**CS-320-R3314: 7-2 Project Two**

Pierrot Ngimbidi

Southern New Hampshire University

**CS-320-R3314**

Angelo Luo, M.S

02/24/2024

**7-2 Project Two**

In the evolving software development landscape, ensuring back-end services' quality and reliability is paramount. This summary and reflections report outlines the meticulous unit testing approach adopted for developing a mobile application, focusing on contact, task, and appointment services. It provides insights into the alignment of the testing approach with software requirements, the effectiveness and efficiency of the JUnit tests written, and the broader testing techniques and mindsets employed during the project's lifecycle. This reflective analysis serves as a testament to the rigor applied in ensuring software quality and as a guide for future endeavors in software testing and development.

Our unit testing approach was methodical, focusing on isolating each piece of functionality within these services. We employed JUnit tests to ensure that individual components behaved as expected across various inputs and conditions. This approach was meticulously aligned with the software requirements, ensuring that every functional and non-functional requirement was tested through specific test cases.  
Our testing strategy was closely aligned with the software requirements, ensuring comprehensive coverage of all features. For example, in testing the Contact service, we ensured that all fields (name, email, and address) met the specified criteria, using assertions like assertNotNull(contact) and assertEquals ("Expected Name", contact.getName()). These assertions reflect the requirement for all contact fields to be correctly populated and validated.

The effectiveness of our JUnit tests was evidenced by a high coverage percentage, indicating that tests exercised a significant portion of the codebase. This thorough coverage, coupled with the absence of defects in delivered features, attests to the quality of our tests. We used tools like Eclipse to measure coverage and ensure no critical paths were untested.

Writing JUnit tests was both challenging and enlightening. It required a deep understanding of the codebase and the intended behavior of each component. Ensuring technical soundness involved rigorous testing of both expected outcomes and edge cases, using lines of code such as assertThrows(IllegalArgumentException.class, () -> new Contact(null, null, "Address")) to validate input constraints.  
Efficiency was paramount in our testing strategy. We used annotations like @BeforeEach to reset the state before each test, ensuring that tests were isolated and independent. This approach minimized interference between tests and allowed for quick identification of issues.

Our project primarily utilized unit testing and code coverage analysis. Unit testing allowed us to validate individual code units, ensuring they performed correctly in isolation. Code coverage analysis helped us quantify how much of the codebase was tested, guiding us toward achieving comprehensive test coverage.  
Techniques not employed in this project include integration, system, and acceptance testing. Integration testing would have allowed us to verify the interactions between different components, while system testing and acceptance testing would have ensured the application met overall requirements and user expectations.  
Each testing technique has its place in a software development project. For instance, unit testing is crucial for early detection of issues at a granular level, while integration testing is essential for ensuring that components work together as expected. System and acceptance testing are critical for validating the end-to-end functionality and usability of the application.

Adopting a cautious mindset was essential, especially given the complexity and interconnectedness of the application's components. This approach was vital in appreciating the subtle ways in which code changes could introduce defects. To limit bias, especially when reviewing my code, I adopted practices such as peer reviews and automated testing tools to ensure objectivity.  
Bias is a significant concern when developers test their code, as familiarity with the codebase can lead to assumptions about its behavior. Implementing a rigorous review process and employing automated testing tools helped mitigate this risk.

The discipline in maintaining a high commitment to quality was non-negotiable. Cutting corners in testing can lead to technical debt, increased maintenance costs, and diminished user satisfaction. Moving forward, my plan to avoid technical debt includes adhering to testing best practices, continuous learning, and integrating feedback from all stages of development to refine testing strategies.

Reflecting on the journey of developing and testing the mobile application for Grand Strand Systems, it's evident that a methodical and disciplined approach to unit testing is indispensable for delivering software of the highest quality. While challenging, the experience of writing Junit tests has been profoundly rewarding, providing deeper insights into the importance of each line of code and its impact on the overall system. Adopting various software testing techniques, combined with a cautious and unbiased mindset, has been instrumental in navigating the complexities of software development. This project underscores the importance of meeting and exceeding software requirements through meticulous testing, a commitment to quality, and a continuous pursuit of excellence. As we move forward, these experiences will serve as a cornerstone for future projects, guiding our software development and testing approach with an unwavering commitment to quality, efficiency, and reliability.

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.

Boni Garcia. (2017). *Mastering Software Testing with JUnit 5 : A Comprehensive, Hands-on Guide on Unit Testing Framework for Java Programming Language*. Packt Publishing.