

Yashraj Patil
2020BTECS00069

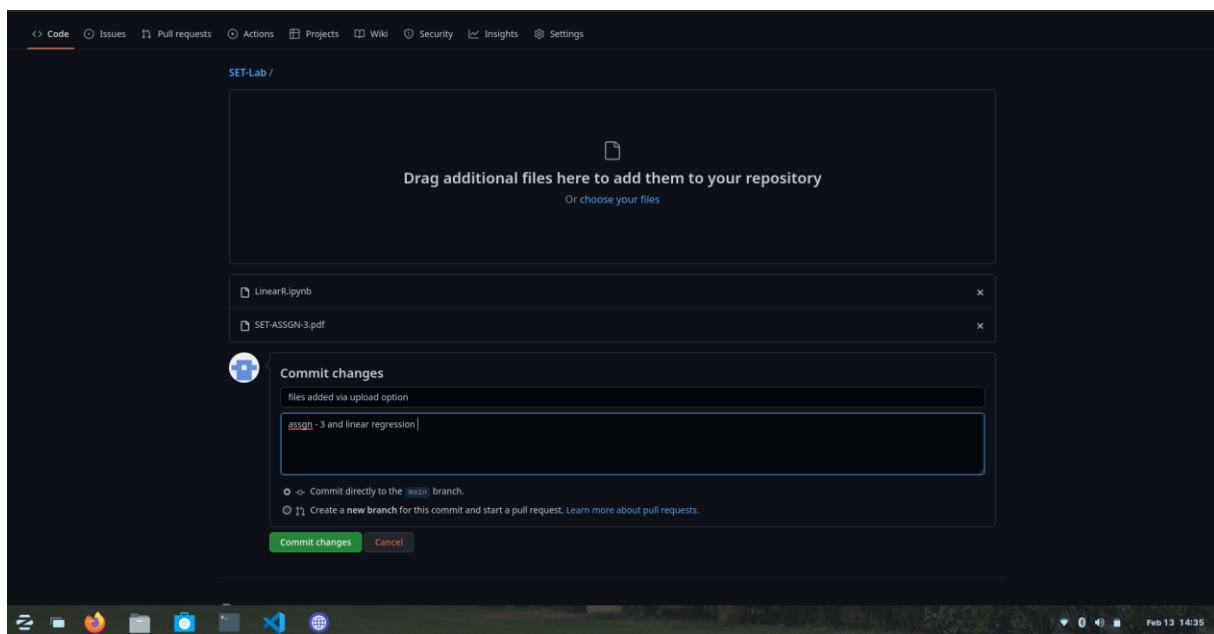
Name: Yashraj Patil
PRN: 2020BTECS00069
ASSIGNMENT - 4
SET LAB
T7 BATCH

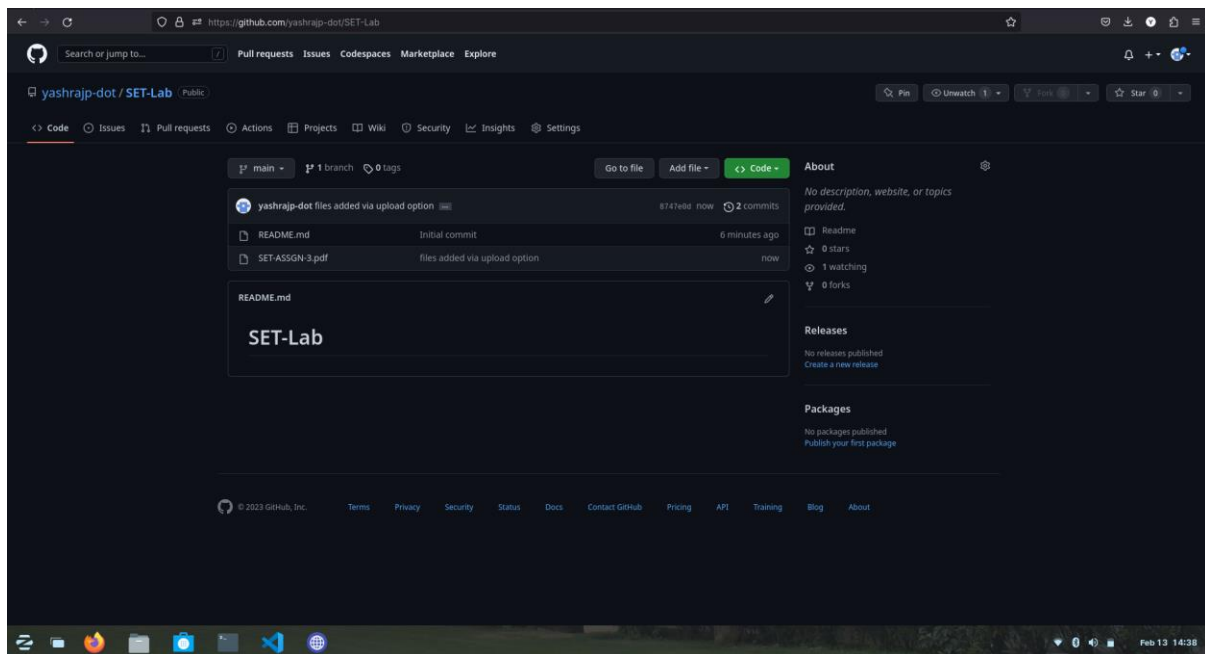
Q 1. Create a repository on GitHub named SET Lab and add files into it (you can add implementation files of previous assignment) perform below operations on it. (Add screenshot as an answer to every question)

1. Perform commit on added files.

To perform a commit on the added files, follow these steps:

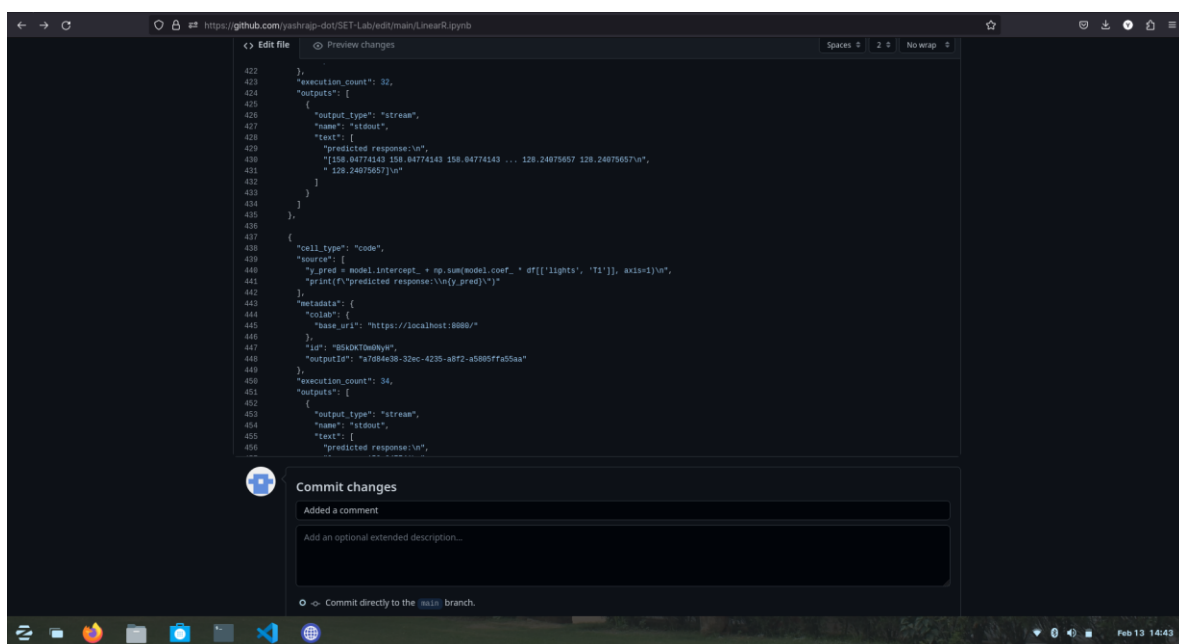
- Go to the main page of your repository on GitHub.
- Click on the file you want to commit changes to.
- Click on the "Edit" button to make changes to the file.
- Enter a commit message to describe the changes you are making.
- Click on the "Commit changes" button.

