**7-2: Project Two**

Xiao (Lindsey) Huang

Southern New Hampshire University

CS 320 Software Test Automation

Professor Mike Prasad

04/14/2023

**Summary**

In project one, I was tasked to develop and test mobile applications that provide clients with Contact, Task, and Appointment services. It allows the client to manage their customers' database and manage their appointments. The approaches I used to ensure I aligned with the software requirements were to follow the requirement stages and specify the requirement elicitation and requirement analysis. It was essential to understand the requirement because it defines the software's test criteria to verify and validate the functionality of the software. It ensured I had delivered a working product to the clients. Requirement elicitation was the first stage in that I reviewed the software's requirements for desired features and test criteria by carefully reviewing the scenario and the goal of the project. Then I generated a comprehensive list of the software to develop and tested it to ensure it meets the requirements and the desired purpose of the software as a checklist. Next, in the requirement analysis, I created unit testing cases for the application to validate its functionalities and ensure it works properly. For example, I created a testing unit to ensure no duplication id can be stored in the memory data structure.

Text

Description automatically generated

Overall, my Junit tests were effective because the coverage percentage for my application was greater than 80%. The Junit testing coverage percentage indicated the efficiency of the application. The recommended coverage percentage is equal to or greater than 80% and is the optimal test coverage rate. Another consideration I had to ensure my code was effective was ensuring the Junit testing cases’ quality met the coding standard, like having a clear and easily understood testing name. also, the testing result was accurate to the test scenario.

To ensure my application was technically sound, I learned to extensively test every method, constructor, and invalid input data. I also ensured that my coding met the specific conditions per the project requirements. I kept my coding simple and documented throughout the coding to ensure the coding had a clear definition for easy maintenance and reusable. For example, the contact’s name requirement can't be null or exceed the length of 10 characters.

Text

Description automatically generated

In the Junit testing, each testing method's name is clearly defined for the reader to understand its purpose. I tested the valid and invalid data to ensure it worked properly and was able to throw errors as planned.

To ensure my application was efficient, I ran the Junit testing I developed to identify potential errors and check that the application's functionalities are performing correctly. The JUnit testing will validate the inputs and the functionalities. Then I can verify it by checking the pass/fail status and the coverage percentage report.

**Reflection**

The software testing techniques I employed are mainly dynamic testing, including white box and unit testing. Dynamic testing is used to test the code when executing in real life environment. It checks the functional behavior of the software. The goal of dynamic testing is to verify that the software meets the functional requirement of the project. The white box testing tests the code when the tester understands the internal structure and the software design. I used unit testing to test each method as a small unit for errors and functionality that meets the requirement.

The software testing technique that I didn't use for the project is black box testing. The black box is used to test the software without the knowledge of the software's internal structure and design. The tester is testing it from the end user's perspective point of view, and the project didn't require the end user interface. Another software testing technique I didn't fully use is the static testing technique. Static testing techniques are techniques that don't need the execution of the code. It's done through manual testing by the tester. When I developed the codes for the application, I tried to visualize the coding process in my mind, but I didn't have the whole mindset to do it. It will likely result in software defects and errors.

Each type of software testing technique has its advantages and disadvantages, and it will be based on the tester and the need of the team and clients to determine what to use. Ideally, it will be best to collaborate on both techniques at different phases of the Software Development Life Cycle. For example, static testing is best used in the early phase of the development process. It helps the team look for errors and defects early and helps determine a cost-effective solution to fix the issues. Also, it allows the tester to analyze it before the codes are completed. The project doesn't require user interaction, so dynamic testing is the best choice, and running the Junit5 testing in the IDE helps detect and ensure the codes have met the requirement without errors.

While developing the application in Project One, I focused on meeting the requirements and following the standard coding practice. Following the standard coding practice ensures that the coding is reusable, scalable, portable, and transparent with class and function names and purposes. Programing cautiously is essential to eliminate and prevent bugs and security vulnerabilities for a secure application. I carefully reviewed my coding throughout the development to ensure I had tested the code’s functionality and the error handling testing. It’s important to ensure the complexity and interrelations of the code are considered to ensure the classes work appropriately independently and can corroborate together to work as a complete system.

It’s easy for the developer to be biased when reviewing their product, as they are confident in their skills. To help limit bias when reviewing, I relied on the coverage percentage report to ensure I had met the optimal coverage percentage recommendation. Also, I maintained a checklist of the requirements to ensure I had developed extensive unit tests to cover all the data.

Learning to be a disciplined software engineer will help me improve my coding skills as I continue developing clean and well-documented code and learn to debug and test code to increase the quality of the product. Cutting corners is never a practice I will participate in because I commit to developing a high-quality and working product for my clients. Cutting corners will cause several severe issues, like introducing security vulnerabilities, causing the team and the clients to lose time, money, and reputation. My plan to avoid technical debt as a practitioner is to follow the software development: requirement gathering, designing, coding, testing, and maintenance. To prevent the application's default, first, I will need to understand the client's requirements through the requirement gathering stage, and then I will use the checklist to design and code the application. Next, I will develop the testing to validate the functions and error handling. Last, I will ensure that my codes are clear and well-documented for others to maintain.

**References**

Boni Garcia. (2017). *Mastering Software Testing with JUnit 5: A Comprehensive, Hands-on Guide on Unit Testing Framework for Java Programming Language*. Packt Publishing.

Hambling, B., Morgan, P., Samaroo, A., Thompson, G., & Williams, P. (2019). Software testing: An istqb-bcs certified tester foundation guide – 4th edition. BCS Learning & Development Limited. Created from snhu-ebooks on 2023-03-06.