Sean Glass

CS-320: Software Test Automation & QA

Southern New Hampshire University

October 15, 2023

**Project Two: Summary and Reflections Report**

**Summary**

When creating software applications, writing efficient code that aligns with a client’s requirements is essential. Each service in project one had its own requirements, although some of them were quite similar, which made writing the code a little easier. For each individual requirement, I would go back and verify that the requirement was being fulfilled before moving on to another task. For example, in the *Contact Service* part of the application, there were a total of eight requirements between the *Contact* class Java file and the *Contact Service* Java file. For each individual requirement, I had to write the code for the requirement and then verify that the code actually meets the respective requirements.

For each requirement, I was able to create high quality tests to verify that everything worked as expected. Running the code to determine coverage percentage is important, so I made sure to do that as well. A higher coverage percentage is better, as that means that more of the code is covered by tests to ensure quality.

To ensure that my code was technically sound, there were several things that I made sure to do. First, I made sure to test one thing at a time. This is often called the *single responsibility principle*. I did this to make the tests easier to understand and to help make it easy to find parts of the code that are not working correctly. Next, I kept the tests independent of each other. In other words, I did not write any tests that rely on another test to pass. Finally, I made sure to have a reasonable level of test coverage to ensure that most of the code is covered by tests. You can see examples of each of these throughout the project in each “test” Java file.

To ensure that my code was efficient, I made sure to do a few things including reducing redundancy and avoiding nested loops. It is important to reduce redundancy for an efficient program because redundant code makes maintenance more difficult because any changes or bug fixes may need to be applied in multiple places. Avoiding nested loops is important because deeply nested loops can make your code less readable and harder to understand. Nested loops also increase the time complexity of your code. (See *Task.java, line 17-23*. Notice the use of the **OR** operator **||** instead of using nested *if* statements.)

**Reflection**

The software testing techniques that I used in project one included black-box and white-box testing. Black-box testing involves entering data and checking if the outputs are what was expected. White-box testing involves reviewing code and making sure the logic is correct. Another technique I used was using edge cases to test the code for issues.

The software testing techniques that I did not use for project one included performance testing and security testing. Performance testing is used to test the speed, scalability, and responsiveness of a system. Security testing is used to identify and mitigate potential security risks.

The practical use of the software testing techniques I used is that they can be used to verify the correctness and efficiency of the code. Not utilizing specific software testing techniques can have implications such as the potential for errors and overlooked security vulnerabilities.

In terms of my approach during this project, I wanted to remain cautious, acknowledging the complexities of the code. I mitigated bias in my code review process by actively deploying unit tests to be diligent in testing my code. It's imperative for software developers to remain conscious of the potential for bias and to exhibit diligence in testing their own code.

A firm commitment to code quality is essential in the software development process. It is crucial to avoid shortcuts in code development and testing to deter accumulation of technical debt. To accomplish this, I relied on unit tests to assess the code’s integrity and ensure that it is free of errors.