**Summary and Reflections Report**

The goal of this follow-up summary and reflections report is to describe how I examined testing methodologies based on software requirements, share my experience developing JUnit tests and their quality, and highlight the testing strategies and the attitude I developed while working on this project. Testing is a vital stage to a successful Software Development Lifecycle (SDLC) because it is where product defects are reported, tracked, fixed and retested, until the product reaches the quality standards defined in the Software Requirement Specification.

**Summary**

For the unit testing approach for each of the three features, I focused on functional testing or specification-based testing throughout this project, concentrating on use case testing such as JUint testing. Prior to integrating an individual unit with the other units, I aimed to develop code for each component to ensure that the code written met its required specifications.

My experience writing the JUnit tests has been focused on being technical sound and efficient. The testing approach fully aligns with the software requirements. For example, the requirements for the Task class states that the task object shall have a required unique task ID String that cannot be longer than 10 characters and shall not be null. As you can see from the JUnit tests, I created to test Task class for each requirement, testTaskIdLength and testTaskIdIsNull.

@Test

    void testTaskIdLength() {

        Assertions.assertThrows(IllegalArgumentException.class,()->{

            new Task ("1234567890111","Tony","This is a description");

        });

    }

    @Test

    void testTaskIdIsNull() {

        Assertions.assertThrows(IllegalArgumentException.class,()->{

            new Task (null,"Tony","This is a description");

        });

    }

public Task(String id,String name, String description) {

        if(id==null || id.length()>10) {

            throw new IllegalArgumentException("Invalid id");

        }

...

}

**Reflection**

Non-functional testing, structural testing, change testing, and maintenance testing are other forms of software testing methodologies that I haven't employed throughout the project. Non-functional testing focuses on a system's usability and performance rather than its functionality. This testing approach mostly employs black-box testing techniques. Structural testing is used to determine how much testing has been done, whereas change testing is based on any problems discovered during testing and implementing the necessary modifications. Regression testing enables the design of a series of tests that show the system's functionality. Finally, maintenance testing occurs after the project has been launched into a live environment and any system modifications have been verified. This technique can be implemented in a variety of situations, such as when new features are needed, the system is migrated to a new operating platform, the system is retired – data may need to be migrated or archived, a planned upgrade to commercial off-the-shelf (COTS)-based systems, or new faults are discovered that need to be fixed (these can be 'hot fixes').

To assess the mindset that I adopted during the testing process, you have to look at what is the goal of testing? I would define the goal of software testing to be identifying and reporting faults. Throughout the testing process to limit bias, it is important to stay focused on the software requirements. Defects are not corrected during testing; they are sent on to the developer to fix. When changes and adjustments are assessed for their impact on other elements of the component or system during testing is another time where bias can be limited.

**Resources**

Garcia, B. (2017). *Mastering Software Testing with JUnit 5.* Packt Publishing.

Hambling, Brian Morgan, Peter Samaroo, Angelina Thompson, Geoff Williams, Peter. (2015). *Software Testing - An ISTQB-BCS Certified Tester Foundation Guide (3rd Edition).* BCS The Chartered Institute for IT. Retrieved from  
https://app.knovel.com/hotlink/toc/id:kpSTAIST01/software-testing-an-istqb/software-testing-an-istqb