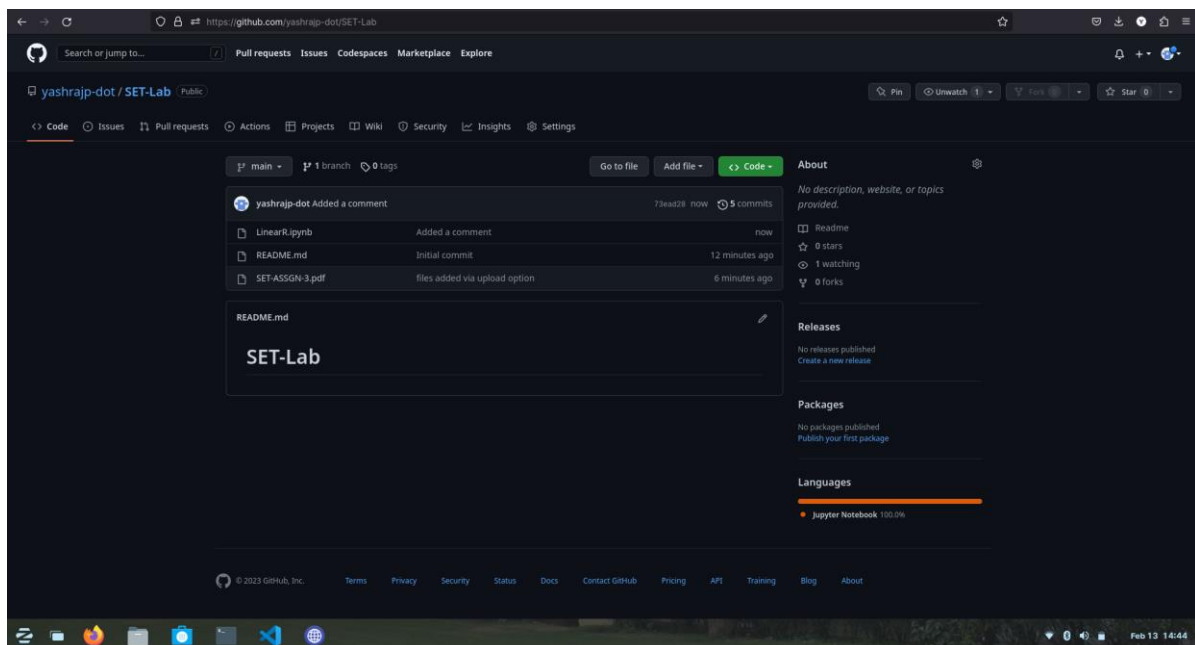
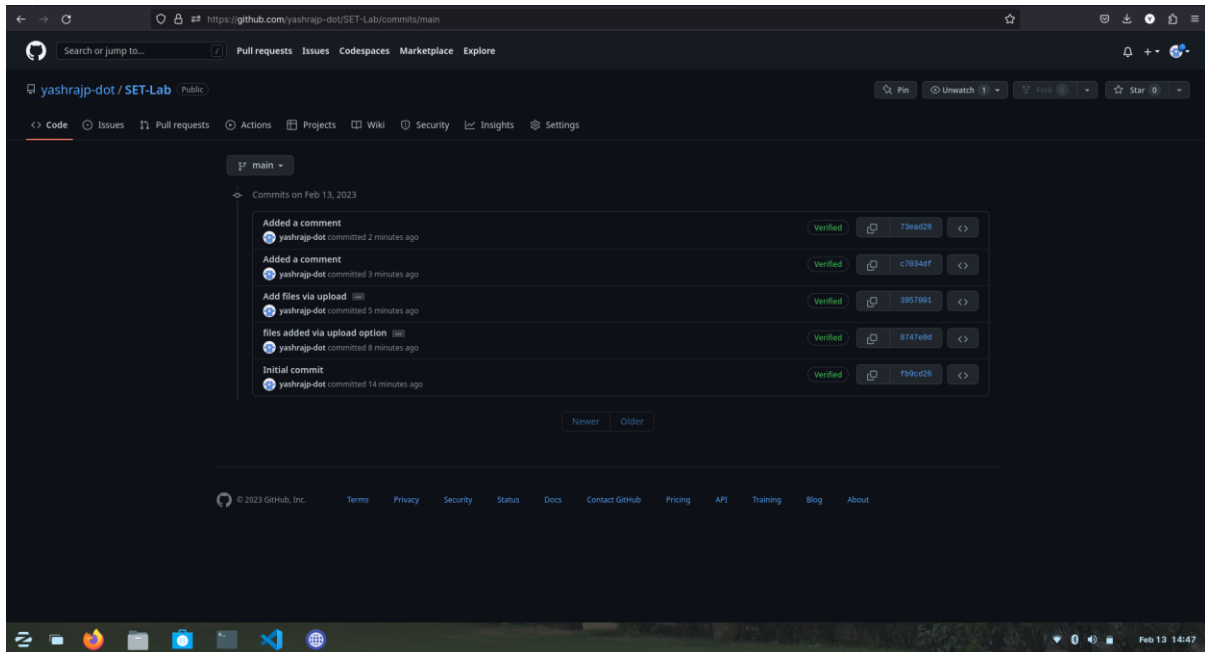




2. Perform update to the existing files (show history)

- Go to the main page of your repository on GitHub.
- Click on the file you want to update.
- Click on the "Edit" button to make changes to the file.
- Enter a commit message to describe the changes you are making.
- Click on the "Commit changes" button.
- To view the history of changes, click on the "History" tab.

