ATLANTIC TECHNOLOGICAL UNIVERSITY

ASSIGNMENT COVER SHEET

To Be Completed By The Student

Lecturer’s Name: Maria Griffin

Assessment Title:

Submission Date:

Student’s Name: Arun Jayarajan Id. Number: L00171224\_\_\_\_

Course / Stage Master’s in DevOps

Subject/Module: DevOps Software Engineering (2022/23)

Word Limit: Actual Word Count: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

I confirm that the work submitted has been produced solely through my own efforts.

Student’s signature: Arun Jayarajan Date: 20-Nov-2022

|  |
| --- |
| **Note**: **PENALTIES**   * The total marks available for an assessment will be reduced by 15% for work submitted up to one week late. The total marks available are reduced by 30% for work up to two weeks late. * Assessment work received more than two weeks late, without prior approval by the lecturer will receive a mark of zero. * Marks awarded will be reduced by 10 % if submitted work is greater than 10% above or below the assigned word limit. * A further hard or electronic copy of your submitted work may be requested, and therefore you must keep a copy on disc. * Incidents of alleged plagiarism and cheating are dealt with in accordance with the Institute’s Assessment Regulations   **Plagiarism:** Presenting the ideas, words of someone else without proper acknowledgement. Refer to the Institutes’ procedures and guidelines for the assessment of learners. |

## Introduction

This lab work is done to deep dive into the GitHub code repository. Exploring the features and possibilities in GitHub as a version control tool.

# Aims/Objectives

1. Examine the setup of a GitHub repository using a sample piece of code.
2. Learn how to use the git commands and UI to perform many operations.
3. Examine the GitHub branching process.
4. Explore the workings of the branches and commits.
5. Investigate on merging (pull requests).
6. Explore merge conflicts and resolve conflicts with both editor and Web UI.
7. Learn more about GitHub as I have never used it.

## Method

1. Created a GitHub Repository - <https://github.com/arunjayarajan/Lab1>
2. Pushed a sample code into the above-mentioned repository using command Line.
3. Installed VS Code
4. Cloned the repository in VS Code.
5. Made changes in a java file and did a commit with command line and pushed to main branch.
6. Created a branch – “Feature1” using command line.
7. Made changes in a java file in Feature1 branch and merged the changes to main.
8. Made changes to same file in both “main” and “Feature1” branch to create conflicts.
9. Resolved the conflicts in VS Code by clicking “Accept Both” changes.
10. Made change to the same file again to get one more conflict.
11. Created a pull request from GitHub UI to merge from Feature1 to main, resolved the conflicts and merged the code to main.
12. Made Pull Request mandatory to merge the code to main branch.
13. Made changes to “Feature1” branch and merged the code to “main”, It asked for code review and the pull request was approved and then merged the code to main.
14. Cloned a repo of colleague <https://github.com/L00170985/Lab_1_Git/tree/Feature_Arun>
15. Created a branch in the repo and made changes to the code of the colleague’s code.
16. Stashed the changes and made another change to the same code.
17. Stashed the change again and did a stash pop to take the changes again and pushed the changes to the repository.
18. Added 3 text files – Note1.txt, Note2.txt and Note3.txt. Did a git revert for Note2 and could see the Note2.txt is deleted from the local repository.

## Results

1. Successfully Created a repository in GitHub and was able to add sample code in it.
2. I had problems in cloning the repository at first with VS code but was later resolved.
3. Made code changes and pushed to repository with command line.
4. Also, made changes to the code and pushed the changes with VS Code.
5. Branch “Feature1” was created to test merging and was able to work on merging with both command line as well as UI. Successfully merged the code changes from feature branch to main branch.
6. Worked on merge conflicts and resolved the conflicts through VS Code and Web UI
7. Resolving merge conflicts is easy in Web UI than in the command line.
8. Worked on different repository and worked on git stash which helps in keeping the changes aside and work on different set of changes. We can always pop the changes again and check-in the stashed changes.
9. Worked on git reflog command, we can see the list of commits in the repo.
10. Explored git revert command, where we can revert the commits done.

## Conclusion

GitHub is a cloud-based code hosting service based on Git version control system. I understood that it helps in collaboration among the team of developers and is equipped with version control and access control. It’s a tool which is very easy to use with the Web UI. The command line makes it even easier to use. Almost all the actions can be done from both command line and WebUI.

I was able to create the repository in the GitHub and clone the repository to my local with command line. I felt that creating branches in the repository makes it easy for the developers to work on a single code without any issues. It is not practically possible for two developers to work or commit on the same branch. Because, when pushing the changes to repository, the local should have the latest code base. So, when two are working simultaneously, and one has done the code changes in the local and other had already pushed the code to repository, The first one cannot push the code to repository without taking the latest code again. So here comes the use of branches.

As a best practice, Feature branches has to be created from main branch for a developer to work on a code base and then merge the changes to the main branch with a pull request. So, with this, new features for a software can be created without disturbing the main code. The code can be tested in the feature branch and pushed to main branch if the code is verified good. A rule can be set to protect the branches from making direct commits. By enabling this option, the code changes should be done on a feature branch and a pull request has to be created to merge the changes to the protected branch. With a pull request, we can set code review process in place. A team leader can verify the changes done by the team member before approving the pull request [Figure 25 – Figure 26].

We can always do a git revert for the commits we did in the repo. The revert will also will be recorded as a commit, so all the changes are tracked [Figure 29].

In my opinion, WebUI makes it simpler to resolve merge issues than the command line does. [Figure 15 – Figure 17]

We can setup code scanning within the Git Hub for the code in the repository. For this we need to configure a scanning tool like CodeQL or checkmarx. Also, there are Dependabots to alert the users when it finds a vulnerability.

When compared to Bitbucket, GitHub is much more user friendly. It’s easier to work on code with GitHub. In both Bitbucket and GitHub, we have options for the CI\CD pipelines. We can use GitActions in GitHub and have pipelines option within Bitbucket. One notable difference between GitHub and Bitbucket is about giving permissions to the branch level. In Bitbucket, we can set the permissions for a branch, we can specify the user list to have the permission to the branch [Figure 30]. Whereas in GitHub, the permissions can be given in repository level only. Bitbucket is from Atlassian, so all the products of Atlassian is easily connected to Bitbucket. We can effortlessly connect to Jira, Confluence, Trello.

In Bitbucket, we can integrate Snyk code scanning tool. By giving permissions, the code in bitbucket gets uploaded to Snyk and it gives you a report on code vulnerability (Figure 31).

In both GitHub and Bitbucket, the build and deploy pipelines can be created using .yml files.

## References and Bibliography

Elegant Themes Blog (2022), Available at: <https://www.elegantthemes.com/blog/wordpress/github-vs-bitbucket> (Accessed: 20 October 22)

HubSpot Blog (2022), Available at: <https://blog.hubspot.com/website/what-is-github-used-for> (Accessed: 20 October 22)

Interviewbit (2022), Available at: <https://www.interviewbit.com/blog/bitbucket-vs-github/> (Accessed: 16 October. 22)

Kinsta (2022), Available at: <https://kinsta.com/knowledgebase/what-is-github/> (Accessed: 16, October 2022).

Tunggal, AT. UpGuard (2022), Available at: <https://www.upguard.com/blog/bitbucket-vs-github> (Accessed: 16 October 2022)

## Appendices

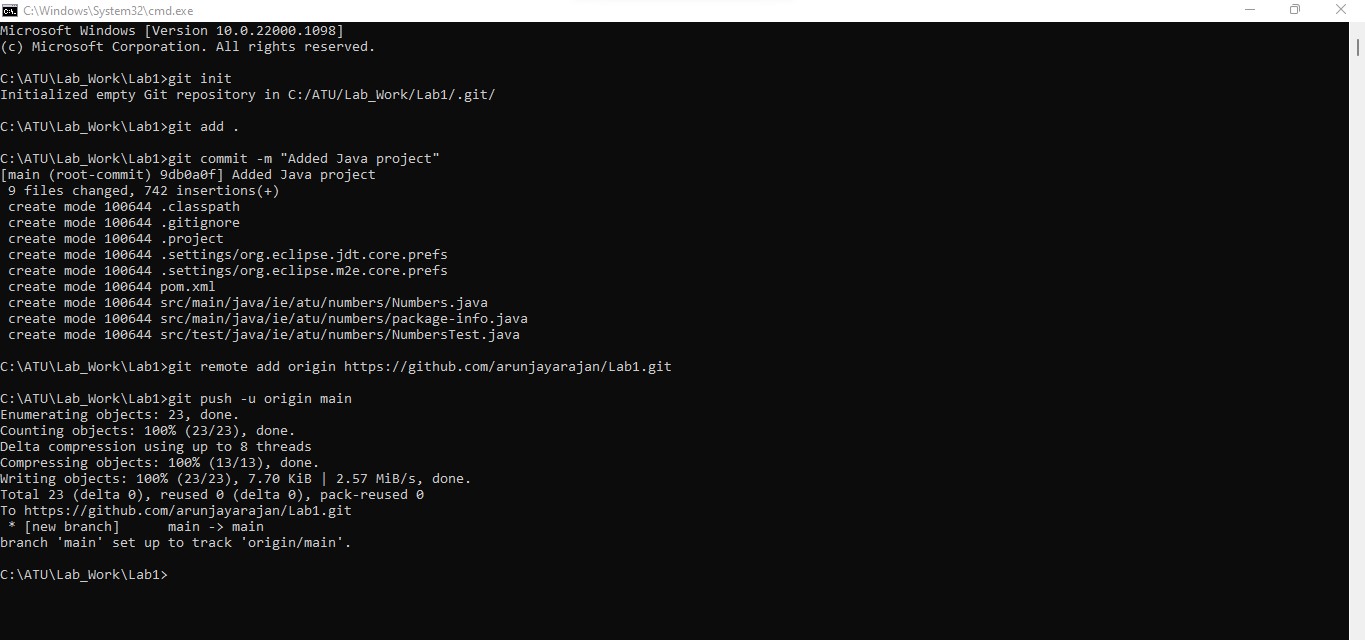
Figure1: Git Initialised and pushed the changes to GitHub repository. 

