Grand Strand Systems, a software engineering company that focuses primarily on developing and testing back-end services, recently developed a mobile application for a customer and delivered contact, task, and appointment services modules. These modules had varying technical requirements that had to be validated and tested to ensure compliance with the company's technical specifications. Thorough and rigorous testing was accomplished to ensure that the product was technically sound and met stringent technical requirements.

To align the modules with the software requirements, I began by ensuring that I captured all the requirements for each module. Careful analysis of the technical specifications ensured that I was able to capture character requirements, interactions with functions within the modules, and any other specifications that were noted in the technical requirements. Character limitations, no null or blank entry fields, and the ability to add, remove, or modify functions within the modules were outlined in the technical requirements. Adhering to and testing these functions helped to ensure compliance with the outlined requirements.

A close-up of a white background

Description automatically generated

The depiction above demonstrates how the code was technically sound. The example of the Task class shows how the technical requirements are met in the Test class. The requirements could have been put in other places in the code but keeping them in a concise location helps with the readability and maintainability of the code. This also makes the code more efficient. This also allows for further development if changes need to be made. The screenshot also captures the proper implementation of the getter and setter technique, commonly used in Java. This also makes the code more efficient. Lastly, my usage of in-line comments helps with the readability and maintainability of the code. The in-line comments help capture what portions of the code perform which functions, allowing for easy interpretation.

There were many software testing techniques that I used when developing these modules. Critical analysis of the technical specifications was required to not only ensure proper implementation but also to ensure proper validation. Running tests to try to violate and stress the imposed requirements were required to ensure software functionality and requirements conformity. Some techniques that I did not use during this project included testing specific characters or strings. From a software security perspective, the omission of certain characters or strings would be required to help protect against injection-style attacks. Another testing technique that I did not incorporate was peer-review testing. Having another individual analyze and test my code could help identify and eliminate aspects that I may not have seen. Whether it is the development process or the testing phase, working together with others is a great way to incorporate different perspectives.

Analyzing requirements and designing tests to ensure that those requirements are met is practical and necessary. There are several implications for this style of testing. If the developer who designed the code is also the developer responsible for testing the code, bias can become present and can be detrimental to the testing process. To help mitigate this bias, having developers specifically for application development and having different developers for testing is necessary. Bias is hard to eliminate or mitigate if the same developer is creating and testing the same code.

Caution is required when developing and applying tests to existing software. Testing is designed to stress the code and ensure compliance with requirements. If testing isn’t thoroughly thought out and vetted, then it can result in invalid, undesirable, or destructive results. Testing also must be thoroughly analyzed before and after commencement. Do the tests make sense? Are the tests checking to see if the code executes as designed? Even though the tests may pass, maybe the tests are faulty. A test may yield passing results, yet that may be because the code does what it is supposed to, although not entirely what it was intended to do. This could be the result of bias in the testing process, a rushed testing process, or the use of incomplete or inaccurate tests.

Bias is one of the most adverse ideologies that can affect an accurate and complete testing process. Eliminating bias should be a goal in all software development teams and should be considered throughout the entire software development lifecycle. The software development team should have individuals assigned for module development and have different individuals assigned for testing. This can help eliminate bias from the module developers affecting test results. Having as many independent minds looking at the software and tests as possible helps create a diverse environment that can shed light on aspects of the code that may not have been readily apparent.

Discipline is important to software development because it ensures that a high-quality product is distributed to users and that data integrity is intact. Deadlines, profit margins, and management oversight can lead to corners getting cut. Inadequate testing can lead to data insecurity and harm to users, the company, and the public. End-users trust that our application will protect their data from malicious attackers, so we must ensure that our software is rigorously and thoroughly tested to do so. If we cut corners and develop insecure code, then what trust do end-users have in our product? Why would they trust us to protect their private data? It is our duty and responsibility as software engineers and developers to ensure that all customer and company data is protected from malicious intent. This can only be accomplished with a thorough, adequate, and complete testing program.