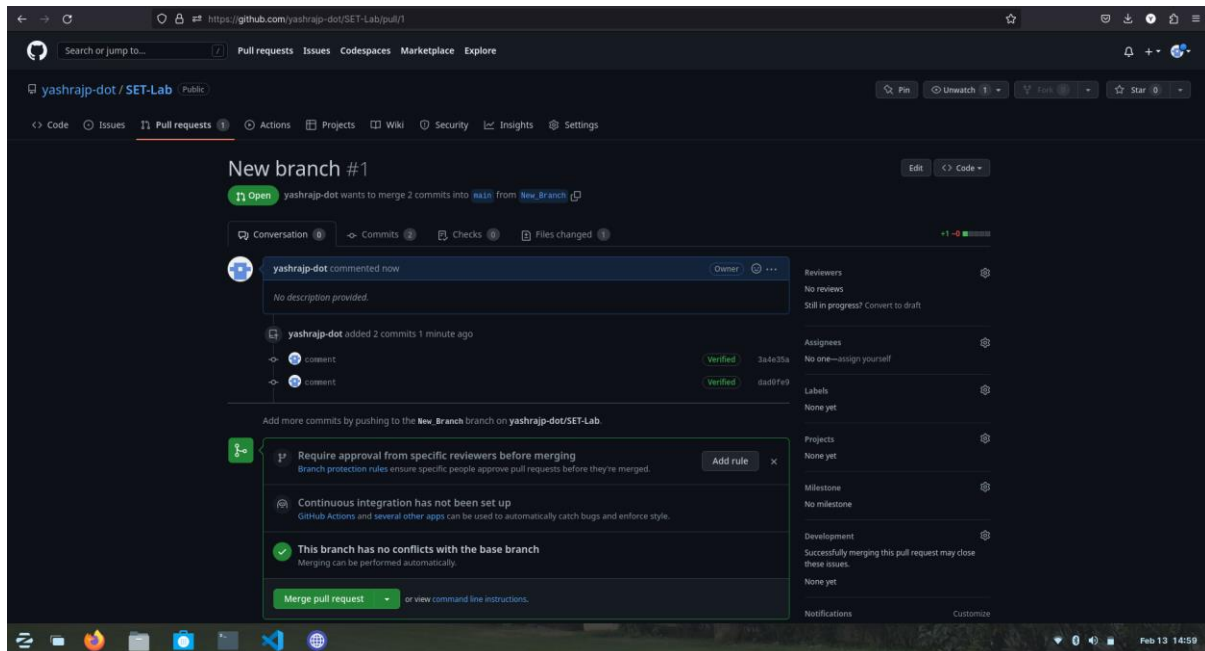




3. Create another branch

To create another branch, follow these steps:

- Go to the main page of your repository on GitHub.
- Click on the "Branch: master" dropdown menu.
- Enter a name for your new branch and click on the "Create branch" button.



4. Create pull request

To create a pull request, follow these steps:

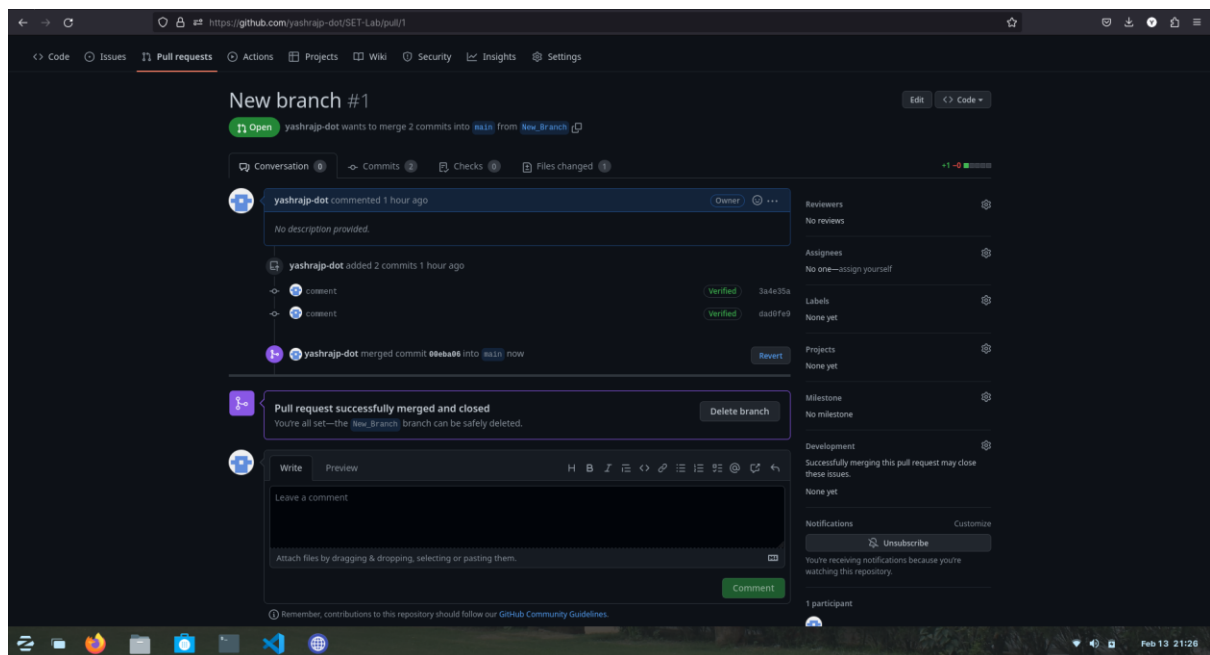
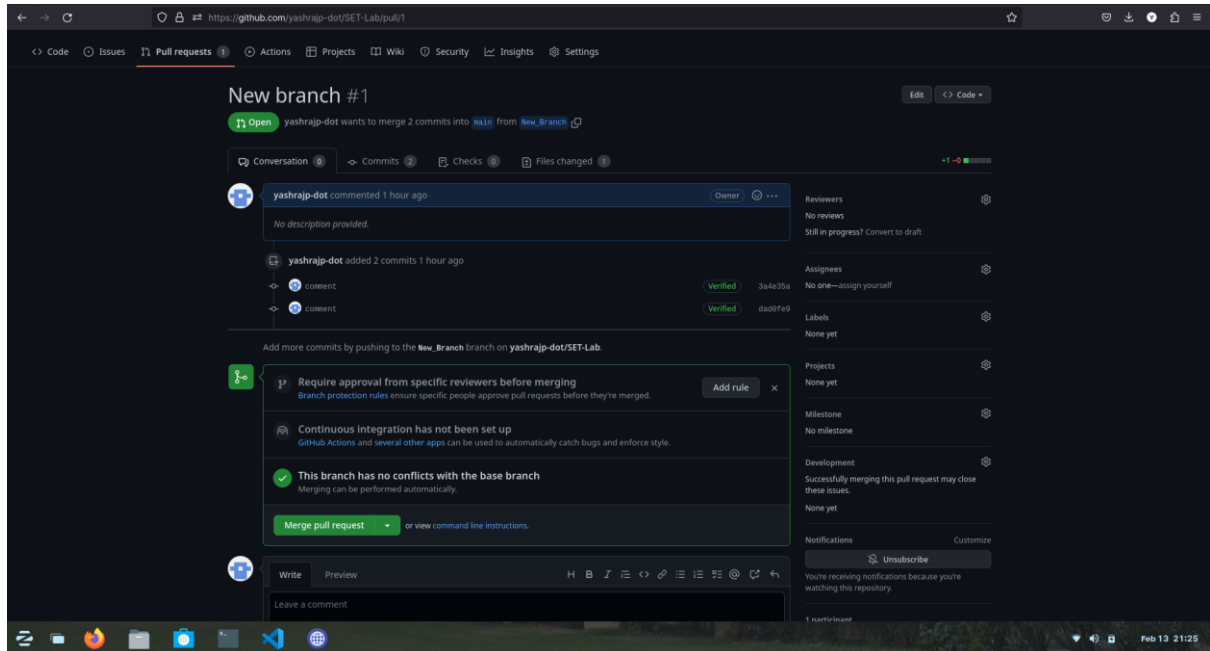
- Go to the main page of your repository on GitHub.
- Switch to the branch you want to merge into the master branch.
- Click on the "Compare & pull request" button.
- Enter a title and a description for your pull request.
- Click on the "Create pull request" button.