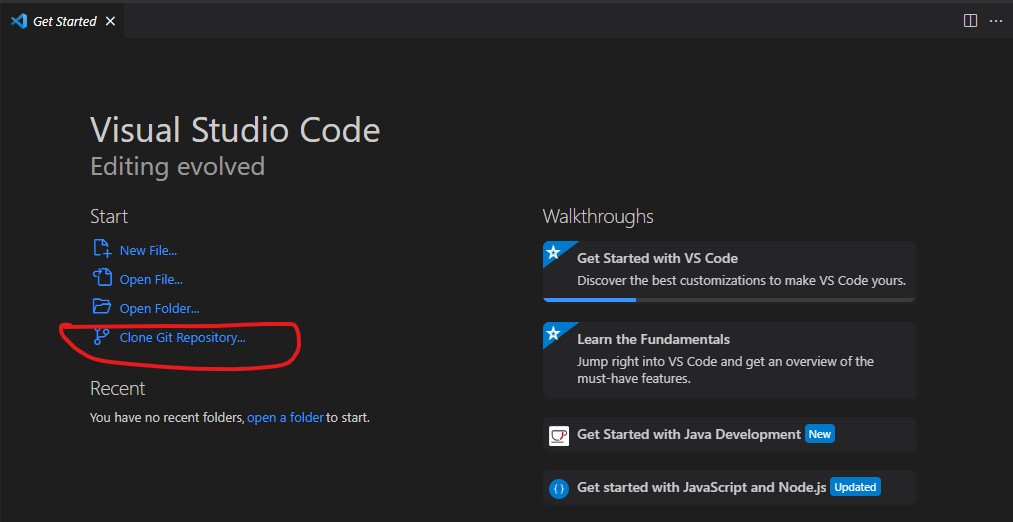
Figure 2: Cloned the same git repo from VS Code

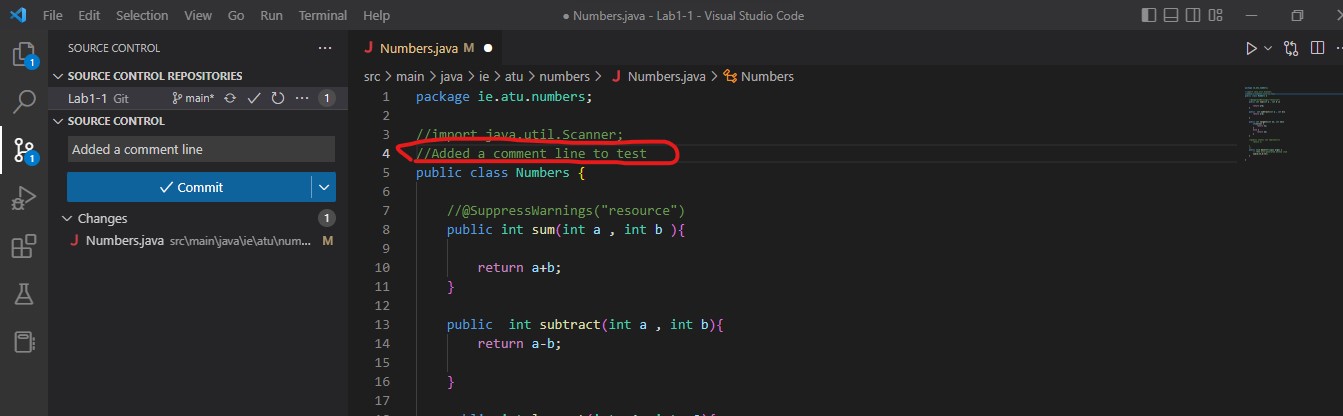
Figure 3: Updated the code using VS Code and commited.

Figure4: Git fetch and pull command

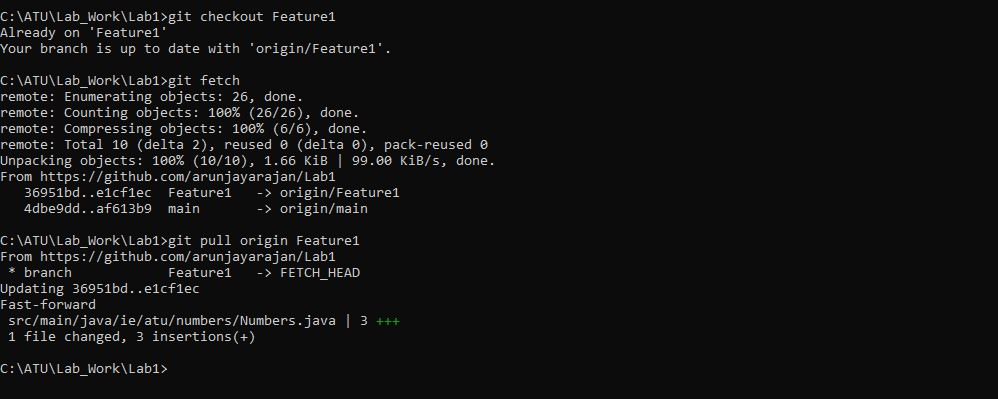


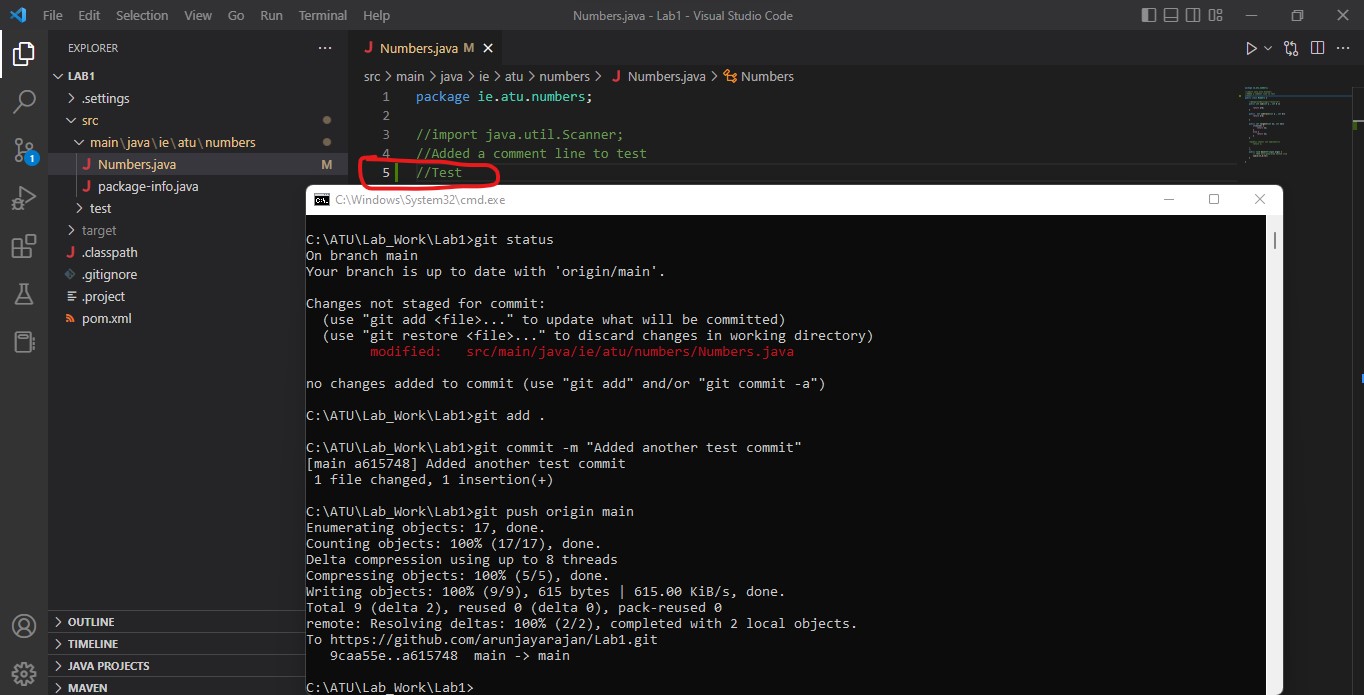
Figure 5: Updated the code again and pushed using command line. 

Figure 6: This figure shows that we are on main branch.

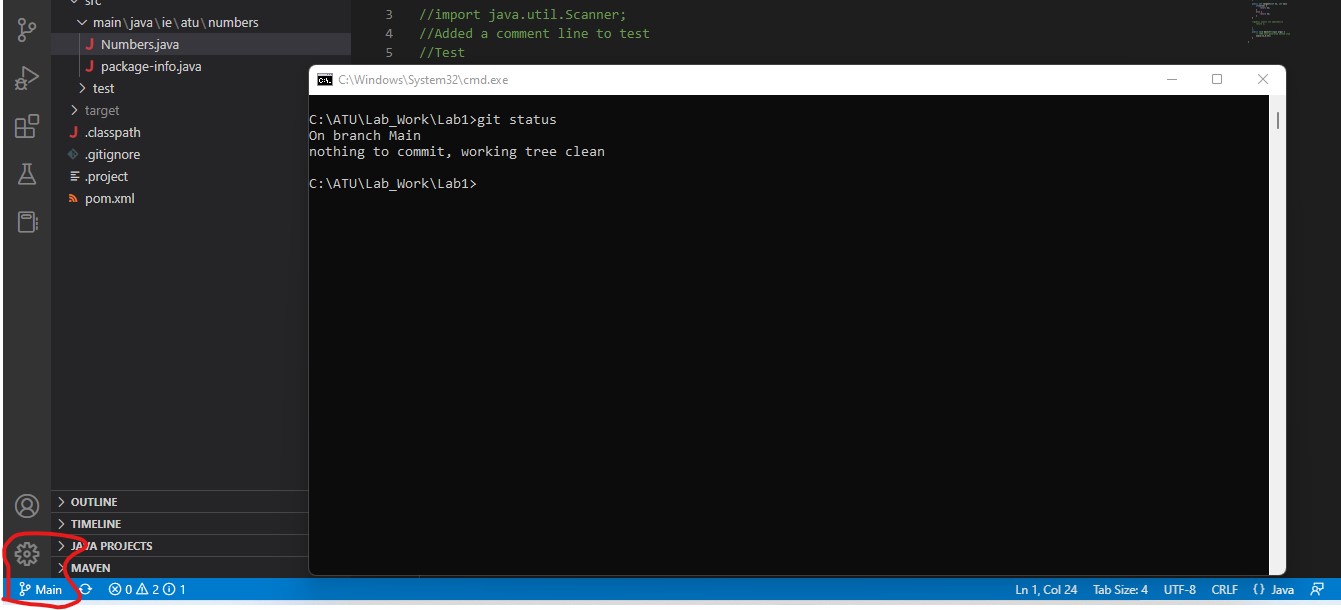


Figure 7: Branch automatically switched to Feature1 in VS code when we switched to Feature1 branch from command line.

