Keith Spraglin

CS320 Project Two

October 17, 2024

Summary

Unit testing is extremely important when creating software solutions, as it helps reduce the risk of errors in your programs and improves code quality by quickly detecting and fixing defects. The unit testing approached I applied for the three features were similar as I made it to align with the software requirements. A prime example is a Contact service that has certain requirements such as Contact objects must have a unique Contact ID string, must not exceed 10 characters, and cannot be null or updatable. All they wanted was to require Contact objects to have a first and last name that must not exceed 10 characters and cannot be null. They also required a phone field that was a string, must contain exactly 10 digits, and could not be null. The Contact class also required an address field that could not exceed 30 characters and could not be null.

The Contact Objective has a service of its own, the ability to use crud operations with the contact object. These were required for one of the three features, along with the rest of the features with similar requirements. To pass the unit test, the software must implement the requirements. While the overall quality of the test was good, I tested each operation for validity of the test cases.

I had a hard experience with the project, but better one gathering the skills, code of each individual element for the project, there is still a lot to learn.

A computer code with text

Description automatically generated

The idea of the above image is to check the validity of the parameters from the contact class. I created a test case to Verify TestID was long enough. Ideally this should be effective but as you see I was getting errors in the code.

A screenshot of a computer program

Description automatically generated

The code above is testing operation on the contact service. This way is good to know if your code is effective.

Reflection

In this project, I used two main software testing methods: black box testing and white box testing. With black box testing, I input data and checked if the outputs were correct. White box testing involved looking at the code to make sure the logic was right. I also created JUnit test cases to find edge cases and catch any possible errors. I didn’t use performance testing or security testing for this project. Performance testing checks how fast a system works and how well it handles changes, while security testing looks for weaknesses that could be exploited.

The testing methods I did use helped confirm that the code was correct and worked well. They helped reveal mistakes and identify edge cases. However, not including performance and security testing means some problems might have been missed. Throughout this project, I focused on being careful and thorough. I tried to minimize bias in my code reviews and unit tests to fully evaluate how everything worked. As a developer, it’s important to recognize and manage any biases while testing your own work. In the end, staying committed to quality is vital when coding and testing. Skipping steps can lead to issues down the line. To maintain high standards, I depended on code reviews and unit tests to ensure the code was thoroughly checked and free of errors.