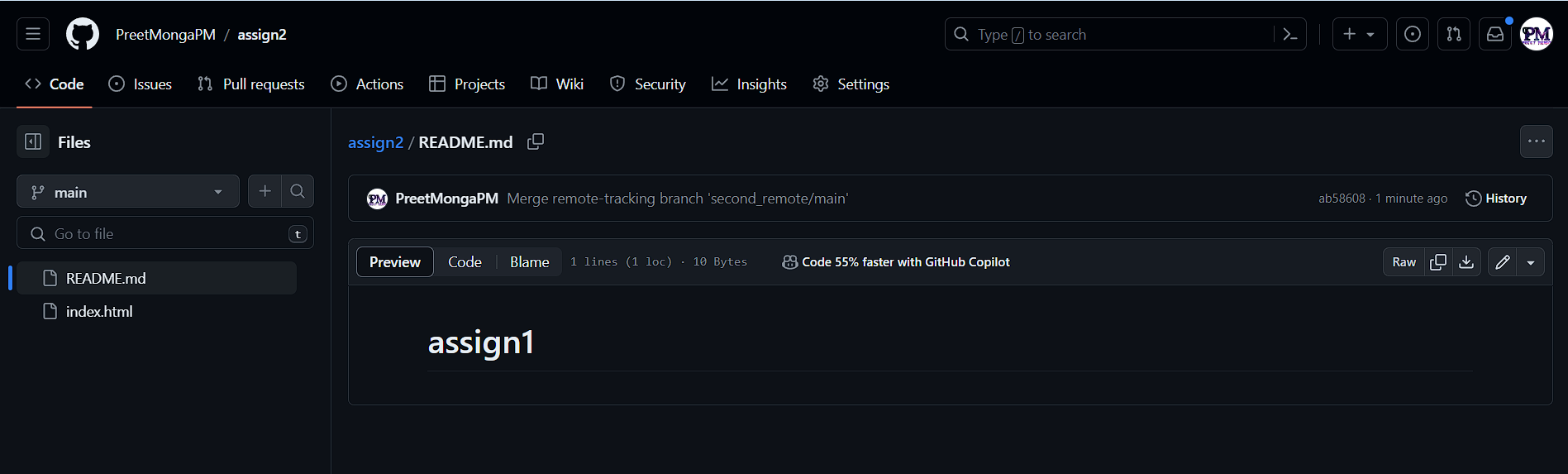


1. Go to GitHub and refresh the page, we can see both are same now.



****



****

1. Task Done.