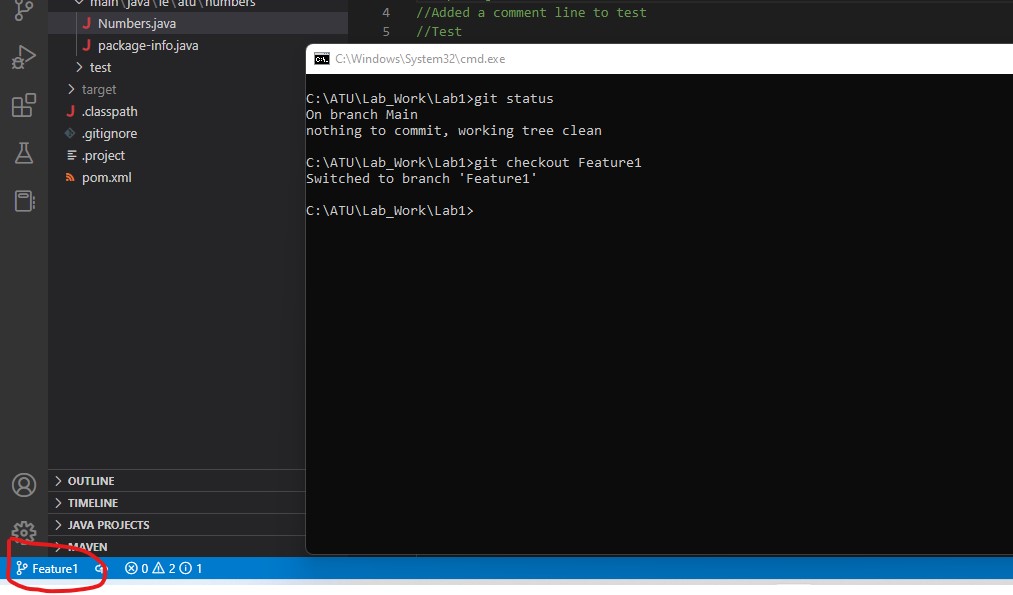


Figure 8: Changed merged from Feature1 to main.

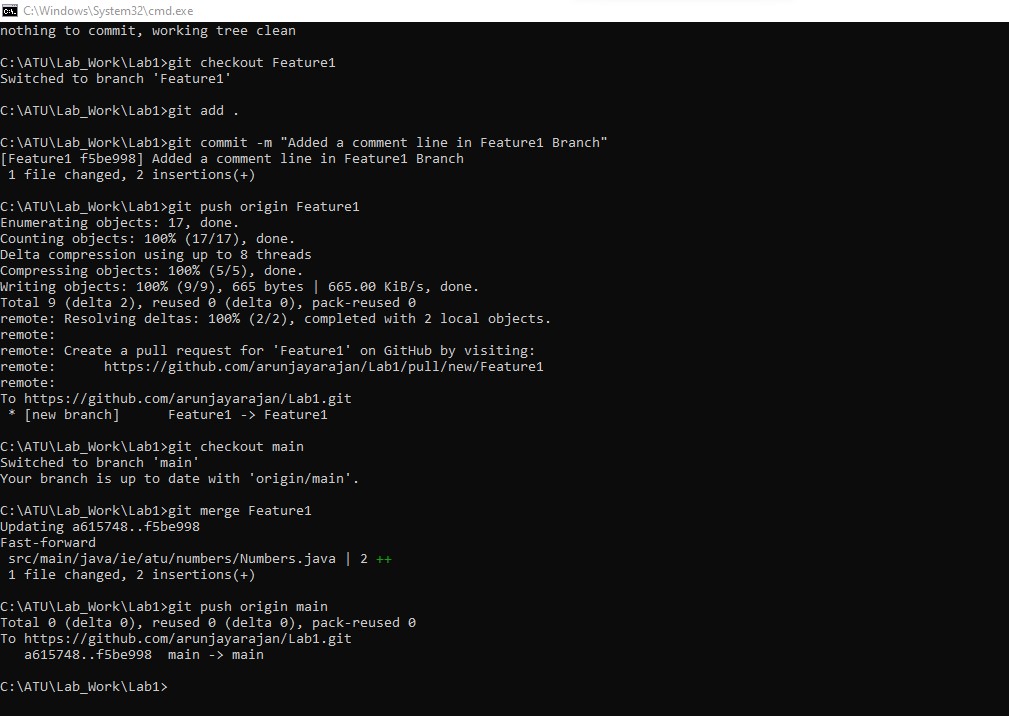


Figure 9: Changes reflected in Feature1 branch.

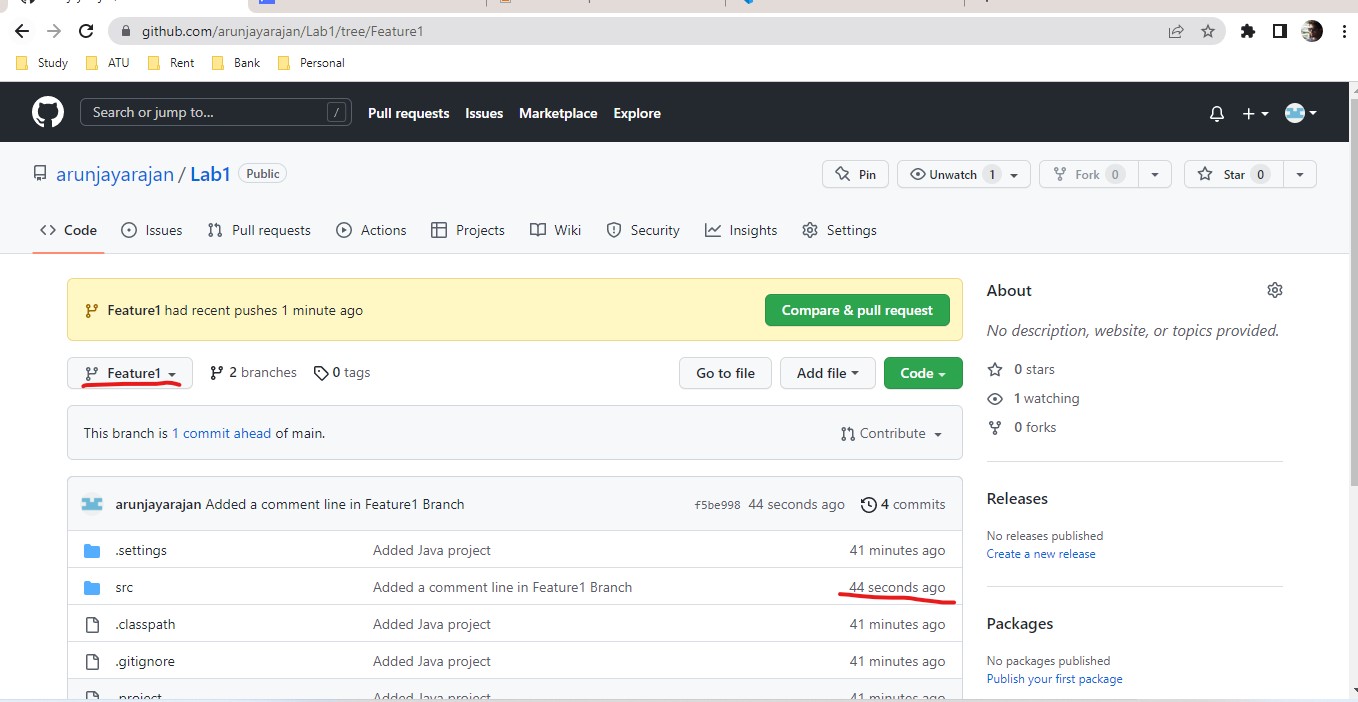


Figure 10: changes reflected in main branch, verified in WebUI

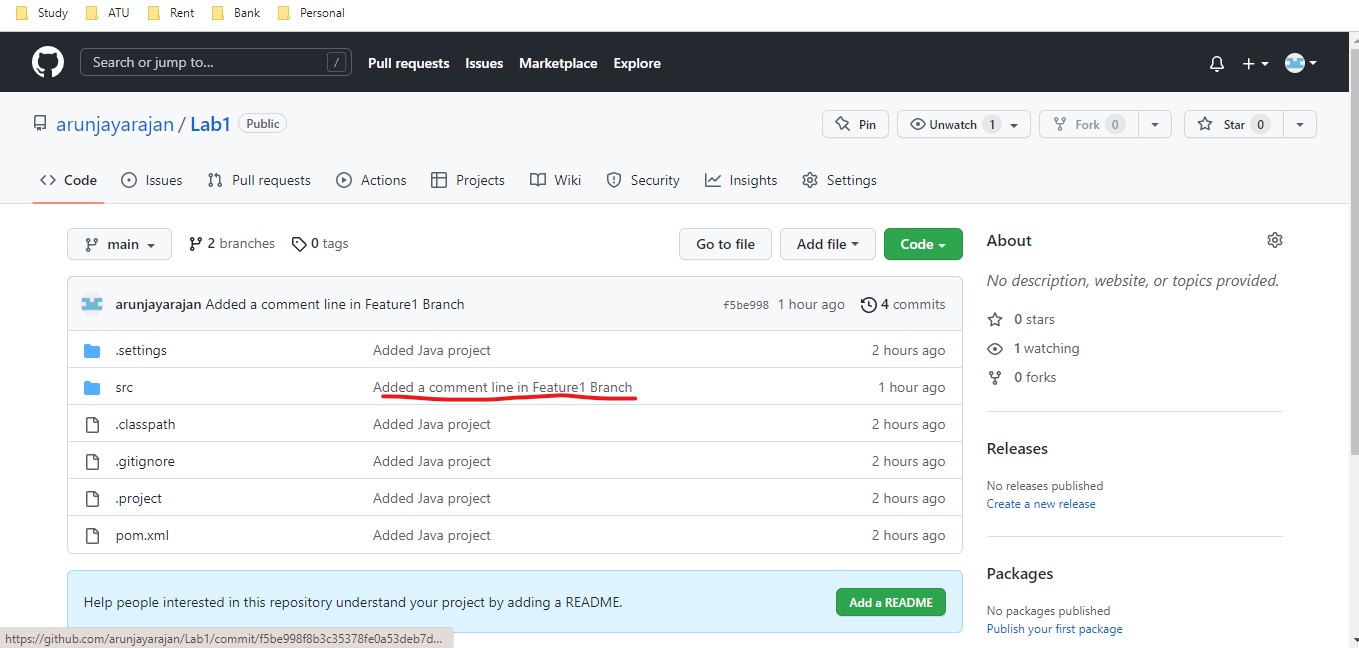


Figure 11: Made change to same file in both main and Feature1 branch to get a merge conflict. This figure shows the merge conflict.

