Based on the project's scenario, I was able to analyze various approaches to software testing based on requirements and apply appropriate testing strategies to meet requirements. Through demonstration of these competencies, I was able to improve my masteries of these skills. Now I will construct a summary and reflections report to be submitted that summarizes my unit testing approach. Displaying my experience writing the JUnit tests and delivering the overall quality of my JUnit tests. This report will also highlight testing techniques and the mindset I adopted while working on this project. While analyzing various approaches to software testing based on requirements and applied appropriate testing strategies to meet requirements, I was able to develop a mobile application for the customer. This report is based on my experience completing Project One.

My unit testing approach for each of three features was able to align with the assignment's software requirements, displaying the tasks functionality through source code built with the java programming language. Each class assignment that I had to comprehend and apply to my program had strict requirements. For example, among the 6 class files each file had requirements that were unique to the classes' functions, and I feel that I was able to reach a consensus that related to the topic's assignment. Tasks such as applying if statement conditionals and writing methods gave me the ability to write these programs with efficiency and execute what was needed. Through these actions and the JUnit testing framework I developed and ran automated tests ensuring that the code worked as intended. The overall quality of these tests displayed 85% plus coverage rate among all tests. In other words, the effectiveness based on the coverage percentage verifies the strict precautions that I followed.

2.

Software testing techniques were apparent in this project, and I was able to identify some that applied to this specific scenario. The first technique that I would like to mention is functional testing. Functional testing tests the basic functionalities of an application against a set of specifications. While keeping this thought in mind, it was simple to verify how the project would function and what the overall dynamic of the application would appear as. This technique and alaysis paired with unit testing gave my projects its unique appearance and integrity. Unit testing tested individual components or functions of the code in isolation ensuring that each part functions correctly. Further context of this concept can be identified through all my class files because of the way the assignments that I wrote were coded. The last technique that I would like to mention that helped with my project would be usability testing. Usability testing assesses how user-friendly and intuitive the software interface is for end-users. I would like to donate this idea to my individual classes that I have created due to their functionality's adaptiveness and modifiable structures.

Other software testing techniques that I did not relate to this specific project would first be regression testing. Since I verified the assignments only briefly, I was not able to determine the software development life cycle of the classes. Regression testing ensures new changes do not negatively impact the existing functionalities of the software. Another software testing technique that I would like to talk about that is not commonly used, especially in this specific scenario for my project, would be chaos testing. The purpose is testing software resilience by intentionally introducing faults or "chaos" into the system to observe how it behaves and recovers from failures. In this specific case I did not find the need to introduce many faults to analyze the project and how the recovery would display to the user.

For each of the techniques that I have discussed the practical uses and implications for different software development projects and situations apply several distinct coding project scenarios. The reason for their adaptiveness stems from the concept styles and structure of certain scenarios. Displaying similar method uses and qualifications. When using similar methods across different projects the relatability between projects displays similar nature in style and framework.

b.

Assessing the mindset that I adopted while working on this project. In acting as a software tester my extent of employing caution showed its apparency with the organization of my classes’ structures and similarities. It is important to appreciate the complexity and interrelationships of the code I was testing because each specification required a slightly different execution. For the execution to be proper the following of the guidelines of each assignment had real regulation. You would not be able to exhibit the program’s complexity without following the proper rules with caution.

Ways to limit bias in the review of my code starts with viewing the assignment and critically thinking about each way you can display the program to the user while implementing its functionality. On the software developer side, I can imagine that bias would be a concern if I were responsible for testing my own code. I would say this because as the solemn user you are subject to a fixed number of ideas in this process. To explain further on my thought process, I would like to establish that bias can emerge in software development when program requirements reflect assumptions, preferences, and or perspective of those creating or specifying the software. You receive puzzle-like ideas such as incomplete requirements, cultural biases, and algorithmic bias. The biases promote complications among specific users and can lead to project failure.

While evaluating the importance of being disciplined in my commitment to qualify as software engineering professional. It is important to hold and critically think about high program integrity and intellectual reasoning. It is also important to not cut corners when writing and testing code because minor errors in your product can always degrade the product's quality. To avoid technical debt as a practitioner in this field you must remain careful and verify all of your options for each project. For example, while working on this project I made sure to verify all the requirements and ensured that the project was working even while changing factors such as object fields, that way I can see that the project is always working and does not regress. Another way that I was able to avoid technical debt is by keeping the project simple and only doing what was asked of me and not overstepping bounds. Overstepping bounds can at times beneficial because it can display uniqueness but can overcomplicate the product. So, it is good to recognize what is appropriate for your project's lifecycle and what the user is asking of you.

Iroegbu, S. (2022, September 21). *What is software testing? A beginner's guide*. freeCodeCamp. <https://www.freecodecamp.org/news/software-testing-beginners-guide/>