Module Seven Project Two

My approach was aligned to the software requirements as I had taken the list of what needed to be included in the application and worked through each file to ensure all requirements were met. An example was the task application where the task object had multiple attributes, and each attribute had a specific constraint. One of the attributes was task ID this attribute could not be null or longer than ten characters. To ensure that this requirement was met, I had implemented an if statement that would stop the input from passing through if it was an invalid input. The attribute was then tested in the task service file where I had a valid input and invalid inputs entered to ensure that the if statement worked correctly. One example of the test was inputting null and in return an error message would appear, and the input would be blocked from passing through the system.

I know my JUnit tests were effective based on the coverage percentage as I was just over 80% for coverage. This meant that the tests tested about 80% of my code and what was tested was functional. I felt it was effective as I could see a breakdown in each file of what was tested and what was not, and it looked like almost all my attributes and functions were tested.

I ensured that my code was technically sound as I stayed consistent with coding style across all three modules. I made sure each application was simple and clear by making sure I had meaningful names for attributes and functions that made sense to the application and the requirements given. For example, in the appointment application I had functions named “addAppointment” and “deleteAppointment” that clearly explains that one function adds appointments and another deletes appointments. I implemented a README file for each application that quickly summarized what the purpose of the application was as well.

I ensured that my code was efficient by keeping the code as simple as possible. Minimizing the number of operations used helped save unnecessary computations from happening each time. Another way I stayed efficient was using a hash map for all three modules instead of an array or a list. A hash map helps save time with memory allocation and deallocation as the hash map retrieves values associated to a specific key in near-constant time, so a value could be created, found, updated, and deleted much quicker. Whereas, using a list it would take a significant amount of time for the system to iterate through each value in a list to find the value that is being called for.

The software testing technique that was used in this project was modular testing. Throughout three weeks, I had slowly worked on and tested each component to build the final project that could manage appointments, tasks, and contacts for a company. Within each week, I implemented boundary testing that tested the system’s behavior when valid and invalid inputs were entered to ensure that the constraint worked properly. Another testing technique used was functional testing to ensure each function operated as expected and met the requirements. An example was during the contact application where one function involved updating the contact object. The function was expected to update if needed the contact ID, first name, last name, phone number, and address. To ensure that this function worked I implemented tests that called the update contact function with new input.

Other software testing techniques that I did not use in the milestones were integration testing and performance testing. Integration testing is used to ensure that different components function properly together. For project one, all three modules are integrated together to create one final application. Integration testing would be the next test I would implement to ensure each object and service file interact with each other as expected. Performance testing helps evaluate an application’s scalability, responsiveness, and stability. This is done by using extreme conditions to test the behavior of the application and how it handles overloads whether it is in surges or for extended periods of time. This test would help see if the final application with all three modules integrated together can handle any amount of traffic and maintain stability throughout.

The practical uses and implications of these techniques depend on what the purpose of an application is. Unit testing is most practical for situations where it is early in the development stage or for smaller applications. Boundary testing is practical when there are specific constraints for attributes, and they need to be tested to ensure input validation is properly working. Integration testing is practical for larger applications that have multiple components interacting with each other and these interactions need to be tested. Performance testing is practical for applications that are expected to receive large amounts of traffic to test whether or not the application can scale to meet the needs of traffic while maintaining speed and quality.

The mindset I had during this project was how I can keep the code as simple and efficient as possible while meeting all requirements. When creating each object in each module I took time to research what data structure made the most sense to use to store each object and have the ability able to easily access and manage any object. Going with the hash map I was unsure if I was making the right choice as I am not very experienced in coding and did not know if there are better options. After coding the first application I felt confident in the hash map as it had worked how I wanted it to and it worked well.

I tried to limit bias in my review of code by using automated tools to help me truly analyze my code and give me another perspective. The JUnit tests originally told me that my coverage on one of the applications was well below 80% and I would most likely would have not caught that I was not testing all of my code. Using the coverage details to find what tests I was missing helped me significantly as I ended up adding another test and made some other tweaks in the service file.

It is important to not cut corners when it comes to programming and testing as it could lead to an application that does not function properly or as expected. More time will be taken to review the code and then resolve what could have potentially been prevented in the first place. I plan to avoid technical debt and ensure that quality stays important to me by trying to prioritize and plan out my work.