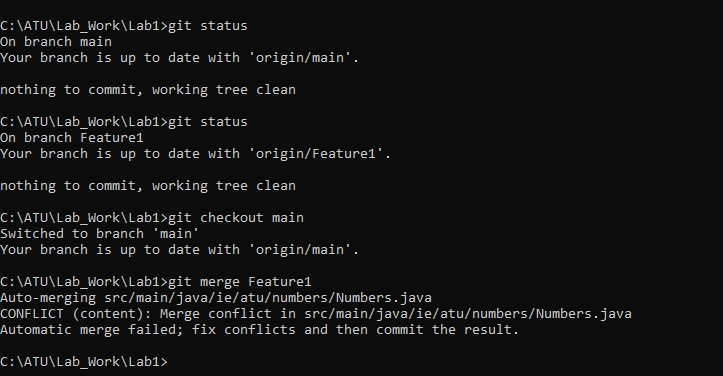


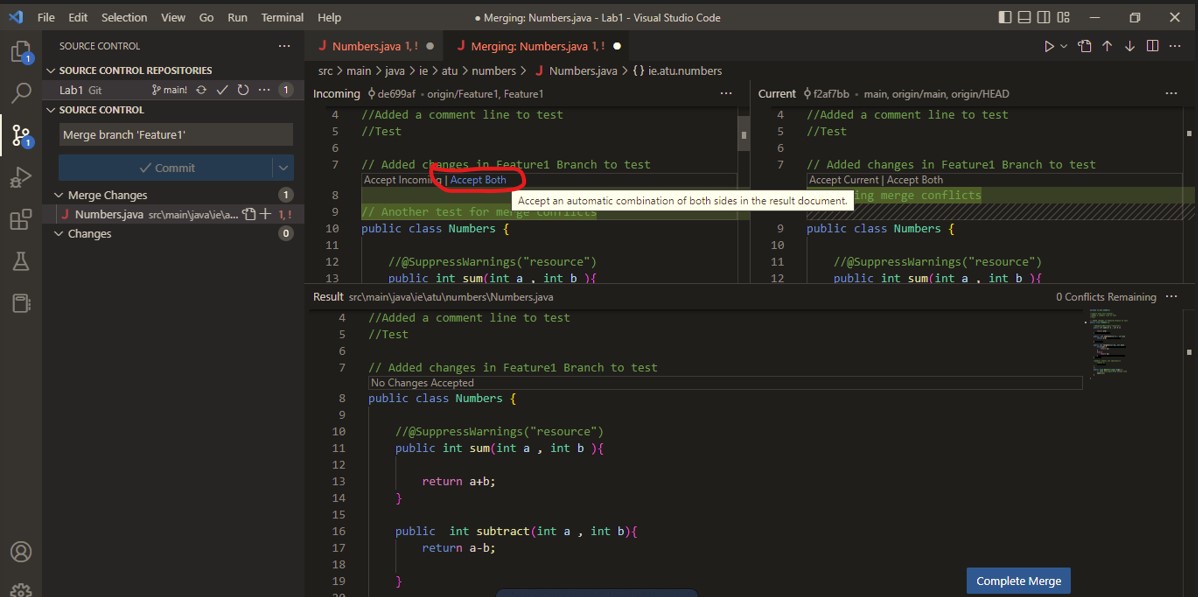
Figure 12: Conflict resolved from VSCode.

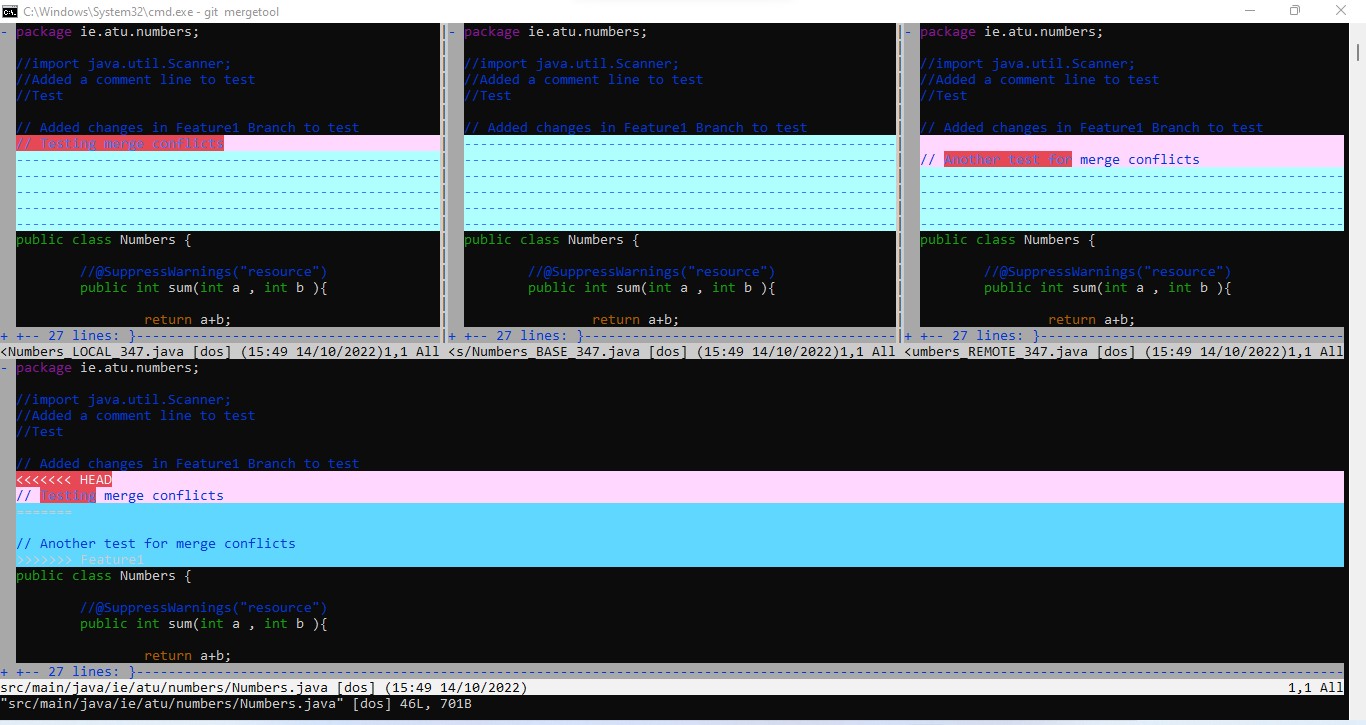
Figure 13: Conflicts can be seen from command line with git mergetool

Figure 14: Merge conflicts resolved and pushed to main.

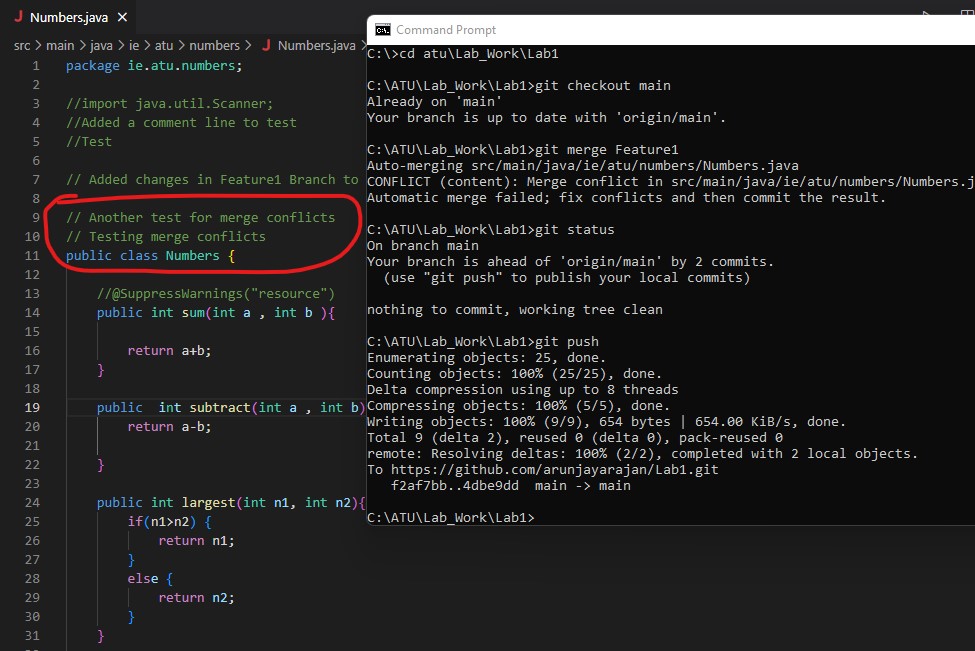


Figure 15: Pull request created from WebUI.

