

**Table of Figures**

[Figure 1: Working on a branch with GitHub Desktop 1](#_Toc108194982)

[Figure 2: Album model insert and delete test 3](#_Toc108194983)

[Figure 3: Featured playlist model insert and update test 4](#_Toc108194984)

[Figure 4: Featured Playlist Song insert and delete test 5](#_Toc108194985)

[Figure 5: Like model insert and delete test 6](#_Toc108194986)

[Figure 6: Report insert and delete test 7](#_Toc108194987)

[Figure 7: All TDD tests completed successfully 8](#_Toc108194988)

[Figure 8: Edit Song class test 9](#_Toc108194989)

[Figure 9: Password Setting class test 10](#_Toc108194990)

[Figure 10: Sign Up class test 11](#_Toc108194991)

[Figure 11: Upload Song class test 12](#_Toc108194992)

[Figure 12: Upload Album class test 13](#_Toc108194993)

[Figure 13: All BDD test completed successfully 13](#_Toc108194994)

# **Git**

First of all, a GitHub repository was made and all team members were invited to the repository. The scrum master cloned the repository and pushed a new Flutter and NodeJS project on it using Git Bash. Then rest of the members cloned the repository and started working on their branches. Similarly, GitHub Desktop was used most of the time to create, fetch, pull, push branches, and create pull requests because GitHub Desktop allows those operations without manual coding on the terminal which made it very easy for every team member.

