My approach was closely aligned with the software requirements because I knew that if I strayed away from it, I might mess up or make an error in my projects. By sticking to the script, I was able to focus on the problems I need to fix with the requirements lined up for me. I was able to zone in on the specific things needed to accomplish those feats. The quality of my Junit tests were serviceable. The coverage percentage was probably around 90% based off how much I was following the system requirements. All in all, that’s not too bad of figure to have in that situation.

I ensured my code was technically sound by incremental testing and careful following of the software requirements once again. I ensured it was efficient by this testing method and by seeing how it lined up with what I had to accomplish with the prompt.

The software testing techniques that I used for this project include black box testing and white box testing, while acceptance testing, exploratory testing, regression testing, performance testing, and security testing are all different types of testing I could have employed for my project. Each of these techniques has unique characteristics and is suitable for testing different aspects of software applications. For example, black box testing is a type of functional testing that focuses on the behavior of the application and does not require knowledge of the internal code. White box testing, on the other hand, is a type of structural testing that involves examining the internal code and ensuring that it is functioning as expected these are the two that I used for my project.

Other software testing techniques that were not used in this project may include usability testing, compatibility testing, and accessibility testing. Usability testing involves testing how easy and intuitive the software is to use. Compatibility testing ensures that the software works as intended on different devices, operating systems, and browsers. Accessibility testing checks whether the software can be used by people with disabilities.

Practical uses and implications of each testing technique depend on the specific requirements and characteristics of the software development project. For example, acceptance testing is often used to ensure that the software meets the client's requirements, while performance testing is critical for software applications that must handle high volumes of traffic.

Mindset: As a software tester, it's important to adopt a cautious and thorough mindset when reviewing code. In this role, the tester must identify any issues or defects in the software, so they must approach testing with a critical eye and pay close attention to details. It's also important to appreciate the complexity and interrelationships of the code being tested to understand how changes in one area could affect other parts of the application.

To limit bias in reviewing code, it's essential to remain objective and unbiased in the testing process. It's crucial to avoid making assumptions or jumping to conclusions based on personal beliefs or preferences. In some cases, it may be helpful to use automated testing tools to reduce the impact of human error and bias.

As a software developer, it's possible to be biased when testing your own code. For example, you may overlook errors or defects because you're too familiar with the code and assume it's working correctly. To avoid this bias, it's important to adopt an objective mindset and try to approach testing as if you were a third-party reviewer.

Discipline in commitment to quality is essential for software engineering professionals. Cutting corners when it comes to writing or testing code can lead to technical debt, which can be costly and time-consuming to fix later on. To avoid technical debt, it's important to prioritize quality at every stage of the development process, from planning and design to coding and testing. This means taking the time to write clean, efficient code and thoroughly testing it to ensure that it meets the requirements and works as expected. By doing so, you can avoid costly rework and ensure that the software is reliable and performs well.