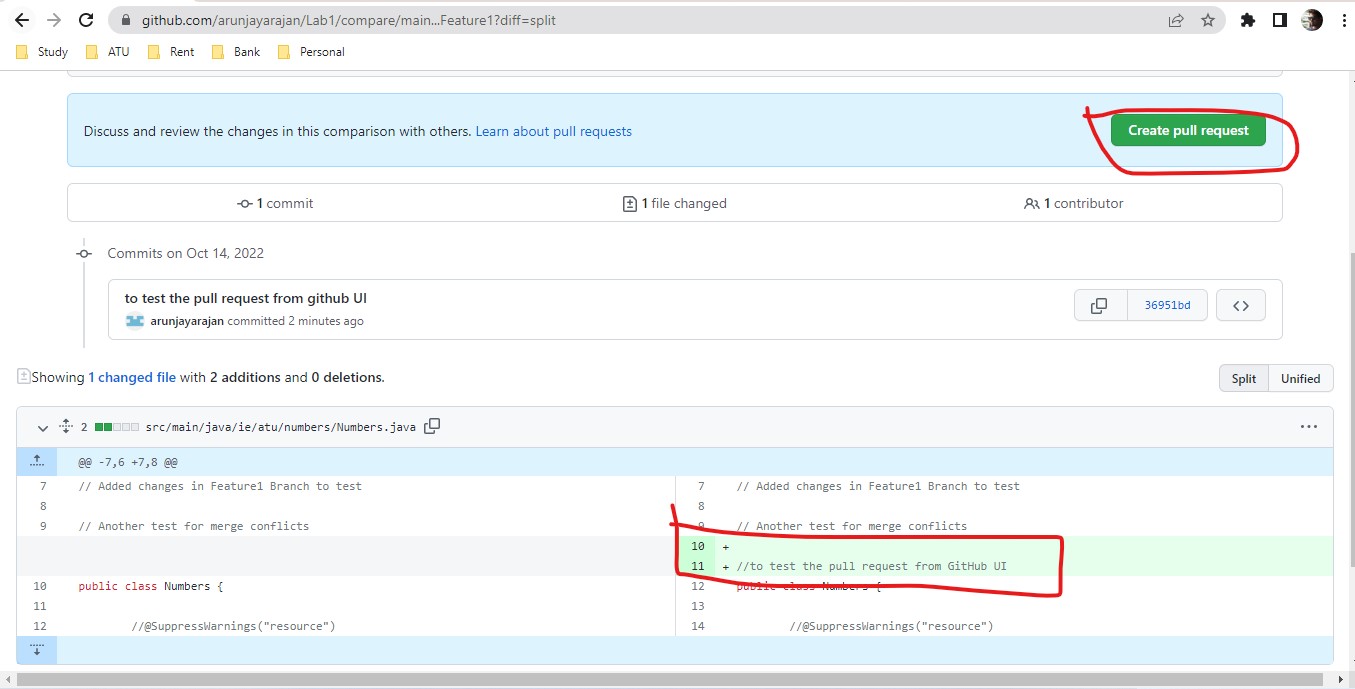


Figure 16: Merge conflicts in WebUI

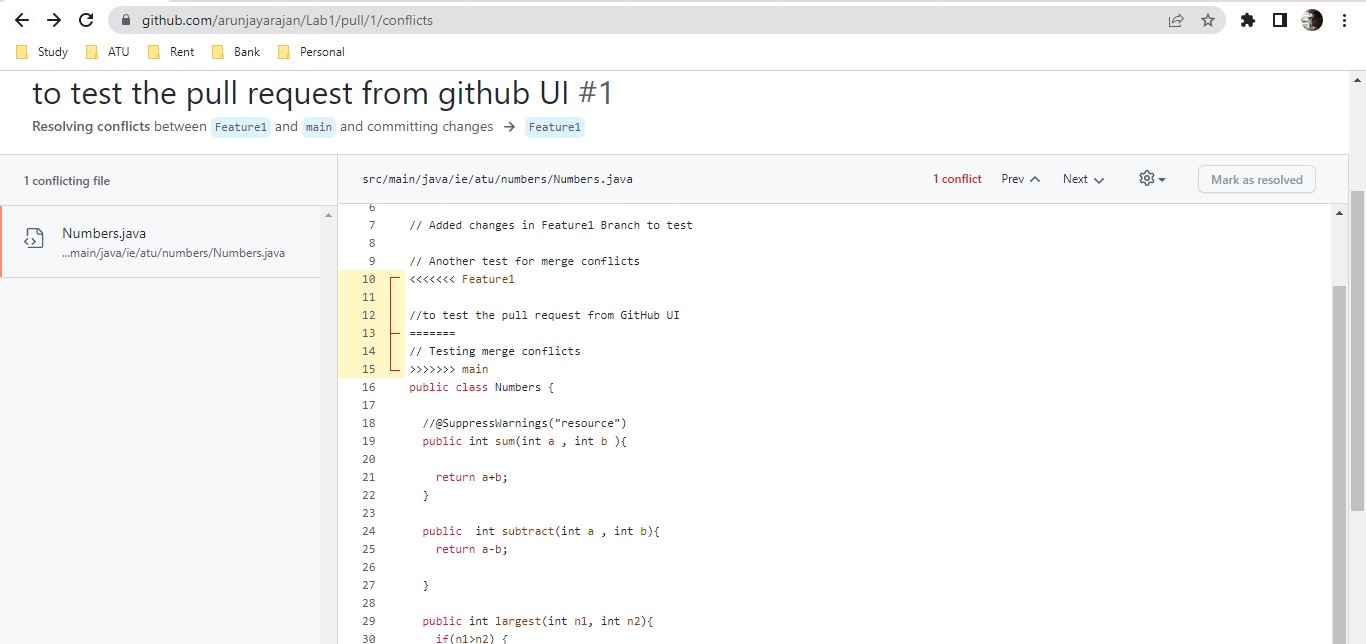


Figure 17: Conflicts resolved from WebUI



Figure 18: Pull request completed.

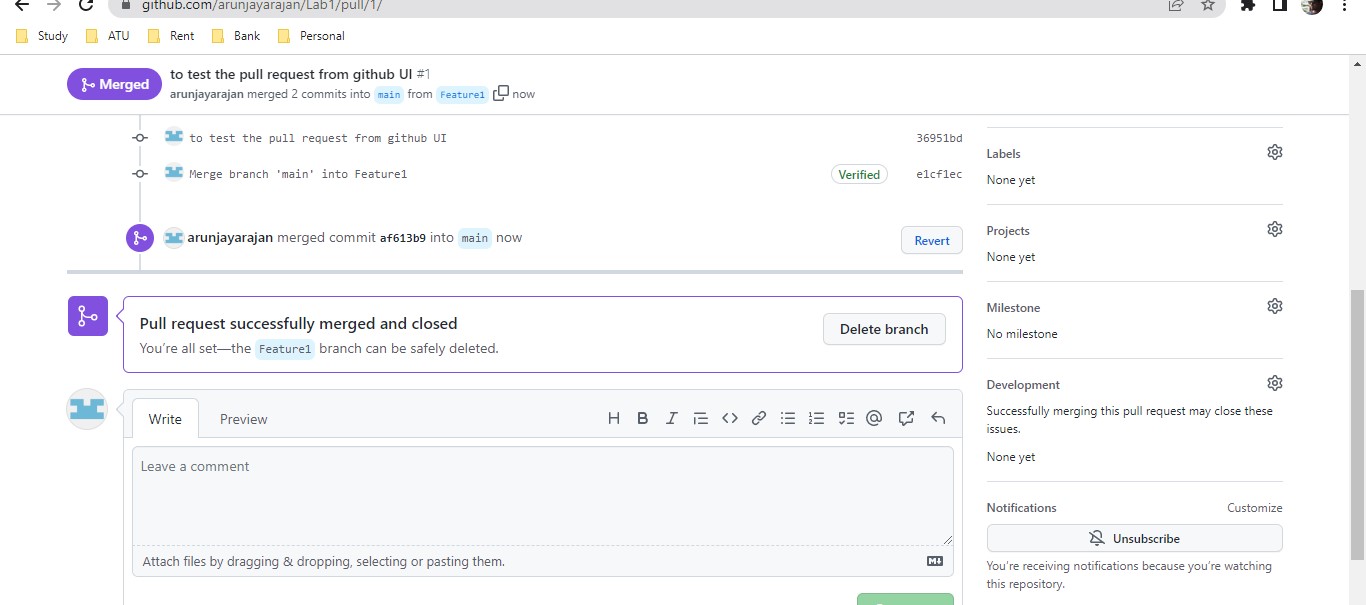


Figure19: Mandatory pull request for making changes to a specific branch.

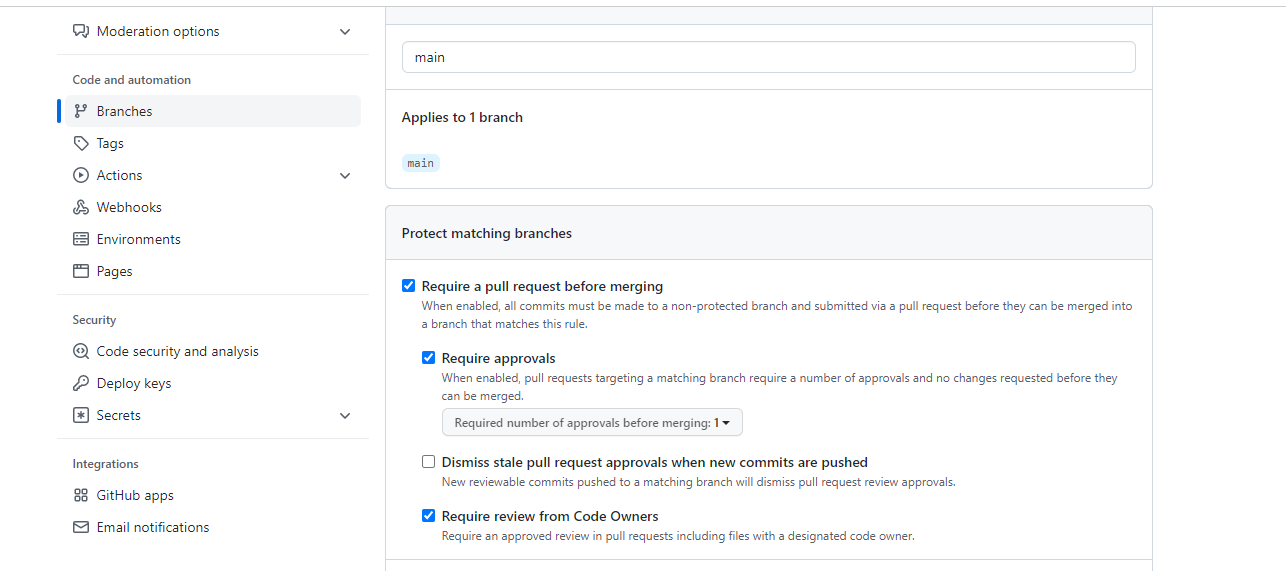


Figure 20: Cloned a different repository.