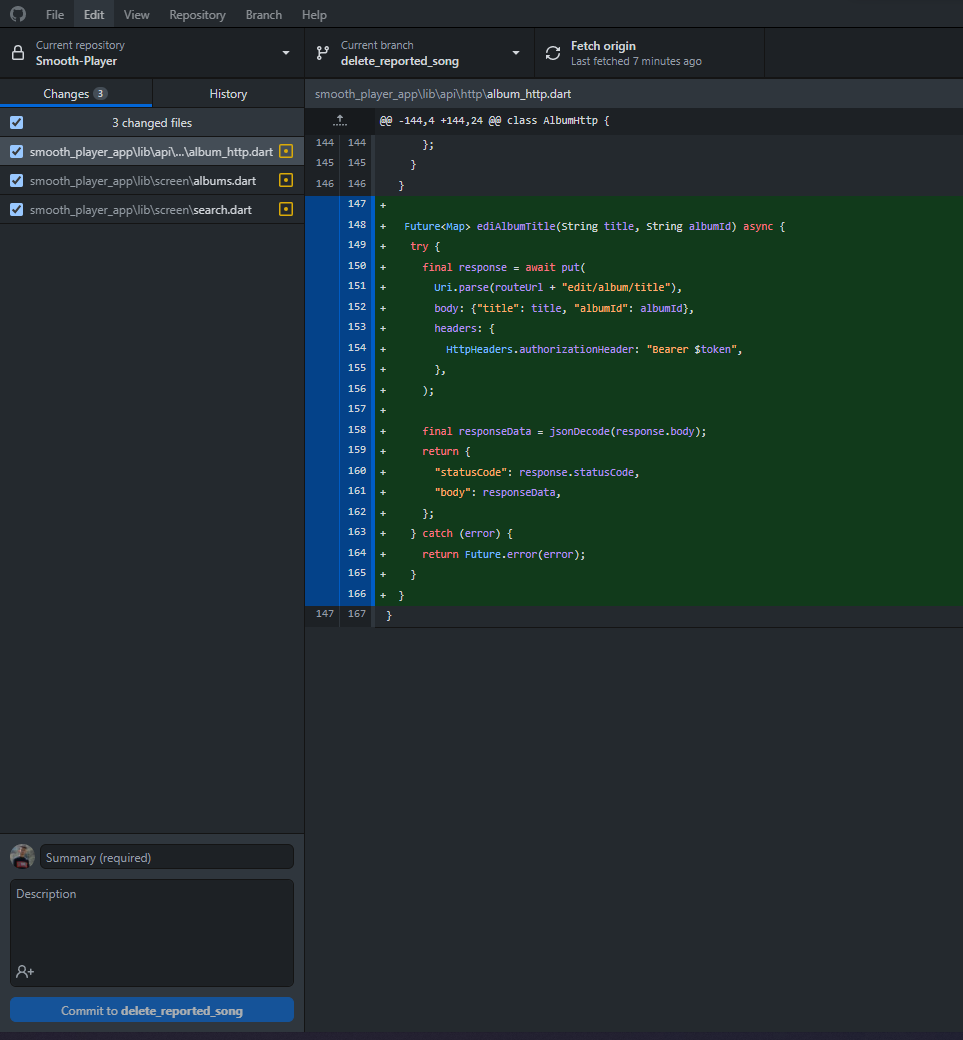


Figure 1: Working on a branch with GitHub Desktop

Git Bash was used only a few times. During each sprint, each team member created a new branch according to the user stories assigned to them and created a pull request to merge their new codes into the master branch. Team members except the scrum master were not allowed to edit and push the code on the master branch because it was the main branch and it should not contain any bugs or errors in the code. The pull requests were created after completing a feature or part of a feature so that other team members also can use the code and start working on their branch too whereas sometimes the pull request was also created at the end of the sprint. Lots of conflicts arose while merging branches on the master branch at the starting period. Then merging was done after resolving the conflicts. Every team member also started updating their branch from the master branch before merging and then no conflicts arose. On the other hand, a few times conflicts also arose while updating from the master branch. Other members could merge a branch into the master branch but required at least two approvals from the team members. Only the scrum master could merge the codes even without the approvals and most of the merging was done by him. No branches were deleted after completing the related feature. Working on multiple branches and merging were difficult during the initial sprints but after facing and resolving the problems related to it, then it was easy and no problems regarding conflicts arose. A brief description of the app and its features along with screenshots of the app's different pages were also provided on the GitHub repository.

# **Testing**

## **TDD**

For the TDD testing part, schema testing of the mongoose models was done. Inserting, deleting, and updating crud operations were done in the schema testing. The tests were done using the “jest” npm library. During the schema test, some of the tests failed because of expecting wrong data instead of the inserted data. The test failed because the inserted “id” would not exist in the database while testing update operation and also expecting only String for object id instead of converting the String into object id and then testing. The tests were done by all of the team members and the object id used on the test was only available in the database of the team member who did that test. This also caused problems because those tests failed on another team member’s project. Therefore, the ids were changed by the scrum master after all sprints were finished and testing of all models was done and all tests passed at last. Some screenshots of tests are shown below.