The screenshot displays a GitHub pull request interface. At the top, the repository name 'yashraj-dot / SET-Lab' is visible. The pull request title is 'New branch #1'. The status is 'Open', and it indicates that 'yashraj-dot' wants to merge 2 commits into 'main' from 'New Branch'. The conversation section shows two comments: one from 'yashraj-dot' with the text 'No description provided.' and another from 'yashraj-dot' stating 'yashraj-dot added 2 commits 1 minute ago'. The right sidebar contains settings for the pull request, including reviewers (No reviews), assignees (No one—assign yourself), labels (None yet), projects (None yet), milestones (No milestone), development (Successfully merging this pull request may close these issues), and notifications (Customize). The bottom of the image shows a Windows taskbar with various application icons.

5. Perform merging of both branches

To perform the merging of both branches, follow these steps:

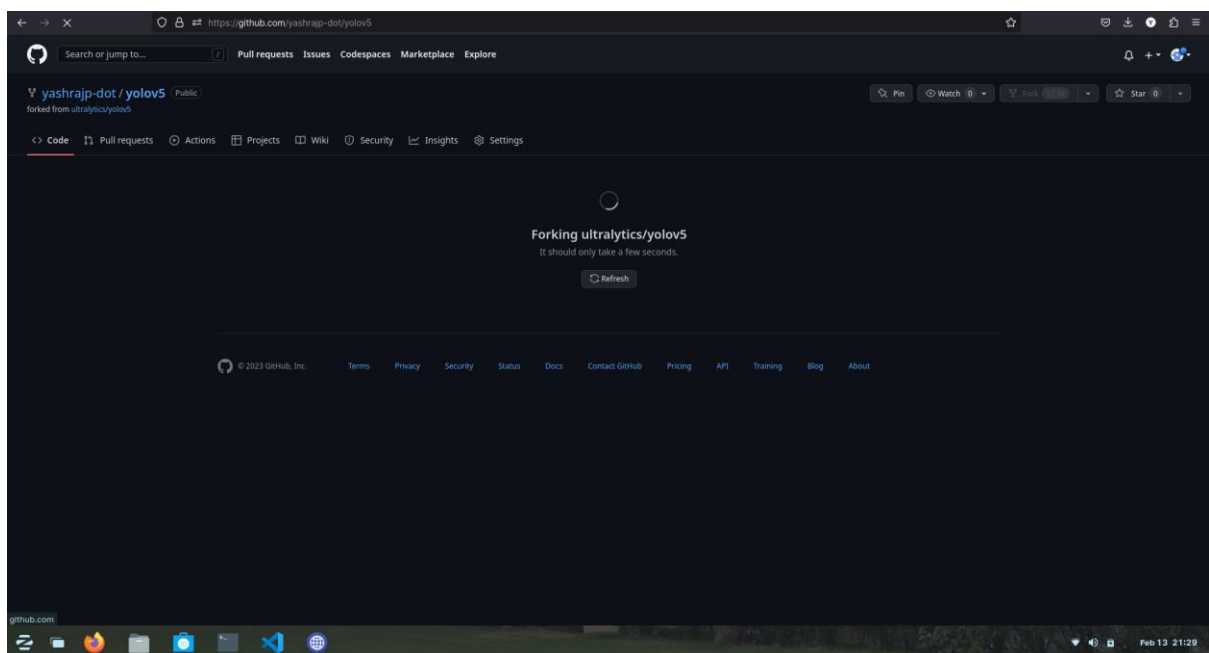
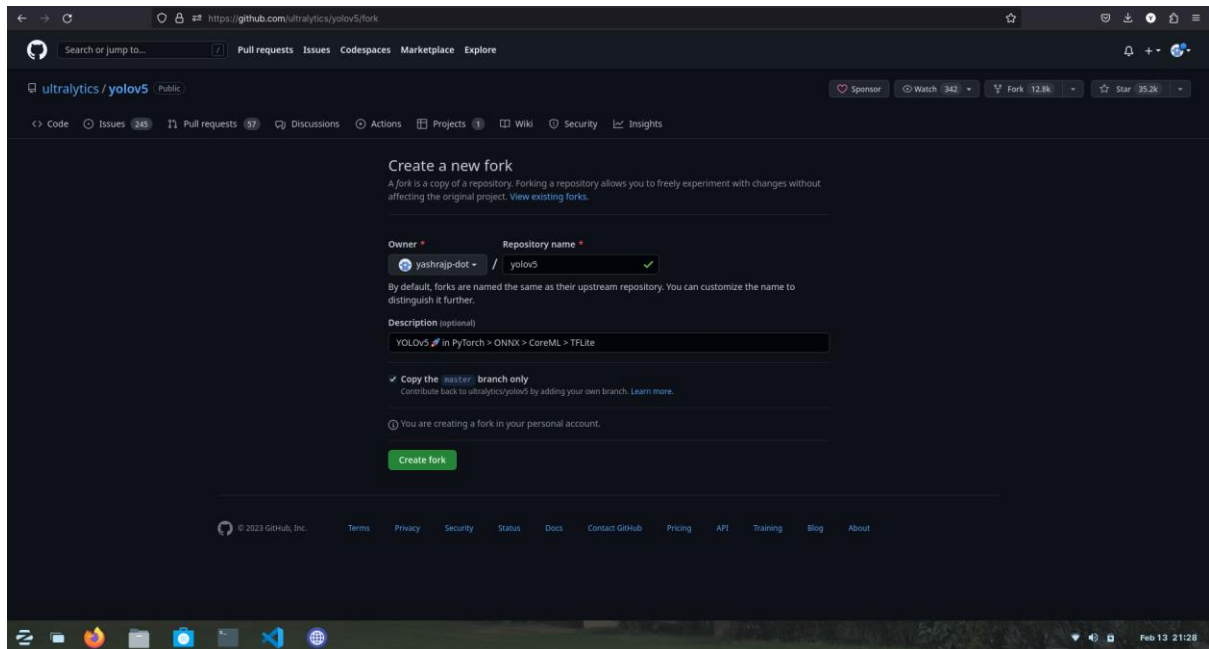
- Go to the pull request you created in the previous step.
- Click on the "Merge pull request" button.
- Click on the "Confirm merge" button.

