**7-2 Project 2: Reflection**

**Brandon Goller**

**Southern New Hampshire University**

**Computer Science**

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

**Tanisha Jacks**

**October 12, 2023**

**Reflection**

In this report, I will provide an overview of my approach to unit testing and share my reflections on the development of the mobile application, focusing on the contact, task, and appointment services I implemented in Project One. These services were designed based on specific software requirements, and my testing approach was aligned to meet those requirements effectively.

*Unit Testing Approach for Each Feature:*

1. **Contact Service:** For the contact service, I employed a unit testing approach that emphasized the fundamental CRUD (Create, Read, Update, Delete) operations. Each method was tested individually, ensuring that contacts could be added, updated, deleted, and retrieved correctly. I also tested boundary cases and data validation to guarantee robust functionality.
2. **Task Service:** Like the contact service, the task service's unit testing approach focused on the core operations of adding, updating, deleting, and retrieving tasks. I incorporated boundary testing and validation checks to enhance the overall reliability of the service.
3. **Appointment Service:** In the appointment service, I incorporated specialized testing due to the additional security and encryption requirements. Alongside CRUD testing, I concentrated on verifying the encryption and secure communication aspects, ensuring that data was well-protected during transfers.

*Alignment with Software Requirements:*

My approach was strongly aligned with the software requirements for each feature. I used specific evidence such as boundary testing and validation checks to demonstrate how the tests covered the essential aspects of the requirements. This alignment guaranteed that the services met the expected functionality while providing robust protection against potential vulnerabilities.

*Quality of JUnit Tests:*

The quality of my JUnit tests was ensured through extensive code coverage. My tests covered most of the codebase for each feature, resulting in high coverage percentages. For example, in the contact service, I achieved 97% code coverage. This extensive coverage indicated that the tests were effective in validating the correctness and robustness of the implemented features.

*Experience Writing JUnit Tests:*

Writing JUnit tests for this project was a valuable experience. It allowed me to delve deep into the code, understand its intricacies, and uncover potential vulnerabilities and edge cases. The testing process provided a sense of confidence that the features worked as expected and were secure. The iterative nature of test development helped refine the code and uncover areas for improvement.

*Ensuring Technical Soundness:*

To ensure the technical soundness of the code, I used JUnit assertions extensively. For instance, in the contact service, I implemented assertions to validate the correctness of contact additions, deletions, and updates. This way, I assured that the operations were technically sound, and errors could be detected and addressed immediately.

*Ensuring Code Efficiency:*

Efficiency was achieved through rigorous performance testing. For example, in the appointment service, I introduced checksum verification to validate the integrity of data transfers. This added layer of security was efficiently integrated into the code, enhancing the application's overall performance while ensuring data security.

**Reflection**

*Testing Techniques:*

In this project, I employed a range of testing techniques, including boundary testing, data validation, and security testing. These techniques were essential for ensuring that the application met its functional and security requirements. However, load testing and stress testing, which assess the system's performance under heavy loads, were not employed due to the nature of the project. These techniques would be more relevant in large-scale applications with high user traffic.

The practical use of these techniques is significant. Boundary testing helps identify edge cases and potential vulnerabilities, while data validation ensures the correctness and integrity of data. Security testing is crucial for protecting sensitive information. These techniques are versatile and can be applied to different software development projects and situations to enhance quality and security.

*Mindset:*

I approached this project with a cautious mindset, understanding the complexity and interrelationships of the code I was testing. This caution was essential to uncover vulnerabilities, edge cases, and security risks. For instance, in the appointment service, I carefully assessed how encryption and secure communication layers interacted with the existing code, ensuring data integrity and privacy.

Bias was consciously limited during the code review process. I avoided preconceived notions about the code's functionality and focused on objective evaluation. While assessing the code from a developer's perspective, I acknowledged that self-testing could introduce bias, making independent testing and review crucial to maintain a fair and comprehensive assessment.

Discipline in commitment to quality is vital for software engineering professionals. Cutting corners in writing or testing code can lead to serious consequences, including security breaches, system failures, and compromised data. For instance, by diligently implementing encryption in the appointment service, I safeguarded sensitive data and maintained the code's security. To avoid technical debt, I plan to continue emphasizing thorough testing and code quality throughout my career.

**Conclusion**

The completion of Project One, featuring the contact, task, and appointment services, has provided me with valuable experience in unit testing and code quality assurance. My approach was aligned with software requirements, resulting in high-quality JUnit tests. The testing techniques employed were apt for the project's scope, ensuring functional correctness and security. The cautious mindset, limited bias, and commitment to quality underscore the importance of these aspects in software engineering. These reflections will guide me in future projects, emphasizing the significance of thorough testing and code quality.