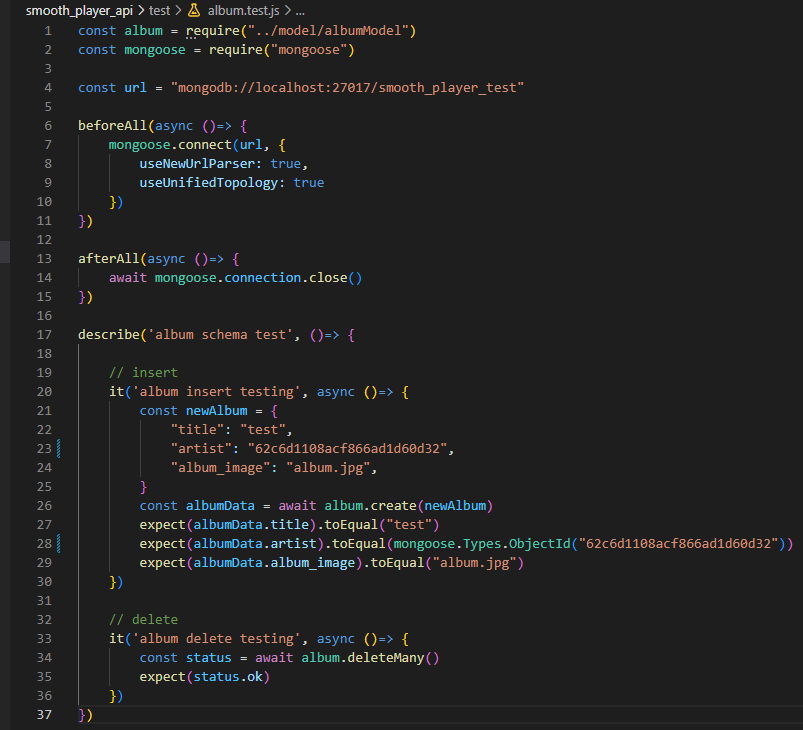


Figure 2: Album model insert and delete test



Figure 3: Featured playlist model insert and update test

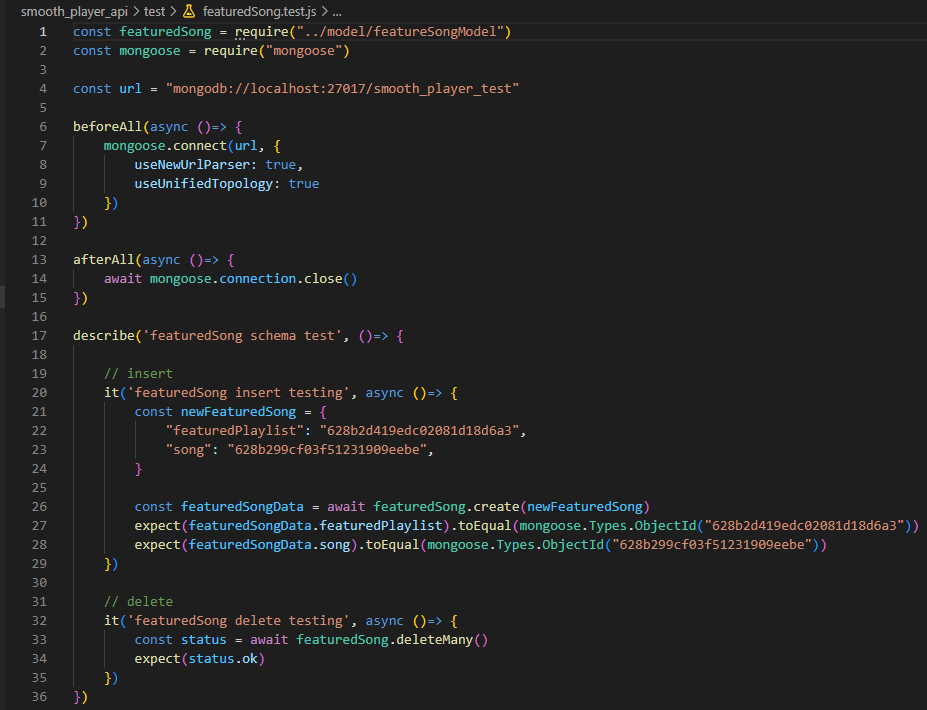


Figure 4: Featured Playlist Song insert and delete test



Figure 5: Like model insert and delete test



Figure 6: Report insert and delete test

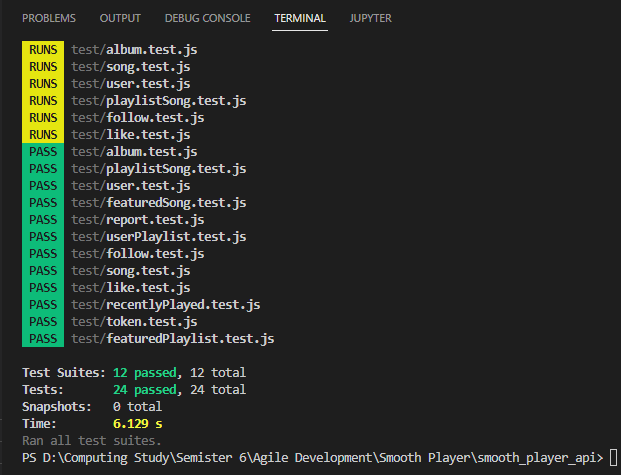


Figure 7: All TDD tests completed successfully

## **BDD**

Similarly, for the BDD testing part, widget testing of the flutter stateless and stateful classes were done. Flutter default testing package was used to test the classes. A single class was tested in a single test file and widgets like Text, ElevatedButton, Icon, Form, Column, SingleChildScrollView, and TextFormField were tested. During the testing period, the Http client errors were faced most of the time and most of the testing failed because of this. The widgets which contain Http classes or the classes