Caitlyn Murphy

CS 320

Journal 2

1. **Summary**
   1. Describe your unit testing approach for each of the three features.
      1. To what extent was your approach **aligned to the software requirements**? Support your claims with specific evidence.

**My testing was pretty aligned with the software requirements. I split each main requirements into a small, more manageable piece. Creating Junit tests for each class allowed me to cover the software requirements. Each class was tested for coverage. This included testing the getters and setters of base classes, and creating exception testing for the services.**

* + 1. Defend the overall quality of your JUnit tests. In other words, how do you know your JUnit tests were **effective** based on the coverage percentage?

**My Junit tests covered a considerable amount of my software. Each service had a coverage percentage between 70% and 90%. I tried to increase the coverage percentage of a few of my tests that were considerably low, as I seen them. I believe I still have a little ways to go before I understand how to get my coverage about 90% consistently, and I plan to work on that.**

* 1. Describe your experience writing the JUnit tests.
     1. How did you ensure that your code was **technically sound**? Cite specific lines of code from your tests to illustrate.

**To ensure my code was technically sound, I created code with increased readability. For the comments, I included what they were for and aligned them with the function I was working on. The ContactService file has examples of this on lines 11, 27, and 39. Also, I removed errors after each Junit check and ensured my test cases passed.**

* + 1. How did you ensure that your code was **efficient**? Cite specific lines of code from your tests to illustrate.

**To create efficient code, I only focused on what the requirements stated. I did not add any additional methods or classes and focused on what was necessary. In addition to this, I used the standard programming naming conventions, such as “testGetTaskAndUpdateSuccess” on line 23 of the TaskServiceTest file. This naming explains clearly what the method is intended for.**

1. **Reflection**
   1. Testing Techniques
      1. What were the **software testing techniques** that you employed in this project? Describe their characteristics using specific details.

**For my software development, I used two testing techniques: functional testing and unit testing. Functional testing determined whether each feature I created worked as the software required. Unit testing tested the small individual parts of each feature.**

* + 1. What are the **other software testing techniques** that you did not use for this project? Describe their characteristics using specific details.

**I chose not to use user acceptance testing as we didn’t have a full user group to evaluate the software with. User acceptance testing allows a small group of users to test the viability of the software created. As well, I chose not to use security testing. This testing would have given me information on how the application was affecting the device it was running on.**

* + 1. For each of the techniques you discussed, explain the **practical uses and implications** for different software development projects and situations.

**Functional and unit testing are pretty normal testing strategies for applications. Functional testing is typically completed by a quality assurance or secondary developer to determine if there were any bugs missed during development. Unit testing is extremely important, so a group of users can confirm that the application is working as they wanted and to check for any irregular use cases.**

* 1. Mindset
     1. Assess the mindset that you adopted working on this project. In acting as a software tester, to what extent did you employ **caution**? Why was it important to appreciate the complexity and interrelationships of the code you were testing? Provide specific examples to illustrate your claims.

**I was cautious of how I thought about the application. I tried to think like a user would instead of how a developer thinks. This involved me separating the times I created code and when I tested code. Both developers and testers have to have awareness as to how they approach the application.**

* + 1. Assess the ways you tried to limit **bias** in your review of the code. On the software developer side, can you imagine that bias would be a concern if you were responsible for testing your own code? Provide specific examples to illustrate your claims.

**As a developer, I tried to separate the times that I completed testing and the times I programmed. This allowed me to have two different headspaces going into the program. Bias would be an important factor in code. I have heard bias among many developers as their code is “perfect” when they test it, but is normally lacking in some way when others test it.**

* + 1. Finally, evaluate the importance of being **disciplined** in your commitment to quality as a software engineering professional. Why is it important not to cut corners when it comes to writing or testing code? How do you plan to avoid technical debt as a practitioner in the field? Provide specific examples to illustrate your claims.

**Cutting corners when writing or testing code can lead to unsuccessful, low security, and even dangerous applications. As a developer, writing quality code is a powerful tool that can improve or harm an users life. As a tester, testing thoroughly ensures that users are happy and prevent unintended consequences. Avoiding technical debt can be difficult, but I plan to stick to requirements written and ensure that my code is of good quality, so another developer can work on it right away, should it be needed.**