Project 2

SNHU

CS-320

William Page

Summary

The testing approach that I used was aligned with the software requirements due to the testing and methods that were used in the main classes and the testing classes. Each milestone had unique requirements that had to be met and then tested. Within the first milestone, it was a requirement that you could be able to add and delete contacts so this was tested in the ContactServiceTesting class with Junit code as follows –

*@Test*

public void testAddContact() {

Contact contact = new Contact("12345", "Cam", "Green", "1234567890", "123 Main St"

contactService.addContact(contact);

*assertEquals*(contact, contactService.getContactById("12345"));

Since the requirements for the Contact milestone wanted the ability to add and delete contacts, I implemented some tests to make sure that the requirements were met and if there was an error within the code the test would flag the specified test case.

I made sure that my code was technically sound by implementing Junit test cases for each requirement thus making sure that the code was technically sound and that the requirements were met within the application. Just like the previous test case shown other test cases were implemented for each requirement to verify that the code was sound. Each Junit test that was implemented was specific to the requirement that it was testing also each set of testing was divided up into different classes that they were relevant to. To make sure that my code was efficient I only implemented the Junit test for the requirements that were provided not anything that was not related. This made sure that each class was relevant to the project and met the requirements.

Many testing methods were used in this project including black box testing as well as white box testing. With black box testing, we used the Junit test and it was implemented into the test cases so, this made sure that the test was testing the functions and performance of the code and ensuring that errors were found if they were present. If an error was found the set case was then flagged and could then be manually reviewed. I also used white box testing since I created the application and also implemented the test for the requirements of the application. Another method used was Functional testing due to the nature of the requirements and how the tests were implemented such as -

public void testAddTask() {

String taskId = "T123";

String name = "Example Task";

String description = "This is a sample task description.";

taskService.addTask(taskId, name, description);

*assertTrue*(taskService.taskMap.containsKey(taskId));

This test is an example of a functional test that was implemented in the milestone and project to test the specific requirements. Also, coding best practices were used to make all classes in the milestones go with the requirements and to make sure the test cases would run and check if the requirements were met.

The way I limited bias in the review of the code was by making sure that all requirements that were set were implemented in the testing even if they would fail. This ensured that the testing and review of the code would be accurate to the requirements of the project and not biased to make it look like all the testing cases were met within the application. Within all of my testing classes like ContactTest each test case that was implemented was to test each requirement to make sure the application met the requirements.

Quality is important in code because without quality applications would be full of bugs and vulnerabilities that could be exploited by certain individuals and lead to loss of data as well as money. This is why coding best practices should be used and every requirement should be tested within an application. Cutting corners can lead to errors down the road and cause other developers who work on your code to run into errors and bugs when using your code. Implementing coding techniques that have been learned in this course and previous courses will help me avoid technical debt on future assignments and projects. Implementing these learned techniques will improve my applications and make sure my code passes every test.

Sources

“Software Testing Techniques: Explained with Examples.” *BrowserStack*, 25 May 2023, [www.browserstack.com/guide/software-testing-techniques](http://www.browserstack.com/guide/software-testing-techniques).