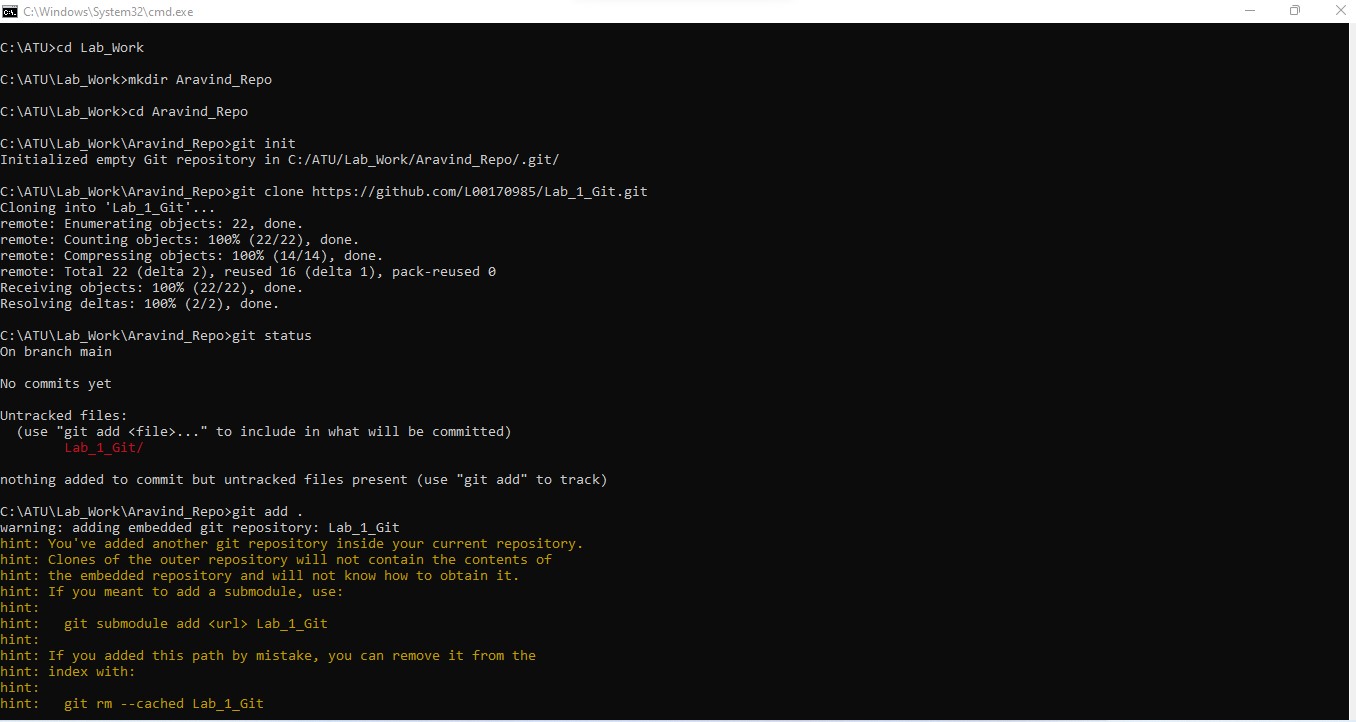


Figure 21: Made changes to a branch of a collegue’s repository and stashed the changes.

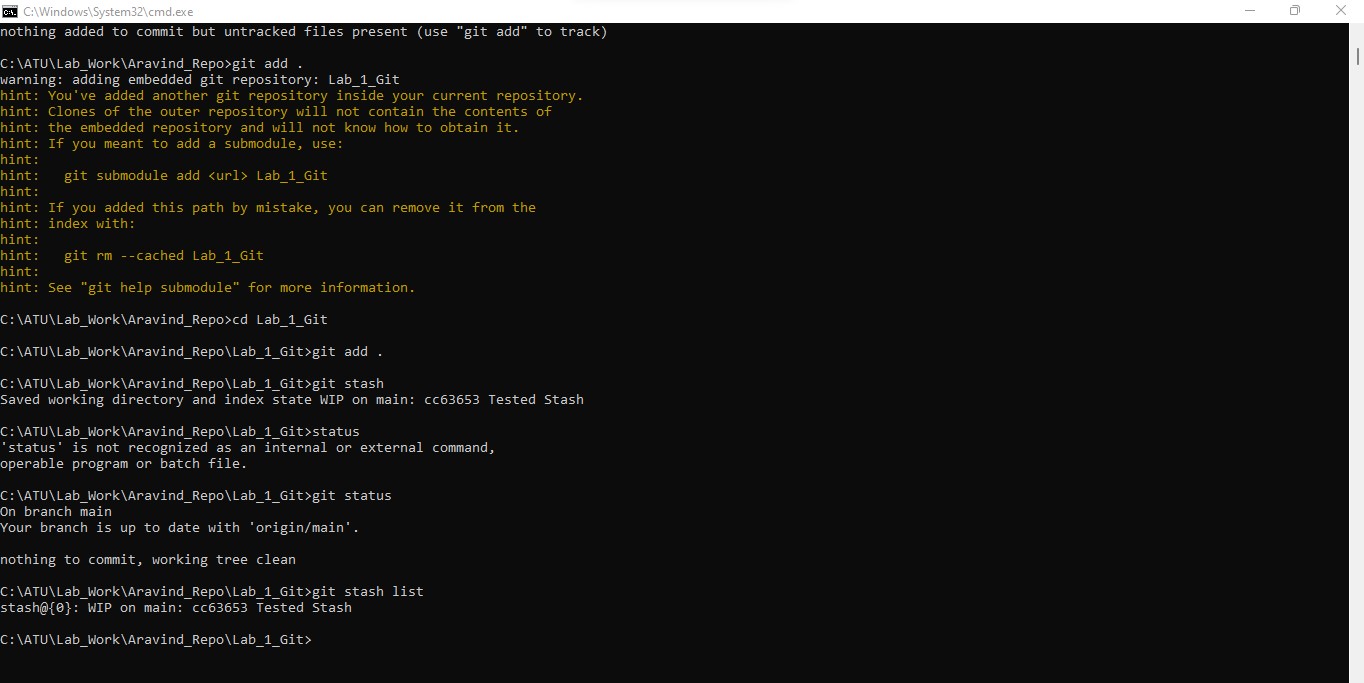


Figure 22:

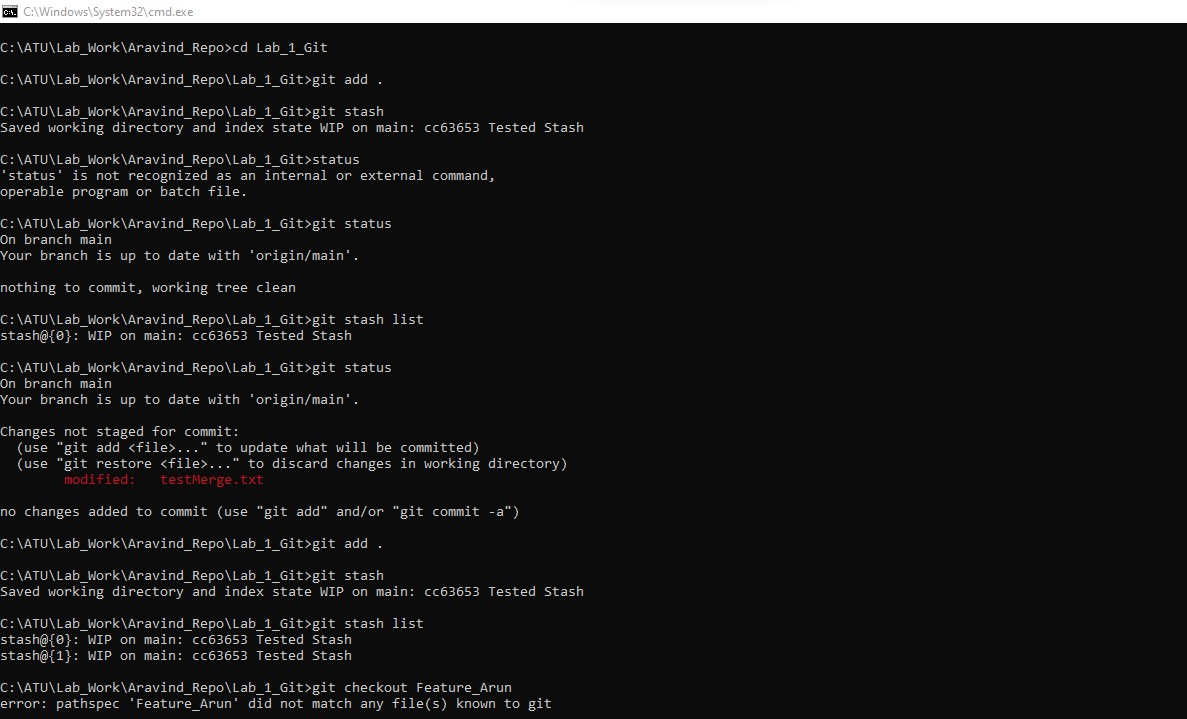


Figure 23: Did a stash pop and commited the code in the repo.

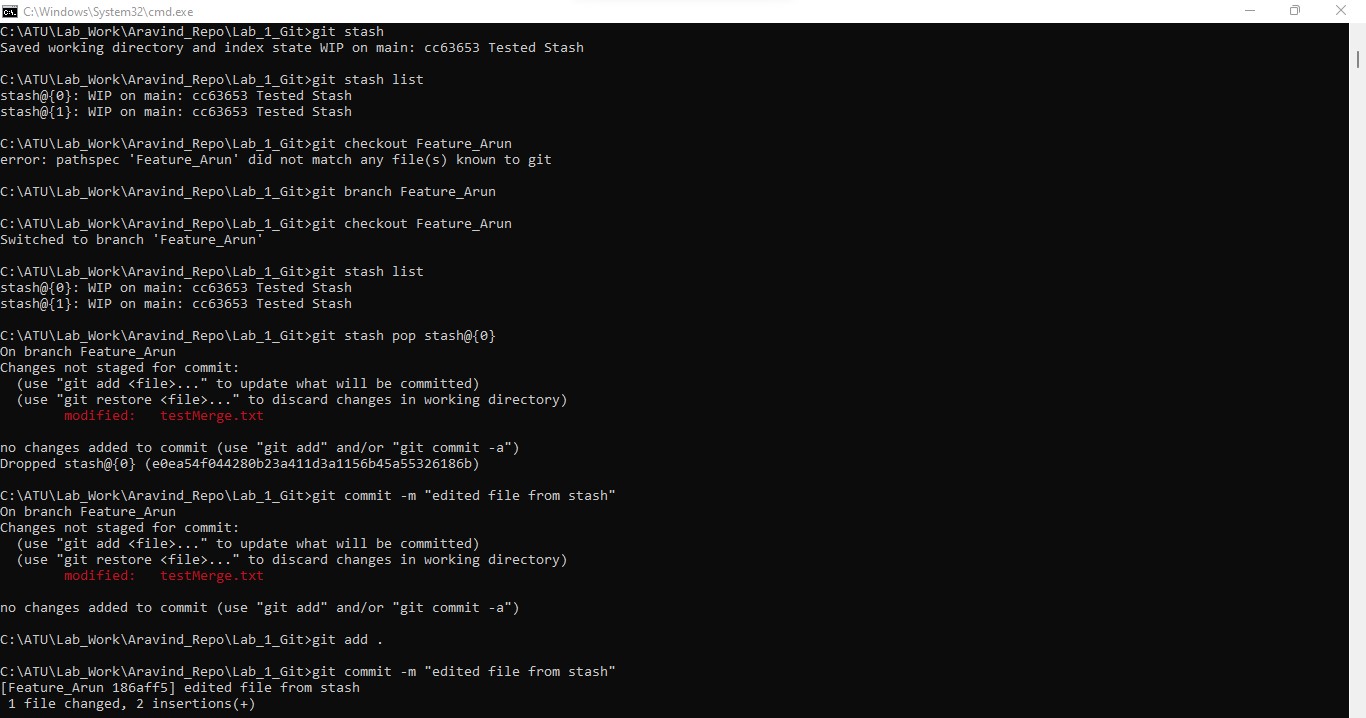


Figure 24:

