### **Victor Tran**

08/20/24

### **Summary and Reflections on Unit Testing for Project One**

In Project One, I was responsible for developing and testing a mobile application by implementing three key services: contact, task, and appointment services. This report reflects on the unit testing approach I used for each service, the specific techniques employed, and the mindset I adopted during the project.

**Summary of Testing Approach**

For each of the three services—contact, task, and appointment—I relied on unit testing as my primary method. Unit testing involves testing individual components of a software application in isolation to ensure they function as expected. This method is particularly useful for identifying and addressing bugs early in the development process, preventing more significant issues from arising later.

In the **Contact Service**, the main focus was on verifying that contacts could be added, updated, and deleted correctly. For instance, I ensured that the service would reject any attempt to add a contact with a duplicate ID, which is crucial for maintaining the integrity of the contact management system. Similar checks were implemented in the **Task Service** to ensure that tasks were managed appropriately, including adding tasks with unique IDs and updating task attributes like name and description.

The **Appointment Service** introduced additional complexity with date management. I wrote tests to verify that the service would only accept future dates and would correctly reject appointments with past dates. These tests were essential for ensuring that the service met the necessary requirements and functioned as intended.

To ensure alignment with software requirements, I used specific assertions within each test case. For example, in the appointment service, I used assertions to check that the system correctly rejected invalid IDs or past dates, thereby ensuring robust error handling.

**Defending the Quality of JUnit Tests**

The quality of my JUnit tests is supported by their comprehensive coverage of both typical and edge cases. High test coverage is critical for effective testing because it ensures that a wide range of possible inputs and scenarios are evaluated, increasing the likelihood of identifying potential issues. For example, in the contact service, tests were designed to handle various scenarios, such as invalid contact information or duplicate IDs, ensuring that the service could manage different situations effectively.

Writing these JUnit tests was an iterative process. I began by testing basic functionalities, such as adding and deleting records, and gradually added more complex test cases to cover edge cases. This structured approach allowed me to build confidence in the robustness of the services and to identify and address issues early in the development process. As highlighted in "Software Testing: An ISTQB-BCS Certified Tester Foundation Guide," a disciplined approach to testing is essential for identifying defects early and ensuring that the final product meets high-quality standards (Black et al., 2020).

**Reflection on Testing Techniques and Mindset**

The primary technique I used was **unit testing,** which focuses on testing small, isolated units of code. Unit testing is effective for verifying specific functions and handling edge cases. However, I did not employ other testing techniques, such as integration testing, which checks how different components work together, or system testing, which assesses the entire application. These techniques are important for larger projects but were not necessary for this assignment.

Throughout the project, I adopted a cautious mindset, focusing on thoroughly testing each component. Understanding the complexity and interrelationships of the code was crucial for designing effective tests. As noted in "Software Testing: An ISTQB-BCS Certified Tester Foundation Guide," testing is critical because "testing shows the presence, not absence, of defects" (Black et al., 2020, p. 28). This perspective guided my approach, ensuring I designed tests to uncover as many potential issues as possible.

To limit bias, I approached the code objectively, focusing on what the code was supposed to do rather than what I expected it to do. Bias can be a concern when developers test their own code because familiarity might lead to overlooking potential issues. By maintaining an open mind and a focus on the requirements, I ensured that the tests were thorough and objective.

Maintaining high standards of quality is crucial in software development. Cutting corners during testing can lead to technical debt, where poor practices create problems that are more difficult to resolve later. By adhering to rigorous testing practices, I aimed to prevent technical debt and ensure that the application met all requirements effectively.

**Conclusion**

My approach to unit testing in Project One helped ensure that each service met the requirements and functioned correctly. Reflecting on the techniques I used and the mindset I adopted highlights the importance of thorough testing, attention to detail, and a commitment to quality in software development. By following a structured approach and maintaining high standards, I was able to deliver a robust application that met the customer’s needs.

### **References**

Black, R., Van Veenendaal, E., & Graham, D. (2020). Software testing: An ISTQB-BCS certified tester foundation guide (4th ed.). BCS Learning & Development Ltd.