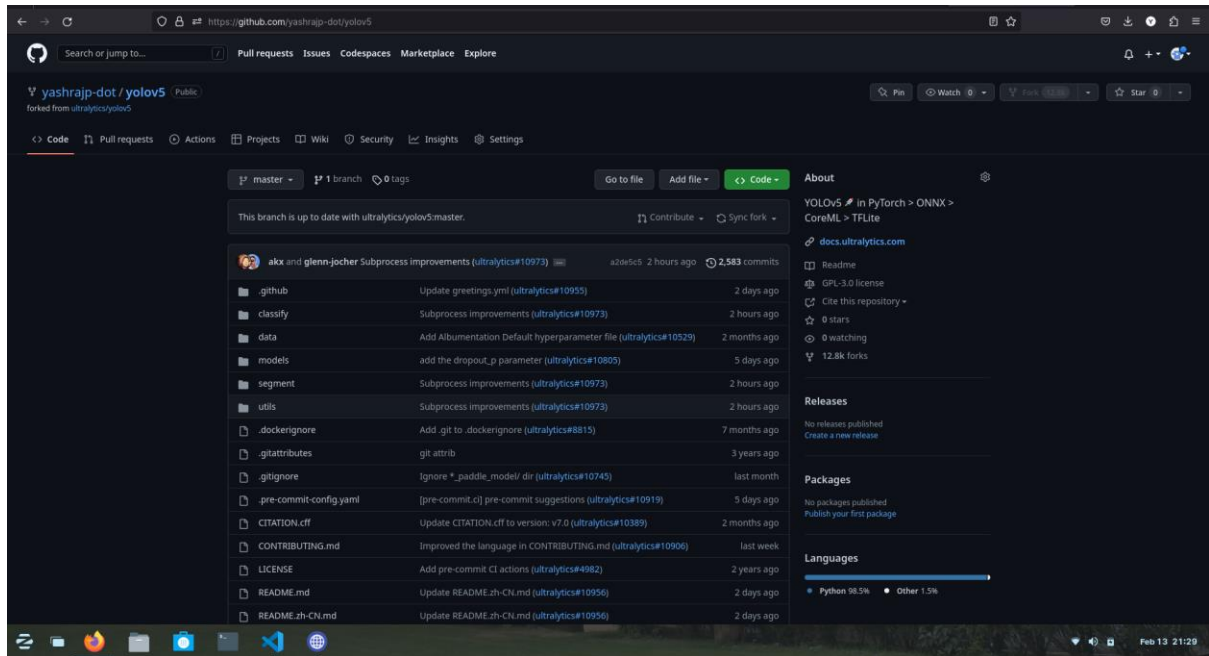


6. Perform Fork operation

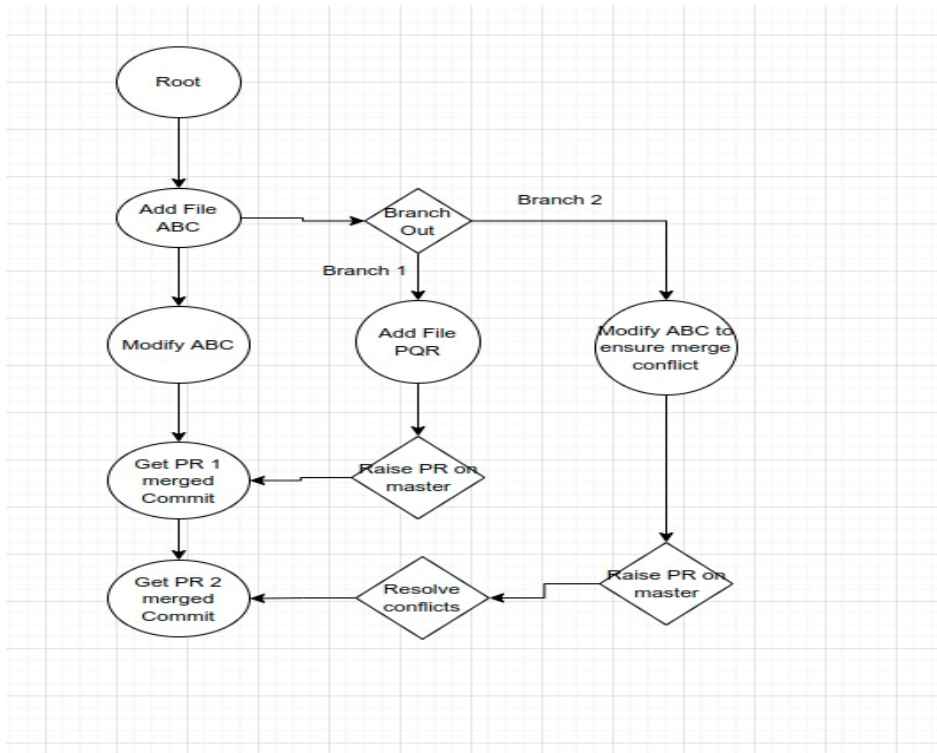
To perform a fork operation, follow these steps:

- Go to the main page of the repository you want to fork on GitHub.
- Click on the "Fork" button in the top right corner.
- Select the account you want to fork the repository to.

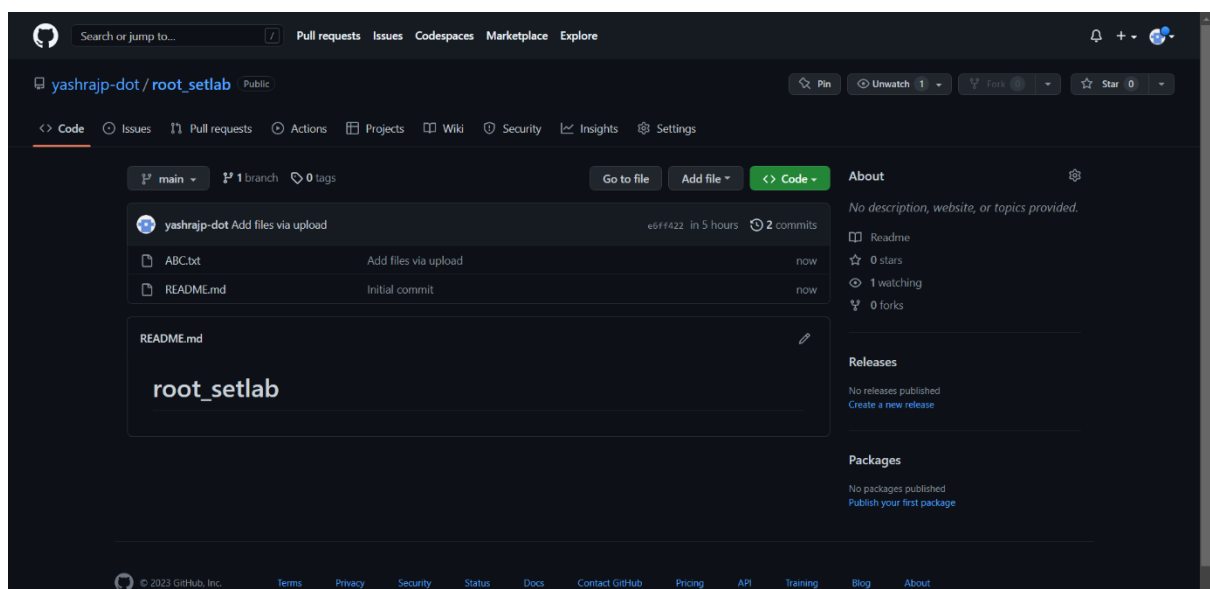




Q 2. For the diagram given below create a GitHub repository and perform operations given in the diagram. (Perform commit operations as given)(Add screenshots as an answer to this question)

