**Project Two**

Michael Kinful

Southern New Hampshire University

CS-320-11372-M01

Omar Toledo Lopez

June 21st, 2024

**Project Two**

The unit tests for the feature of task management covered the creation, update, and deletion of tasks. This ensured that tasks were correctly instantiated with valid inputs, handled updates accurately, and managed deletion without throwing any errors. That was very much in sync with the requirements, checking that tasks couldn't be created with null or empty fields in their construction and adhering to the specified constraints. High test coverage gave an expression of how thorough these tests were to include all kinds of scenarios, even edge cases.

The experience in writing such tests involved translating functional requirements into specific test cases, ensuring that each aspect of task management was validated. It was a guarantee to have a technically sound and efficient code, with all the necessary conditions validated and exceptions handled graciously.

In-detail testing of contact details management functionality ensured that creation, update, and destruction of contact information were done according to the set requirements. Tests were run to prove contacts are created with valid data, updated properly, and deleted correctly. Corresponding high test coverage percentages confirm this, stating that all possible input scenarios have been taken into consideration. These tests had to be written while making sure that the necessary conditions were validated under all scenarios and boundary conditions, with proper input validation; therefore, the technical soundness and efficiency of the code were checked.

The Appointment Management feature was tested for the correct creation, updating, and deletion of appointments. These tests checked that appointments were correctly created with valid dates and descriptions and were correctly updated or deleted. High percentages of coverage revealed that all types of scenarios and boundary conditions have been tested. When it came to writing such tests, special attention needed to be put into the process of handling and checking the dates, into checking all the necessary conditions and handling exceptions. It was meant to guarantee the technical soundness and efficiency of the code.

Basic techniques applied were unit testing and boundary testing. Unit testing ensured the validation of single elements, thus ensuring that bugs were detected and corrected at the very early stage of the process. This technique focuses on small parts of the application, which helps in easy detection and correction of issues at the very earliest period. Another type of testing done was boundary testing, a technique targeting input limits, ensuring robustness against edge cases. It is core to the validation of an application to perform well with extreme values and scenarios and ensure that it works perfectly without exception.

Other testing techniques not used in this project include integration testing and system testing. Integration testing is testing pieced-together parts of an application to make sure that they all work correctly together. It is focused on interactions amongst components; thus, it is essential for detection of problems caused by interactions between components. System testing validates that the entire placed system meets all requirements. This has to do with end-to-end scenarios to make sure the entire application works as expected.

Each of these techniques has practical applications for different software development projects and situations. Unit testing is important for very early detection of bugs, and that the individual components work correctly. Integration testing will let one know whether the components work with each other; in complex systems, especially with parts created independently, which function together. System testing ensures that the whole application complies with all its requirements, which is important in delivering a reliable and functional product to users.

The project required a wary attitude, knowing every complexity and interconnection between them in the code. One needed to understand that complexity to be able to know what might go wrong early enough and be certain of a strong application. At times, understanding how updates affected the tasks helped in evading cascading failures within the application flow.

A test, to prevent bias that may have resulted from uncontrolled assumptions, was written based on requirements rather than assumptions. Assumptions-based testing normally checks for the obvious things and may leave out some scenarios, assuming that the input will always be valid. This approach assured objectivity and was thorough since it considered all possible scenarios. As a developer, testing of one's code can result in bias resulting from familiarity of the code and the assumption that it is correct. The steps toward ensuring thoroughness and objectivity are the most important in avoiding bias when testing.

It is very important to be highly disciplined in committing quality to avoid technical debts. These shortcuts taken while writing or testing code might result in unseen problems, causative of gigantic hardships. Proper tests of corner cases and achieving high coverage percentages will forestall technical debt and point at some possible issues in advance. Code review and constant refactoring are habits that I want to inculcate in myself for writing high-quality code.

In summary, the unit testing done for the three features of Project One was closely conformance to software requirements ensured a high quality and maintainable code base. The written JUnit tests were effective; this is indicated by the high percentages of coverage. It was critical to adopt a conservative mindset and place a limit on bias in the production of reliable and robust software.

**References**

Bansal, A (2024) Best Practices for Unit Testing in Java. Baeldung. baeldung.com

<https://www.baeldung.com/java-unit-testing-best-practices>

Tasnim, R (2023) Java Unit Testing. SoftwareTestingStuff. softwaretestingstuff.com

https://www.softwaretestingstuff.com/java-unit-testing