Jackson Foster

CS-320-J7683 Software Test and Automation

Omar Toledo Lopez

April 15, 2023

Reflection

Unit testing is a critical aspect of software development that helps ensure the code is functional, reliable, and meets the software requirements. In this project, three Java classes were developed, and each feature was tested using a specific approach. The testing approach adopted for each feature was aligned with the software requirements and helped ensure that the code met the requirements.

My approach was designed to follow the requirements set by the client that I was given. From the very beginning, I read the requirements and made sure I based all my code to follow those requirements. These guidelines that were given were made to restrict the input from the user to make it easier to gather the correct information. For example one of the requirements asked to gather input from the user to get their phone number and they wanted exactly 10 digits. They asked for this for a couple of reasons, they wanted to make sure the user inputting information would give them their number including their area code. Without the area code if they were out of state they would not have the correct number. The requirements also stated that they didn't want to phone number section to be null. They would say this to make sure the user in putting their information would actually have to put in their number or it would throw an error and not let them finish putting their information in. This method wasn't made to deter users but more to get accurate information from the people putting their information in.

Due to these requirements, I wanted to make sure my code would stand up to them and make sure there were no flaws, and that users wouldn't accidentally break my code by putting in their information incorrectly. To do this I had to create JUnit tests to reflect on how the user might enter their information. On some tests, I would enter everything as it should be to make sure it would pass and if a user were to use this and do everything as expected they would actually be able to enter their information. On the rest of the tests, I tried to enter information in each field that was incorrect to make sure I could get the errors I was supposed to. On the name field, I would leave it null to make sure if the user forgot to enter their name it would throw an error and not let them continue. After creating these tests I would run a coverage test which would tell me if all my tests covered each branch my code had to find errors. For example, if I had an if statement that had a conditional or statement which means if either of these conditions fails to execute this branch I would have to run that test on both conditions to make sure it would catch on both of those to ensure I wrote the code correctly. If I wrote tests for each branch in my code I would get a coverage test of 100%, if I only covered most of the branches I would get a lower score. To make sure my code was sound I wanted to achieve a coverage test score of at least 80% which I did.

For my Software testing Techniques, I used two different methods JUnit as well as Manual review of my code. These two techniques combined really help me make sure my code ran efficiently and effectively. The JUnit tests were ways of inputting information as it should be inputted and some information that a user that doesn't know how the system works might put in their information. This is very effective because it will show flaws in your code and using these JUnit tests in combination with manual review makes the final product very clean and effective. I did not use integration testing that would allow me to test all the different classes together. I mainly didn't use this approach because I never got to the point where I would need to test them all together but this strategy would be very useful as this project continues on to create a main section of code having a user input their information and make sure they work as expected and don't interfere with each other.

The mindset I adopted during this project is that I was a developer working at a company and that I needed to write this code and people were counting on me to make the backend work while they worked on the front end and UI design. I wanted to make sure my code was one hundred percent functional before sending it to pre-production for extra testing. I had the mindset that I wouldn't be able to submit code that didn't work or had many flaws and that is why I made sure I had a lot of JUnit tests to ensure it worked correctly. I also felt like I had a boss giving me these requirements and that I had to make sure I followed them even if there were better methods to make this product better. For example, I was told to use the Date feature which is very outdated, and even the code editor I was using told me that the functions were deprecated and I had to suppress a warning for it. Although this was part of the assignment if this was the real world I would make sure not to use outdated libraries and functions and this would also lead to more effective code with fewer problems as seen within the appointment class.