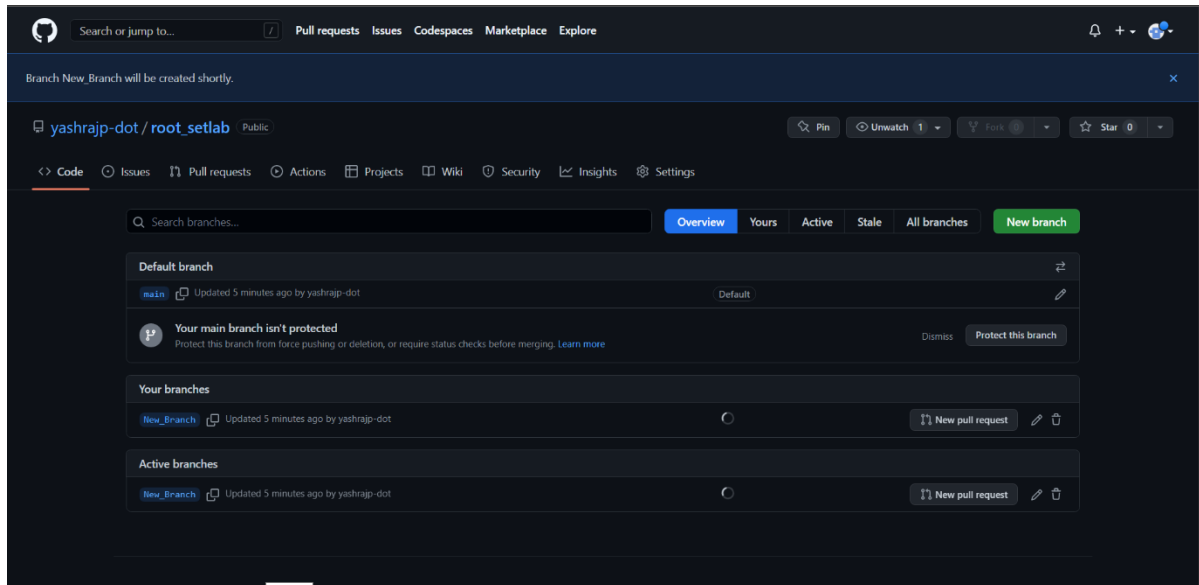


Step 1:
Created New Repo and added a file named ABC.txt

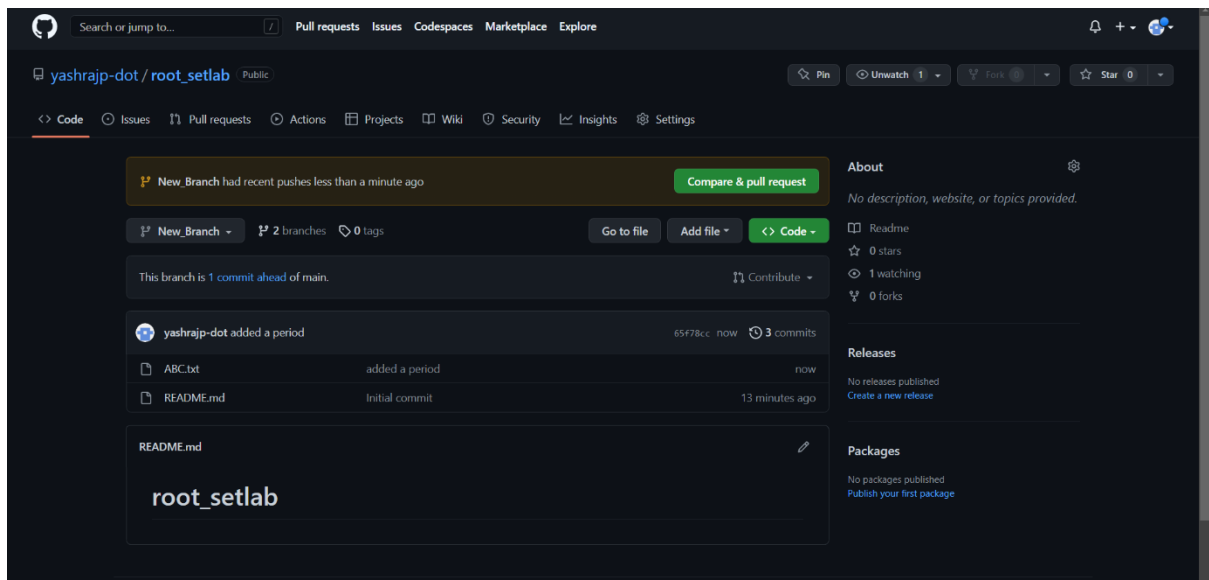


Yashraj Patil
2020BTECS00069

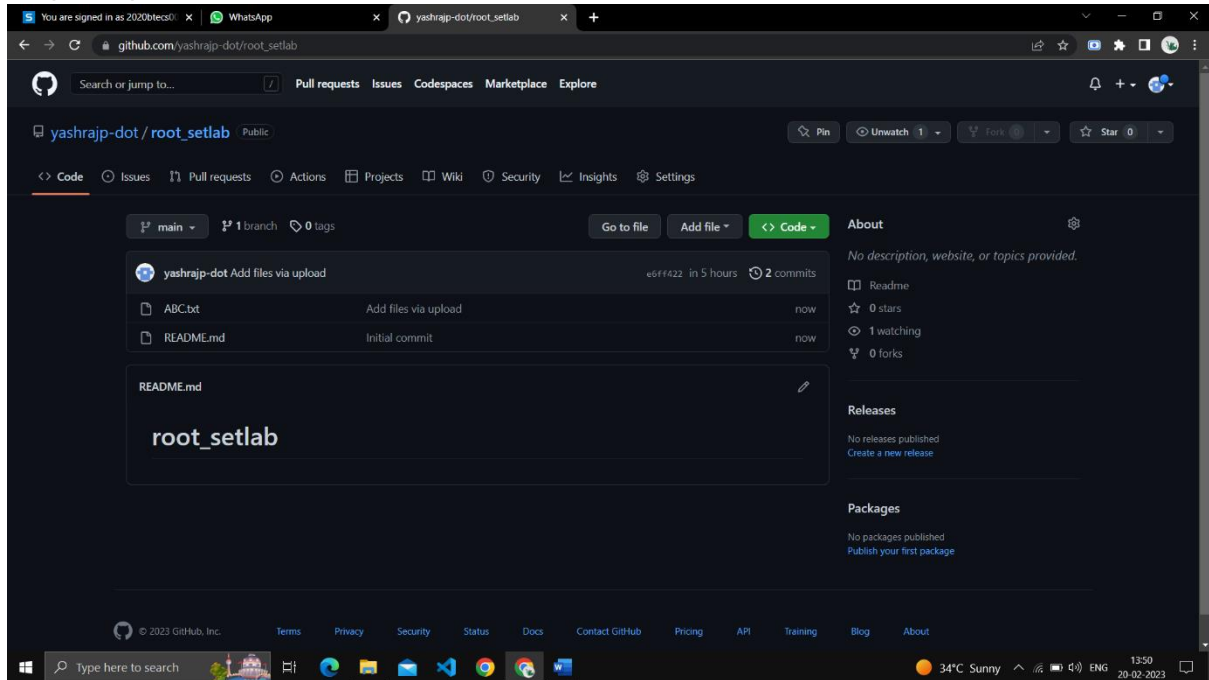
Step 2: Created New Branch.