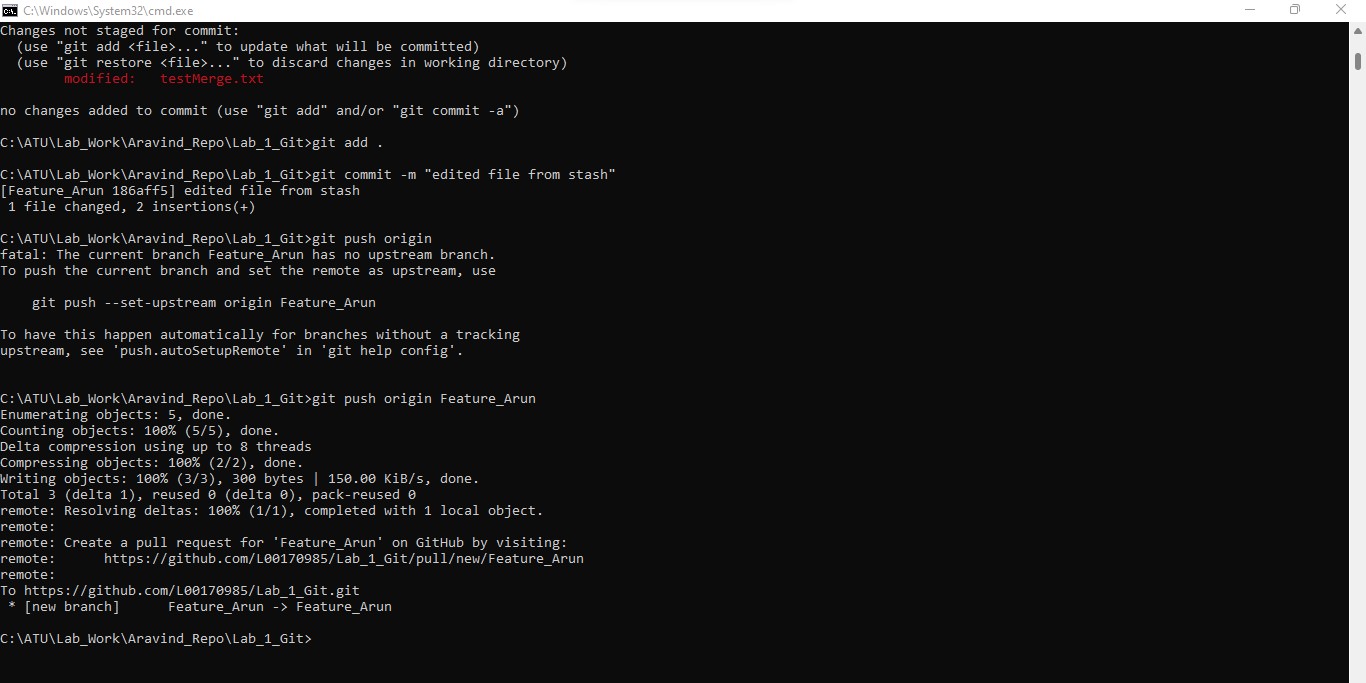


Figure 25: Pull request approval.

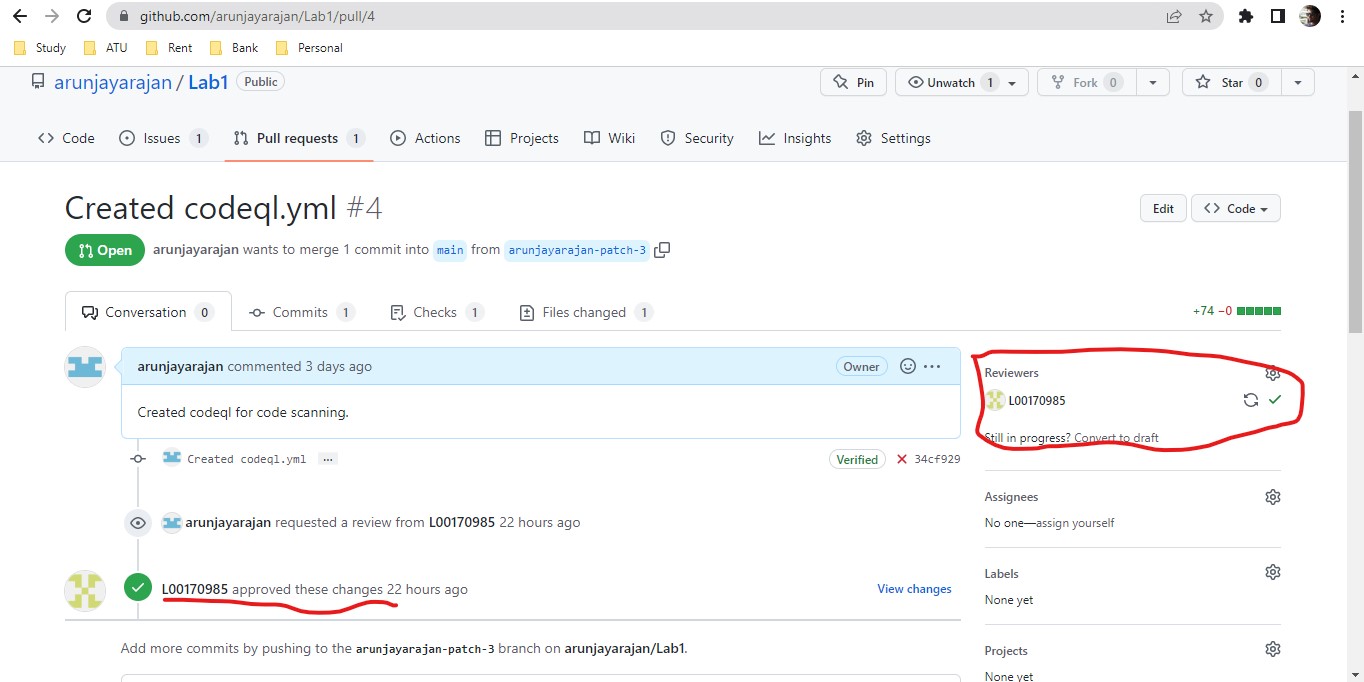


Figure 26: Merging of code complted after the pullrequest is approved.

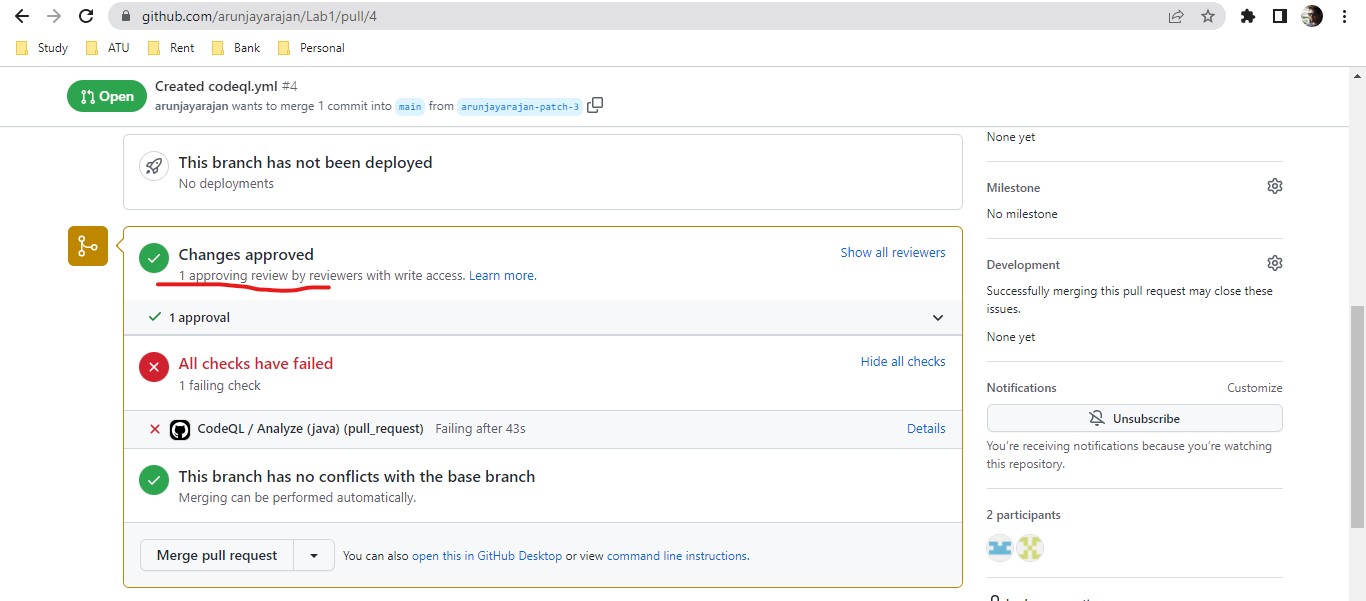


Figure 27: Added 3 commits as Test1, Test2, Test3 and did a git reflog command to listout the changes done.

