In my recent project for Grand Strand Systems, I undertook the development of a mobile application that included contact, task, and appointment services. To ensure that each feature met the specified software requirements, I implemented a rigorous unit testing approach throughout the development process. I created JUnit tests for all three services, focusing on validating key functionalities such as adding, updating, and deleting information.

For the \*\*Contact Service\*\*, I designed tests that rigorously checked specific constraints, including verifying that each contact had a unique ID not exceeding ten characters and that phone numbers were formatted correctly with exactly ten digits. Unfortunately, despite my extensive efforts, I was unable to implement effective JUnit tests for the Contact Service, ultimately resulting in a failure to generate a coverage report for this component of my application.

After completing the Contact Service, I applied a similar rigorous testing approach to the \*\*Task Service\*\*. I developed JUnit tests that validated essential functionalities like task creation and ensured the uniqueness of task IDs. Each test was meticulously mapped to the project requirements, thereby confirming that all necessary validation checks were in place. Likewise, for the \*\*Appointment Service\*\*, I focused on ensuring comprehensive coverage of functionalities related to appointment creation and updates. My tests validated that the appointment date could not be in the past, fully adhering to the specified requirements. While I achieved a coverage percentage exceeding 80% for the Task and Appointment Services, the issues I encountered with the Contact Service prevented a similar assessment of its coverage.

Writing JUnit tests was often challenging yet rewarding. During this process, I utilized assertions such as `assertNotNull(contactService.addContact(validContact));` to systematically validate correct behaviors. Additionally, I organized my test methods using setup and teardown functions, which minimized redundancy and fostered efficiency and clarity within the test suite. However, I faced significant hurdles while transitioning all my projects to use Maven, a necessary step for effectively conducting coverage tests. During the build process, I encountered a compilation failure where my `ContactService.java` file could not find the `Optional` class, leading to persistent errors. Despite my efforts to resolve these issues, I struggled to identify the root cause. The error messages indicated that the Maven Compiler Plugin was experiencing difficulties that ultimately caused the project build to fail. These technical setbacks were frustrating because they impeded my ability to generate expected coverage reports and thoroughly test the Contact Service.

Throughout the project, I focused primarily on unit testing, which allowed me to isolate and validate functionality at the method level. However, I also recognized the importance of integrating additional techniques such as integration testing and end-to-end testing, which I did not implement due to time constraints. Understanding how these techniques complement one another highlighted the necessity of a comprehensive testing strategy across different stages of software development.

A cautious mindset was imperative as I navigated the complexities of the code. Acknowledging the interrelationships between different services within my application was crucial for identifying potential pitfalls and ensuring consistent functionality across services. To minimize bias in my testing efforts, I actively sought peer reviews of my code and test cases, recognizing that, as the primary developer, I could inadvertently overlook certain flaws.

Finally, maintaining a disciplined commitment to quality is essential as a software engineer. Cutting corners in writing or testing code can lead to significant technical debt and complications in the future. Moving forward, my strategy will prioritize thorough testing practices and regular refactoring of my code to avoid such pitfalls. By adopting this disciplined approach, I aim to safeguard the integrity of my current projects while continually enhancing my skills for future endeavors.

Overall, this project has been a valuable learning experience, emphasizing the importance of a structured testing approach and the necessity for continuous learning in software development. Although I faced challenges with the Contact Service's JUnit tests and struggled to generate a coverage report, I am committed to leveraging the lessons learned to improve my capabilities in future projects.