Anthony W Bradley

CS 320 Software Test, Automation QA

Professor Kalysa Wilson

06/23/2024

Summary and Reflections Report

In this project, I developed three services: Contact, Task, and Appointment. Each service was designed with specific requirements and verified through JUnit tests. My unit testing approach for each of the three features was systematic and thorough. I created tests for each requirement, ensuring that all fields met the specified conditions, such as length and nullability. I also tested the functionality of each service, including adding, deleting, and updating records.

My approach was closely aligned with the software requirements. In the Contact Service, I ensured that the contact ID was unique, not null, and not updatable. I also verified that the firstName, lastName, phone, and address fields met their respective conditions. Similar tests were conducted for the Task and Appointment Services.

The overall quality of my JUnit tests was high, as evidenced by the 80% coverage. This means that much of my code was tested, increasing the likelihood of catching and fixing bugs. Writing the JUnit tests was a meticulous process. It required a deep understanding of the code and the requirements. It was a challenging yet rewarding experience, as it helped ensure the reliability of my code. To ensure that my code was technically sound, I adhered to best practices for coding and testing. I wrote clean, readable code and organized my tests in a logical manner. I also made sure to test edge cases to ensure my code could handle unexpected inputs or scenarios. To ensure efficiency, I avoided redundancy in my tests and code. I also used appropriate data structures which helped optimize the performance of my services.

The software testing techniques employed in this project were primarily white box testing and unit testing. White box testing allowed me to understand the internal workings of my code and create tests based on that knowledge. Unit testing enabled me to test individual components of my code in isolation, making it easier to identify and fix issues. There are other software testing techniques that were not used in this project, such as integration testing and system testing. These techniques involve testing the interaction between different components and the system, respectively.

Each testing technique has its practical uses and implications. unit testing is great for catching issues early in the development process, while integration testing is crucial for ensuring different components work well together. Throughout this project, I adopted a cautious mindset. I understood the complexity and interrelationships of the code I was testing. I knew that changing one part of the code could potentially affect other parts. This understanding guided my testing strategy and helped me create comprehensive tests. I also made efforts to limit bias in my review of the code. As a software developer, it’s easy to assume that your code is free of errors. However, this project reinforced the importance of thorough testing, regardless of who wrote the code.

This project highlighted the importance of discipline in my commitment to quality. Cutting corners when writing or testing code can lead to technical debt and potential issues down the line. I plan to maintain high standards for my work and continually learn and adapt to best practices. This project served as a valuable learning experience in my journey as a software developer.