containing Http classes in the initState function were failed. Therefore, only those widgets and classes which does not contain Http client errors were tested successfully. Most of the tested classes contain TextFormField and ElevatedButton. The tests were done by all of the team members and some already passed tests were failed at last. This was because some modifications were done on those tested classes until the last sprint. Therefore, the failed tests were also modified, and then the tests passed at last. Some screenshots of the tests are shown below.



Figure 8: Edit Song class test

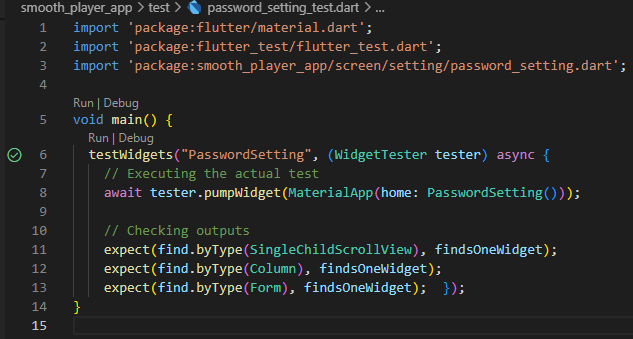


Figure 9: Password Setting class test



Figure 10: Sign Up class test

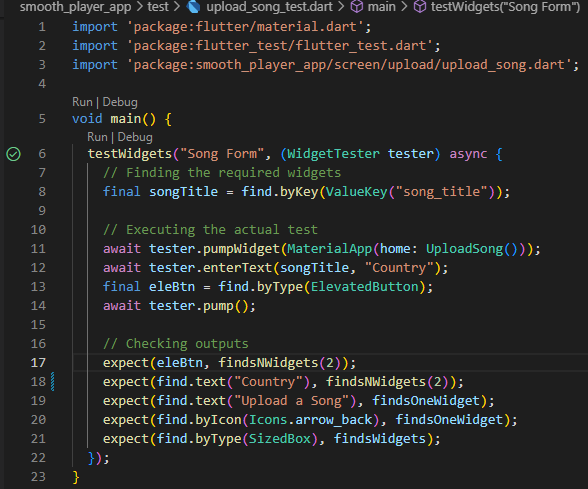


Figure 11: Upload Song class test

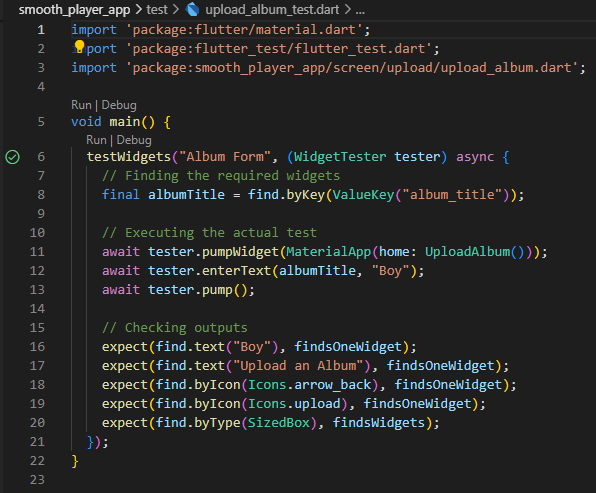


Figure 12: Upload Album class test

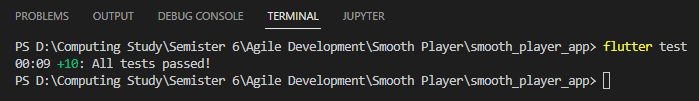


Figure 13: All BDD test completed successfully