Module Seven: Project Two

Cameron Lee

CS-320: Software, Test, Automation, & QA

Summary

Requirements are a crucial element in software development as they contain the purpose and goals of the product. So, it was important to make sure that the final project met the requirements set by each milestone rubric. In my project, each service had length requirements for ID’s, names, descriptions, phone numbers, etc. In my task service, for example, I had a requirement in my name and description setters that made sure that if the length was greater than the set amount or the variable was empty, it wouldn’t update the variable. This was shown on lines 40 through 50 in my task class. Another important requirement for this project was to make sure that a user couldn’t set an appointment date that was before the current date. This was an especially difficult requirement for me to meet as I had issues with obtaining the current date and reformatting it. But eventually, it is shown in my appointment class on lines 28 and 29, that the appointment date can not be null and the appt date must be after the current date, which was initialized in line 11. Ultimately, I think the overall quality of my JUnit tests were great. Using the two examples above, in the task service test and the appointment test, I used the assertions statements to equal true if, and only if, they met all the requirements set in the previously mentioned classes.

Looking at my JUnit tests specifically, there was a lot that went into making sure they were technically sound. An example of this would be on line 39 of my contact service test JUnit class. On this line, I am updating the contact to assert true and update all parameters set for a contact. All the variables were updated correctly with this line, and all the variables met the requirements set so that the true assertion would pass in the JUnit test run. Code can be technically sound but lack efficiency, so I made it another focus, like in all my projects, to make the code as efficient at its purpose as possible. A great example of this would be creating a global “appointment” in the appointment service test JUnit class on line 14. This allowed me to use that created appointment in all three of the tests without having to rewrite the appointment within each test. Ultimately, I’m proud of the outcome of this project considering where I started on initial milestones. It has taught me to stay fresh on my coding skills even when not in a coding class, and it has taught me the importance of utilizing JUnit tests to make sure code meets all requirements.

Reflection

The main software testing technique that I used during this project was a static analysis of my code. Static analysis is the process of examining code with the aim of identifying defects that may or may not cause failures (GeeksForGeeks, 2021). One specific example of my analysis was inspecting each class and removing any redundant code that I had in the project. This was done to make the code as clean as possible and prevent any potential issues from arising. One piece of redundant code that I had was with the date aspect of creating appointments. I had created an extra, unneeded date object to hold a future date. This happened as I was trying to format the date to a simpler format. A software testing technique that I would have used in a professional setting but didn’t on this project was peer review. I think that getting another set of eyes on a piece of code, and another person’s perspective, can sometimes catch errors, improve efficiency, or clean up pieces of code. I think each testing technique has practical uses in certain projects. I think a big contributor to the success of each technique relies on what the goals of the project are. Knowing those goals up front can help plan a way to effectively test the code.

My mindset on this project did contain caution, and at times was a little overwhelming. Specifically, the contact class being the first milestone had me overwhelmed as I felt rusty on my programming skills and struggled to find a direction to go in for the milestone. My mindset changed as I combined my three milestones together and felt like I had a solid foundation of code to work with and perfect. One of my greatest strengths is limiting my amount of bias when it comes to a perspective on something. I had no issues with bias when reviewing my own code several times throughout the project. I knew that I had errors in my code and points to improve on, so I can imagine that if I had a bias towards myself, I would gloss over a lot of the stuff that I took time to fix. A specific example was the issue of getting the tests to run on Eclipse through my PC versus through the virtual lab. I had no issues with running it on my PC, but there were issues on the professor’s side where it wouldn’t run. If I had a bias towards myself, and a mindset of “I know my program is perfect”, I wouldn’t have re-typed the entire project, copying from Eclipse to my virtual lab to export it and make sure there were no run issues. Ultimately, I think discipline is a critical virtue to have. Without discipline, in a professional setting, you can let other team members or a boss down on a big project. For a project that might contain sensitive information, it’s important not to cut corners because there could be a potential security leak that could release customer’s information to unwanted individuals. For a project like this, if I were to cut corners, I wouldn’t be contributing to the potential skills I hope to learn before entering the professional software workforce, therefore limiting my ability to succeed in my hopeful career.

**References**

@neeru360. (2021, March 1). *Software testing techniques*. Software Testing Techniques. Retrieved October 15, 2022, from https://www.geeksforgeeks.org/software-testing-techniques/