Robert Cook

CS-320 Software Test Automation & QA

Project Two

October 12, 2023

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 approach of unit testing aligned with software requirements by testing each individual requirement and any related edge cases. This allowed me to make sure that the code met the specified requirements of the software.**

* + 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?

**The quality of my JUnit tests is high due to the coverage of the code being one hundred percent. I also added @BeforeEach setup method to initialize for example “AppointmentService” before each test. This ensures that the tests start from a clean slate and do not interfere with each other.**

* 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.

**When writing my JUnit tests, I ensured my code was technically sound by adding comments throughout my code and using proper naming conventions. I used assertions to verify expected outcomes. For example on line 22 in AppointmentServiceTest testDeleteAppointment method an assertion checks if the size of the appointment list is zero after deleting an appointment.**

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

**Each test method is concise and focused on specific functionality. An example of this is the testAddContact method on line 17-21 in the ContactServiceTest.java. This verifies the behavior of adding a contact without including unrelated operations or assertions. This makes the tests efficient.**

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

**In Project One, I employed a blend of software testing techniques, incorporating both black-box and white-box testing strategies. Black-box testing revolves around inputting data and subsequently verifying whether the output aligns with the expected results. In contrast, white-box testing delves into the intricacies of the code, scrutinizing the logic for correctness. Additionally, I leveraged the power of edge cases to rigorously assess the software, deliberately targeting boundary conditions and challenging the system with unconventional scenarios. By adopting this multifaceted testing approach, I aimed to enhance the reliability and robustness of the software.**

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

**In Project One, I concentrated on specific software testing methodologies, omitting performance testing and security testing from the evaluation. Performance testing evaluates factors such as system speed, scalability, and responsiveness, helping identify potential bottlenecks and efficiency issues. On the other hand, security testing is instrumental in uncovering vulnerabilities and implementing protective measures against potential security risks. While these testing types were not explicitly employed in Project One, their significance in comprehensive software assessment remains evident.**

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

**The practical application of the software testing methodologies I employed lies in their ability to confirm the accuracy and effectiveness of the code. Neglecting certain software testing techniques can lead to consequences, including the risk of introducing errors and overlooking critical security vulnerabilities. By conducting comprehensive testing, we can mitigate these potential issues and enhance the overall quality and reliability of the software.**

* 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 recognized that software can be intricate, involving multiple components and interactions. As such, I took the time to thoroughly understand the codebase, considering how different parts of the code connected with each other. I was diligent in creating and executing test cases to cover various scenarios and potential edge cases. This aimed to identify any unexpected behaviors or issues within the software. I implemented unit testing as a crucial part of the testing process. This allows for the isolation of specific code components which enables in-depth assessment of their functionality. For example, I created unit tests to scrutinize critical functions or methods within the code, ensuring that they worked as intended.**

* + 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.

**To ensure an objective assessment, I proactively integrated unit tests into my workflow, thereby demonstrating thoroughness in the testing process. It is of paramount importance for software developers to maintain awareness of the potential for bias and to uphold a rigorous commitment to testing their code comprehensively.**

* + 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.

**Maintaining a strong dedication to code quality stands as a fundamental pillar in the software development journey. The significance of steering clear from hasty shortcuts during code creation and testing cannot be understated, as it serves to prevent the accrual of technical debt. In pursuit of this objective, I placed considerable reliance on unit tests, employing them as a robust means to scrutinize the code's structural soundness and guarantee its error-free state.**