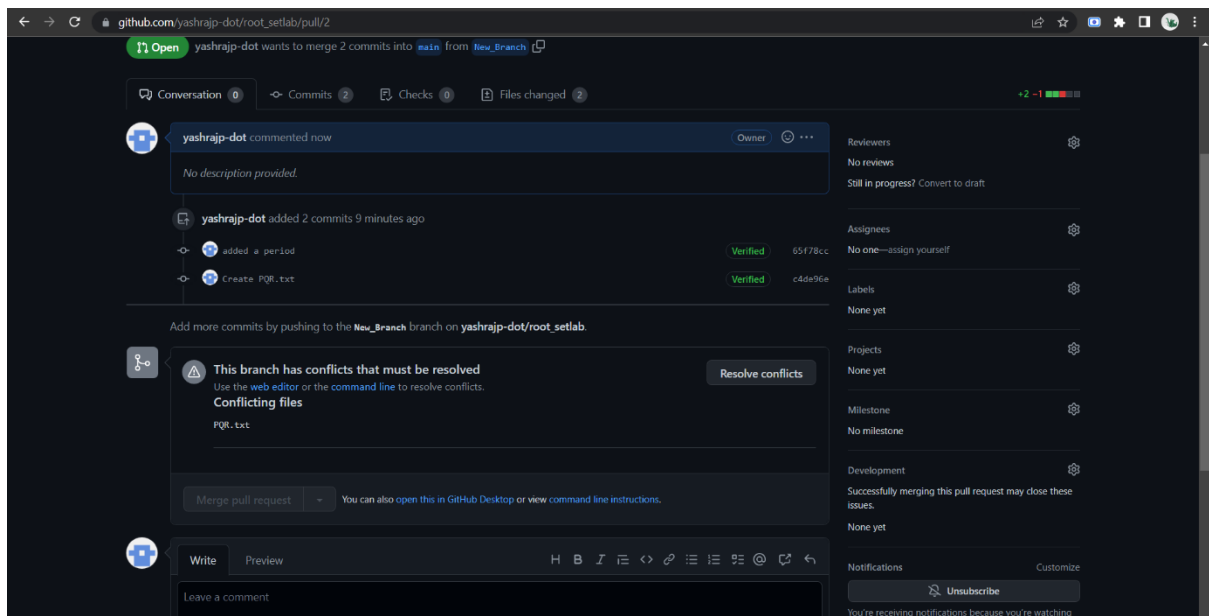
Step3 : Updated file in new branch.



Step 4: Updated main branch file ABC.




Step 5: Merged new branch to main branch




Conflicts found:

Yashraj Patil
2020BTECS00069

Add more commits by pushing to the **New_Branch** branch on **yashrajp-dot/root_setlab**.





**This branch has conflicts that must be resolved**
Use the [web editor](#) or the [command line](#) to resolve conflicts.
Conflicting files
PQR.txt


[Resolve conflicts](#)


Resolved Conflicts:

Add more commits by pushing to the **New_Branch** branch on **yashrajp-dot/root_setlab**.



**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.

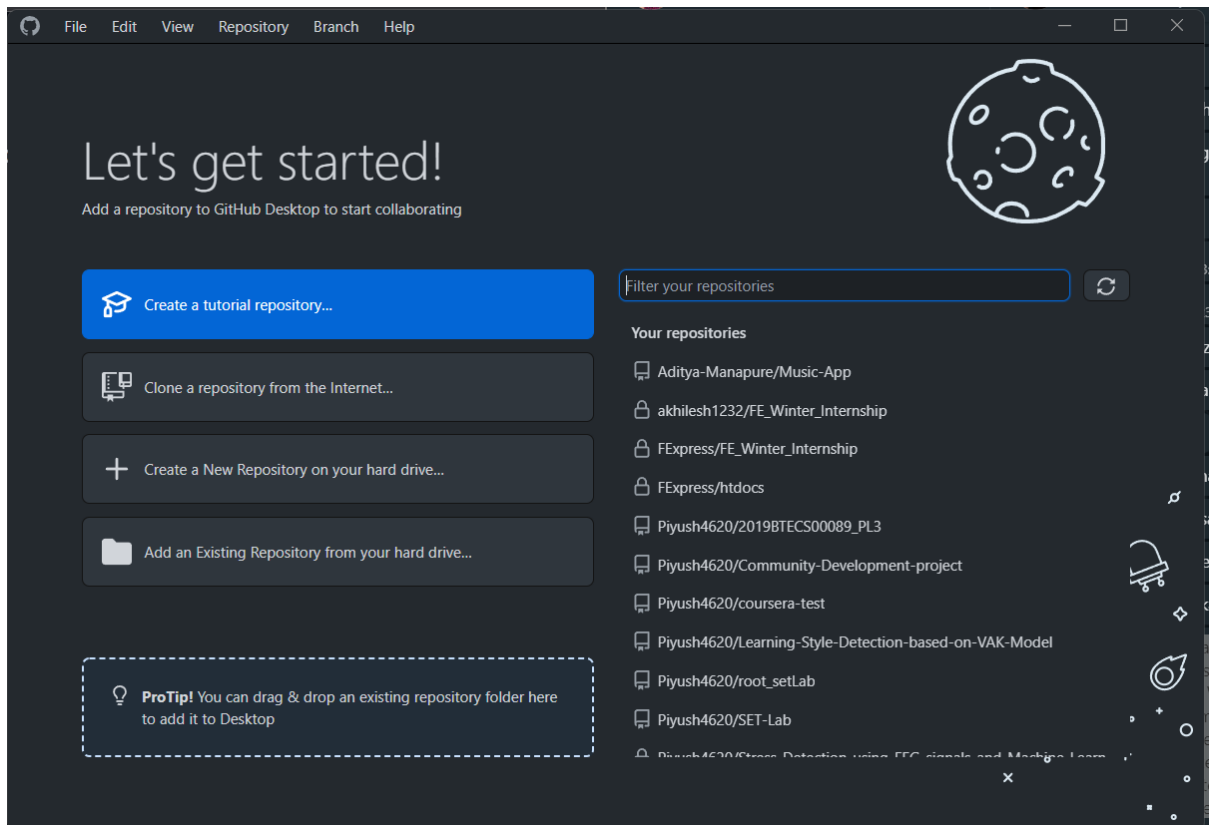
[Merge pull request](#)  You can also open this in [GitHub Desktop](#) or view [command line instructions](#).

