Okemeta Olorungbemiga

CS-320 Project Two

Professor Travis Smith

Southern New Hampshire University

August 16, 2024

While developing software, the goal should be to make sure all requirements are being met and above all to satisfy the needs of the client, which also was one of the things I had in mind while developing this code. The requirement was to have three different services, the Contact Service, the Task Service, and the Appointment Service, all of which are required to have the ability to modify, add, delete, and update, meeting all the software requirements was a success because after running my software test, yes the code did run and the test passed, yes I was able to successfully pass the add, delete, and update all services, and the software was able to meet an overall coverage of 87% which shows me almost everything was covered accurately.

Ensuring the test worked was the major reason for the test, at the same time I had to consider simplicity, which meant writing the code in a way that would be easily readable, doing that I had to do things such as completing lines, e.g.

//getters

public String getID() {

return ID.

Something else I had to do was make it easy to read if I wanted to write a display text message due to an error or whatever. Eg

(throw new IllegalArgumentException("Invalid description")

Developing the code, I had to make sure it was efficient, I made sure to test the lines of the code and make sure the code worked accordingly per the request such as testing for each line of code expecting a positive and passed test.

public class TaskServiceTest {

@Test

public void testAdd() {

TaskService cs = new TaskService();

Task c1 = new Task("1", "Thomas", "Drakes");

Task c2 = new Task("2", "Grace", "Ghost Mode");

Task c3 = new Task("3", "Sutton", "Reimbursed");

assertEquals(true, cs.add(c1));

assertEquals(true, cs.add(c2));

assertEquals(true, cs.add(c3));

This developed software had a lot of requirements as discussed earlier, which are pretty much all sections of code being should have the ability to make modifications, such as additions, deleting, and making necessary updates, this was all I needed to realize how important the Junit testing would play a very important role in this development. What is a Junit test? This is a test automation framework for Java programming language specifically, used to do a unit test on a code. The Junit test was then implemented while utilizing the White-Box testing approach, this helped me keep track of how the code was working unit test by unit test, helped me check if there was any error because the code would fail to run, and I could track my mistakes to see what it was. To clarify, White Box testing is the type of testing that focuses on testing the internal structure of the code being utilized. Another form of testing that was implemented was Integration testing, this helped me check how different units of the code interact with each other and I was able to ensure the services are all working perfectly together.

There were a few other tests I could have implemented that I didn’t implement. One of which Is acceptance Testing, this is the form of testing that involves testing to see if it meets all users' needs with no defects. Another testing that wasn’t implemented would be performance testing, which is the type of testing done to check if the system can handle a particular load or task given to it. I didn’t implement this testing because I was not giving any test load requirements.

The techniques that Were discussed are generally used in software testing to ensure the efficiency of a code, Junit can be used by developers can track their code step by step to see which test ran or which failed and see why that happened, which practically would save time, and a lot of resources. Alongside the Integration, developers with multiple codes can see how well their code interacts during the testing phase coverage.

Limiting the complexity of the code was what I had in mind while developing the project, I was at the same time trying to provide the best result, which was vital. Employing caution was needed 100% Appreciating the complexity and interrelationships of the code I was testing was very crucial because this helped me when it came to how I needed the case to be designed to ensure all parts were covered.

I tried to limit bias by approaching each set of code with a different mindset of making sure everything gave the expected result with no defect whatsoever, on the software developer side, in this situation, if a developer was meant to test their code, Bias could be present, because a developer could overlook a minor error in the code as long as it gives the expected output.

As a developer, one of your core values should be discipline. Yes, it isn’t easy to spend hours developing software, or having to deal with bugs, issues, and all other stuff, but as a developer, you must be disciplined, because not only does this help promote professionalism, but it also helps save a lot of calamities. It is very important to never cut corners when it comes to writing or testing codes because as researched and seen across the globe, this could lead to a huge financial loss and setback depending on if cutting chase leads to damages or not which most time would lead to that. The Mariner 1, Chris Higgins (Sep 11, 2023) “**The incident was one of many early space launch failures, the bug was identified and fixed rapidly, though the failed launch cost $18,500,000 in 1962 dollars—north of $150 million today.**” The Mariner is a perfect example of how cutting corners and not maintaining discipline can cause a Spacecraft to crash due to incompetent code computation. In other to avoid technical debt, I plan on maintaining discipline while coding, and making sure constant reviews are being made for my code for it to meet all requirements.

References

<https://science.nasa.gov/mission/mariner-1/>

<https://www.mentalfloss.com/article/502943/day-1962-nasa-launched-and-destroyed-mariner-1>