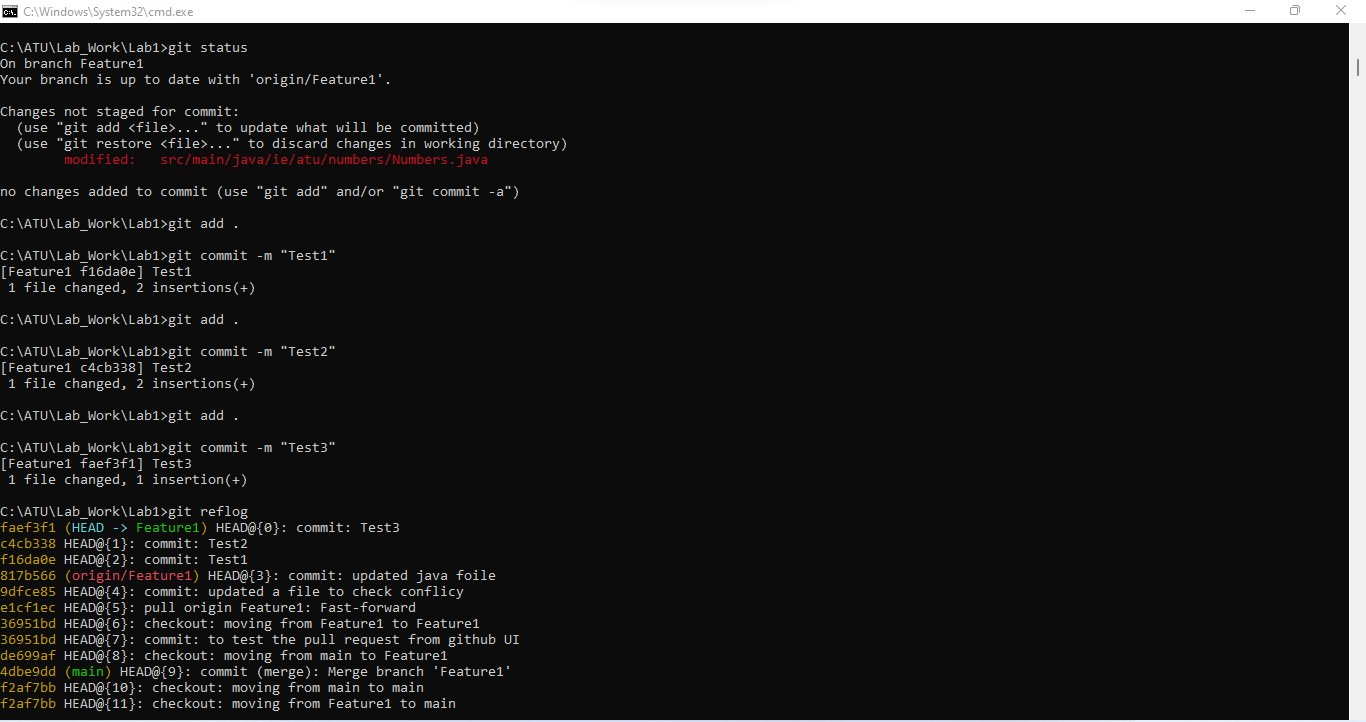


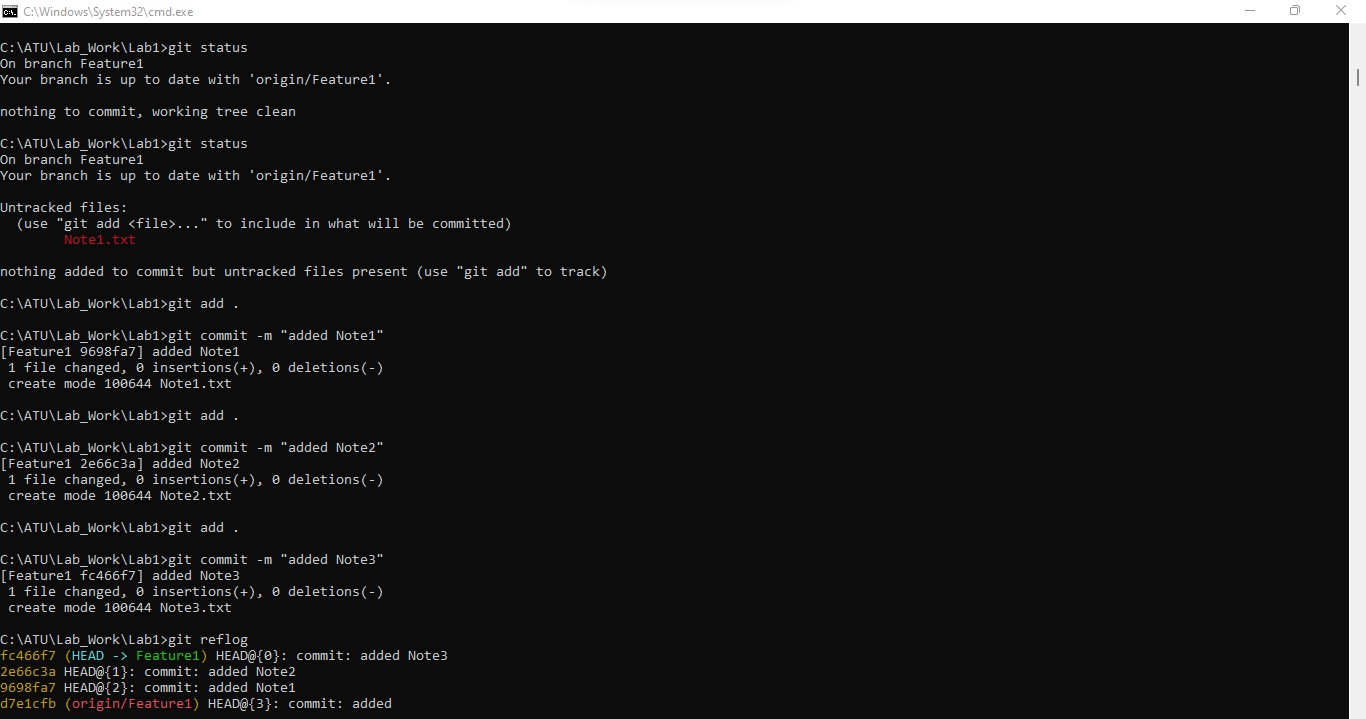
Figure 28: Added 3 Txt files.

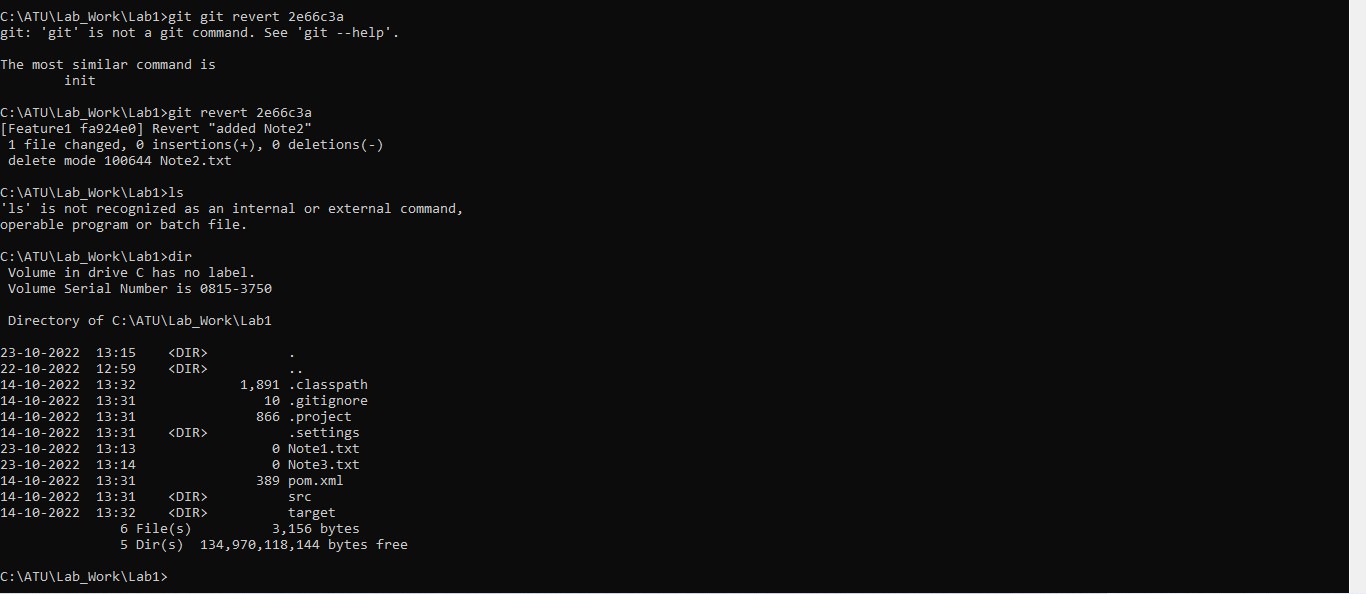
Figure 29: Did a git revert for Note2 and can see Note2.txt deleted from the repo. 

Figure 30: permissions to branch in Bitbucket.

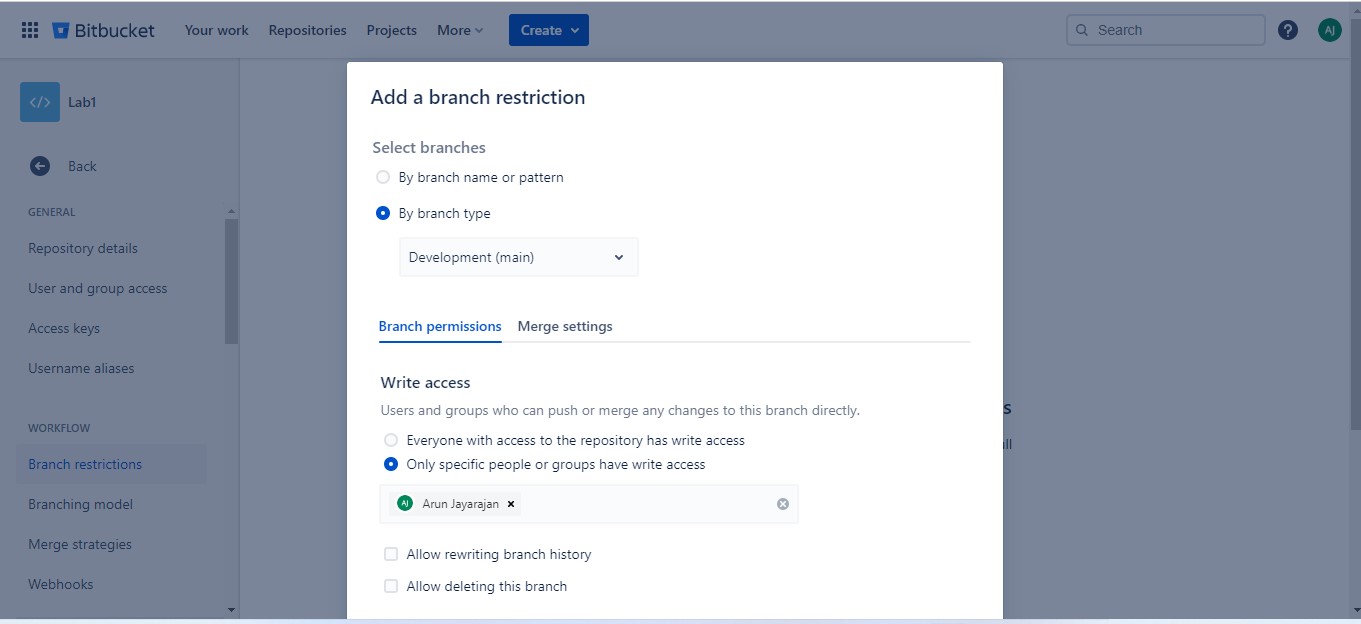


Figure 31: Integrated Snyk with Bitbucket.

