**Pooja Singh**

**Dt: 12/16/23**

**Summary and Reflections Report**

**Alignment to Requirements:**

For any project, it is important to understand and meet the expectations of the client with their software. Therefore, it is important for the developers and the testers to understand and analyze the requirements in order to deliver the best solution which suffice client’s business needs. In this project, I have used happy path testing as it is a well-defined test case using known input, which executes without exception and produces an expected output. It is used to test the application through a positive flow to generate an expected output. By using happy path and error path testing I ensure we have 100% test coverage for all the classes, functions, and the all the lines of the code.

**Effective Tests:**

I have used the Junit test which covers 100% of the software. With this, I made sure that all the requirements were met. Because of the100% test coverage, the code behaves as expected. The strategy I used for the unit test was to try all the input combinations and make sure that the code didn’t break.

**Technically Sound Code:**

To make the code sound technically, I tested all the input boundary conditions for all the objects. For example under the ContactTest class (line 30 - 129), I have tested and checked the output with all the possible inputs and finally created the correct object and got correct output after testing. I didn’t write the repetitive test cases but at the same time we want to have exhaust of testing. I made sure that the name of the test cases should be self-explanatory so that the reader of the code can understand the scenario of the test case.

**Efficient Code:**

By adding multiple test cases I made sure that the code is efficient. This helped to keep the code short, neat and organized, which makes it more efficient.

**Techniques Employed:**

I have used Unit testing in which individual components of a program are tested in isolation to ensure they function correctly. This method, primarily performed by developers, focuses on the smallest units of code, like functions or methods, to identify and fix issues early, enhancing overall software quality. Unit testing helped me in refining, simplifying debugging, and providing clear documentation of the software's functionality. It's an essential practice for maintaining code integrity, facilitating efficient development, and ensuring reliable, bug-free software.

**Other Techniques / Uses and Implications:**

There are many techniques with which we can test software. The following are a few examples of them.

Black Box Testing– Black box testing which examines the functionality of an application without peering into its internal structures. In this a tester checks whether the function is working accurately in all conditions or not. However, the software’s internal design, back-end architecture, components, and business/technical requirements remain completely unknown. Black box testing is used in every level of software testing and it helps in identifying how the system responds to user actions, its performance, usability, and reliability.

White Box Testing - White box testing is also known as structural testing or code-based testing, and it is used to test the software’s internal logic, flow, and structure. It is used to analyze systems, especially when running unit, integration, and system tests. The tester creates test cases to examine the code paths and logic flows to ensure they meet the specified requirements.

Functional Testing – These are designed and run to verify all functions of a software. It checks that each function works in line with expectations. Integration Testing, System Testing, Acceptance Testing and Regression Testing are a few of its types.

**Caution:**

In my mind during this project, I was having very little caution because the simple requirements. Testing complex code can be critical which can easily lead to over testing, or a test could introduce a bug unintentionally.

**Bias:**

The simple codes would review accurate and unbiased. But complex code tends to cause bias in review. This will cover the code unintentionally and make you belief that tests cover a section of code that was overlooked.

**Discipline:**

Without discipline it is hard to maintain good quality of the code, which is required for better business reputation and user experience. Cutting corners could be dangerous as it leads to defects in the code which can cause serious threat to security or user experience. For example, in banking software cutting corners in code could lead to a security breach that could lead to severe losses in the business. Working with a good team that communicates and cooperates can greatly enhance the quality of the code in a system.

**Cited References:**

Geeksofgeeks Team, (2023). White box Testing – Software Engineering.

https://www.geeksforgeeks.org/software-engineering-white-box-testing/

Lambdatest Team, (2019). What Is Unit Testing: Detailed Guide with Best Practices.

https://www.lambdatest.com/learning-hub/unit-testing