Alfredo Romero

CS 330 Software Test Automation

SNHU

06/18/2023

# 7-2 Project Two Submission

Throughout the development of the mobile application for our customer, I diligently followed a well-structured unit testing approach for the contact, task, and appointment services. Each feature was thoroughly tested to ensure alignment with the software requirements, and the quality of our JUnit tests was upheld through comprehensive coverage and adherence to best practices.

For the **contact service**, our testing approach focused on creating, updating, and retrieving contacts. Test cases were designed to validate these operations, such as creating a contact with accurate information and verifying its proper storage in the database. In the **task service**, our emphasis was on testing the creation, updating, and deletion of tasks. Various scenarios were covered, including tasks with different priorities, updates to task details, and the accurate removal of deleted tasks from the system. Similarly, the **appointment service** was meticulously tested for its creation, retrieval, and cancellation functionalities. Test cases were designed to ensure the handling of appointments with different time slots, proper updates, and appropriate cancellation procedures.

To maintain the overall quality of our JUnit tests, we employed several strategies. Comprehensive coverage was achieved by designing test cases that encompassed different functionalities and scenarios within each feature. Regularly monitoring code coverage reports allowed us to identify any gaps and improve test coverage as needed.

In addition, we adhered to established unit testing best practices, writing testable code, employing suitable assertions, and minimizing dependencies on external systems. By keeping our test cases focused and isolated, testing only specific functionalities or scenarios, we ensured the effectiveness of our JUnit tests. The experience of writing JUnit tests was invaluable, allowing us to validate the correctness and robustness of our code. Following a systematic approach, we began with a deep understanding of the requirements, designing precise test cases, and implementing them using JUnit. Our test descriptions were clear and concise, facilitating understanding for other team members. To ensure technical soundness, we paid meticulous attention to assertions and verification steps within our tests. For example, in the contact service, we included assertions to validate the properties and values of retrieved contacts, affirming the proper execution of contact-related operations. Efficiency was also a key consideration in our test implementation. We strived to maintain concise and focused test code, avoiding unnecessary setup or redundant assertions. This streamlined approach ensured efficient execution of the tests, delivering results promptly and effectively. For instance, in the task service, we only included relevant assertions specific to the task being tested, eliminating any superfluous verification steps.

# Reflection

## Testing Techniques:

Throughout this project, we employed a variety of software testing techniques to ensure the high quality of our mobile application. One of the techniques we utilized was black-box testing, where we focused on inputs and outputs without considering the internal code structure. This allowed us to test the application externally, ensuring that the features behaved as expected and aligned with the requirements. Additionally, we applied boundary value analysis to thoroughly test the application with inputs at the boundaries of valid ranges. This technique helped us identify any issues related to data validation or boundary conditions. For example, when creating appointments, we tested for both valid and invalid dates and times to ensure flawless handling of edge cases. We also employed equivalence partitioning, which involved dividing input data into equivalence classes to streamline testing. By selecting representative test cases from each class, we ensured comprehensive coverage while minimizing redundancy and maximizing efficiency.

While these techniques were crucial for our project's success, there are other valuable software testing techniques that we did not employ. For instance, exploratory testing could have provided valuable insights by allowing testers to interact with the application in an unscripted manner, uncovering potential issues and usability concerns. Usability testing, on the other hand, would have focused on evaluating the application's user-friendliness and intuitive design. Lastly, performance testing would have helped us assess the application's performance under various conditions and loads.

Considering the complexity and interrelationships of the code we were testing, it was important to adopt a cautious mindset. Appreciating the intricate connections between different components allowed us to anticipate potential issues and ensure thorough testing coverage. For example, when testing the appointment service, we considered how changes in time slots could impact the retrieval and cancellation functionalities. By anticipating these interdependencies, we were able to design relevant test cases and uncover potential bugs. To limit bias in our code review, we approached it with a critical and objective mindset. We focused on the functionality and correctness of the code, rather than personal preferences or preconceived notions. We also engaged in peer code reviews, seeking input and feedback from colleagues to ensure a well-rounded evaluation. If we were responsible for testing our own code, bias could potentially arise due to familiarity and assumptions. To mitigate this, we would enlist the help of other team members or employ automated testing tools to ensure an unbiased assessment. Being disciplined in our commitment to quality as software engineering professionals is of utmost importance. Cutting corners when writing or testing code can result in subpar performance, functionality, and reliability. It can lead to the accumulation of technical debt, making future maintenance and enhancements more challenging. To avoid technical debt, we must prioritize thorough testing, code reviews, and adherence to coding standards. By investing the necessary time and effort upfront, we can minimize the likelihood of issues surfacing later in the development lifecycle.

In conclusion, the unit testing approach for the contact, task, and appointment services was aligned with the software requirements, and the JUnit tests demonstrated high quality and effectiveness. Through comprehensive coverage, careful code validation, and efficient test implementation, we ensured the robustness and reliability of the mobile application. By employing various software testing techniques, considering the complexity of the code, limiting bias, and maintaining a disciplined commitment to quality, we strived for excellence in our software engineering practices.

## APA citation

Hambling, B., Hambling, B., Morgan, P., Samaroo, A., Thompson, G., & Williams, P. (Year). Software Testing: An ISTQB-BCS Certified Tester Foundation guide (4th ed.). Publisher.

Computer Weekly. (2002). Clear and present danger: why we refused to give up. Computer Weekly, 3. Retrieved from Business Insights: Global database.

Jakubiak, N. (2022, December 6). JUnit Tutorial: Setting Up, Writing, and Running Java Unit Tests. Retrieved from https://www.parasoft.com/blog/junit-tutorial-setting-up-writing-and-running-java-unit-tests/

*Tool Support For Testing* by Peter Williams, et al. *Software Testing : An ISTQB-BCS Certified Tester Foundation guide - 4th edition*, edited by Brian Hambling, BCS Learning & Development Limited, 2019. *ProQuest Ebook Central*, https://ebookcentral-proquest-com.ezproxy.snhu.edu/lib/snhu-ebooks/detail.action?docID=5837074.