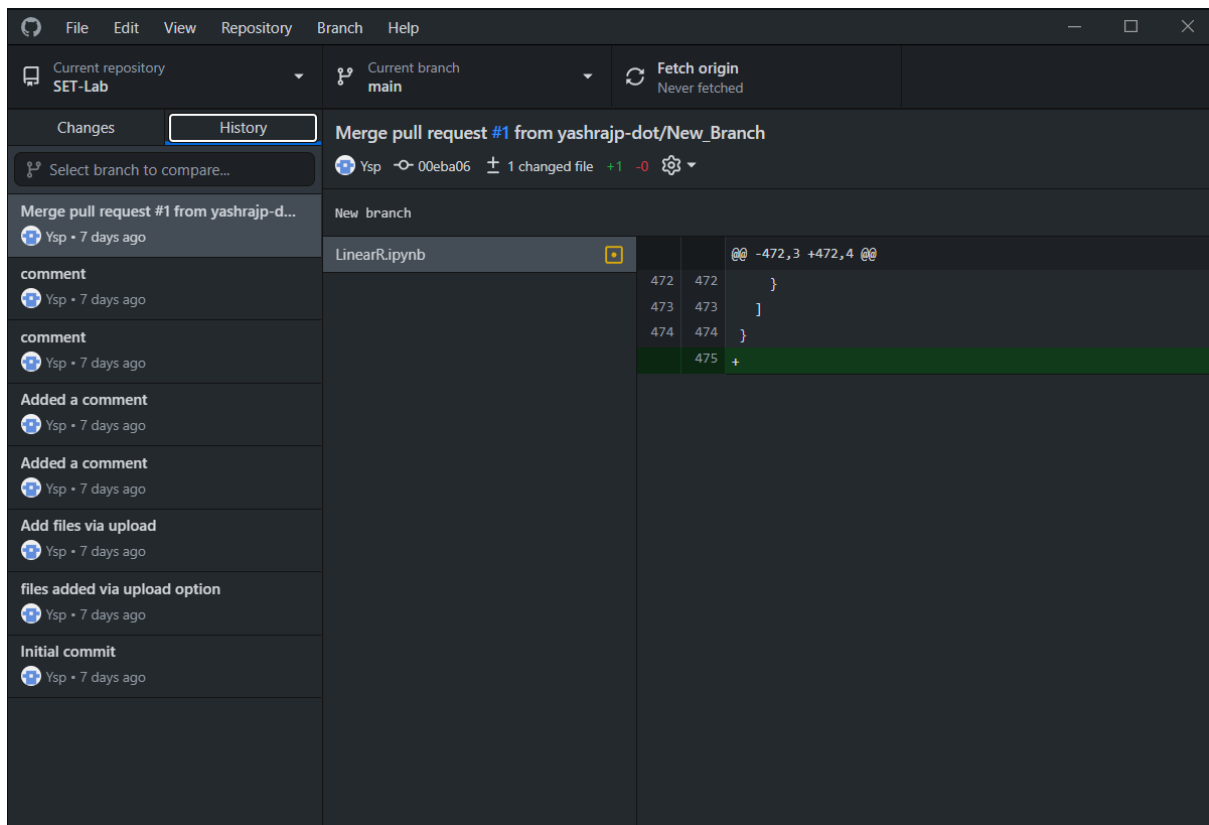
Q 3. What is GitHub desktop? How to install GitHub on a local machine? Install GitHub on your local machine and access repository created in question no 1 (add screenshots).

GitHub Desktop is a free and open-source graphical user interface for the Git version control system. It provides an easy-to-use and streamlined interface for working with Git repositories on your computer. With GitHub Desktop, you can manage your repositories, view changes to your code, merge branches, and make commits directly from the application. It is designed to make working with Git more accessible, especially for users who are new to Git or who prefer a graphical interface. The application is available for Windows and macOS, and it integrates with the GitHub.com web-based platform for hosting and sharing code.

❖ Here are the steps to install GitHub on your local machine:

1. Install Git: GitHub Desktop is built on top of Git, so you need to have Git installed on your machine first. You can download the latest version of Git from the official website (<https://git-scm.com/downloads>). Follow the instructions for your operating system to install Git.
2. Download GitHub Desktop: Visit the GitHub Desktop website (<https://desktop.github.com/>) and click on the "Download for Windows" or "Download for macOS" button, depending on your operating system.
3. Install GitHub Desktop: Once the download is complete, run the installer and follow the on-screen instructions to install GitHub Desktop on your machine.
4. Launch GitHub Desktop: After installation, launch GitHub Desktop from your start menu or dock. You will be prompted to sign in with your GitHub account. If you do not have a GitHub account, you can create one for free on the GitHub website.





Q 4. Differentiate in between GitHub, Git and GitLab.

Git, GitHub, and GitLab are related but distinct tools in the world of version control and collaboration. Here's a brief overview of each:

1. **Git:** Git is a distributed version control system that was created by Linus Torvalds in 2005. It allows developers to track changes in their code and collaborate with other developers on a project. Git is the underlying technology that powers most version control systems, including GitHub and GitLab.
2. **GitHub:** GitHub is a web-based platform that provides hosting for Git repositories, as well as a range of tools for issue tracking, project management, and collaboration. GitHub is one of the largest hosting platforms for open-source projects, and it provides a wide range of tools and integrations to help developers work together.
3. **GitLab:** GitLab is an open-source Git repository manager that provides a web-based Git repository manager, issue tracking, and project management tools. GitLab also provides a range of additional tools and features, including continuous integration and deployment, security scanning, and more.

Q 5. What is version control? Explain with example.

Version control is a system that records changes to a file or set of files over time so that you can recall specific versions later. It is commonly used in software development to track changes in source code and to collaborate with other developers.

For example, imagine that you are working on a project with a team of developers, and you all make changes to the same code base. Without version control, it can be difficult to keep track of who made which changes and when, and you may overwrite each other's work. With version control, you can each make your changes in a separate branch, and then merge those changes into a common branch when they're ready.

Each time you commit a change using version control, you create a new version of the code base, which is stored in a repository. You can then use the version control system to view the history of changes, compare different versions, and revert to an earlier version if necessary.

There are several version control systems available, including Git, Subversion, and Mercurial. Each has its own strengths and weaknesses, and the choice of which one to use depends on the needs of the project and the team.