**Project Two**

Makayla Meeks

CS 320 Software Test, Automation QA

K. Wilson

February 21, 2025

As a software engineer at Grand Strand Systems, I recently completed a project focused on developing a mobile application for contact, task, and appointment services. This report highlights my experience in unit testing, particularly through the implementation of JUnit tests, and reflects on both the techniques used and the mindset I adopted throughout the project.

My approach to unit testing was centered around meeting the functional requirements of the project, ensuring that each of the three core features of the mobile application—contact management, task management, and appointment scheduling—performed as expected. For each feature, I wrote multiple JUnit tests to verify the functionality under various scenarios, including both valid and invalid inputs. This approach aligned closely with the software requirements, as demonstrated by the comprehensive test coverage I achieved. To quantify effectiveness, I tracked the coverage percentage, which confirmed that the majority of the application code was tested. This high coverage ensured that critical paths and edge cases were accounted for, reducing the likelihood of undetected bugs.

In terms of test quality, I focused on both technical soundness and efficiency. To maintain technical soundness, I adhered to best practices such as clear assertions and method isolation, which minimized dependencies and promoted reliability. For example, in one test case for the task management feature, I ensured that the test accurately validated the proper status change when a task was marked as complete. Regarding efficiency, I optimized the structure of my tests by avoiding redundant logic and ensuring concise code. An example of this efficiency is seen in the way I utilized reusable methods for common validation checks.

During the project, I employed several software testing techniques to ensure robust validation of the application. The primary techniques used included boundary value analysis, equivalence partitioning, and test-driven development (TDD). Boundary value analysis helped identify edge cases where user inputs might break the system, while equivalence partitioning reduced the overall number of tests by grouping similar inputs. TDD allowed me to continuously refine my code based on immediate feedback from failing tests. However, I did not implement load testing or integration testing due to time constraints and the scope of the project. These omitted techniques would have been useful for testing scalability and interactions with external systems.

Practical implications of the techniques I employed include improved reliability and maintainability of the software. Boundary value analysis, for example, helped prevent errors from occurring at input extremes, which is crucial in real-world scenarios where data variability is high. TDD reinforced the importance of iterative improvements, enabling a cycle of rapid development and correction.

Mindset played a crucial role in my testing process. I adopted a mindset of caution by thoroughly analyzing the interdependencies between various components of the application. This approach helped me appreciate the complexity of the system and ensure comprehensive testing. Limiting bias was another key priority. Since I was responsible for both developing and testing my code, I sought feedback from peers and relied on automated testing tools to detect issues that I might have overlooked. Lastly, I maintained discipline in my commitment to quality, recognizing the long-term benefits of preventing technical debt. By adhering to coding standards and avoiding shortcuts, I reduced the risk of future maintenance challenges.

In conclusion, this project underscored the critical role of thorough unit testing in delivering reliable and maintainable software. The techniques and mindset I applied throughout the project have strengthened my approach to software engineering, reinforcing the importance of careful planning and continuous improvement in both development and testing.

**References**

Hambling, B., Morgan, P., Samaroo, A., Thompson, G., & Williams, P. (2019). *Software testing : An istqb-bcs certified tester foundation guide - 4th edition*. BCS Learning & Development Limited.