Tim Vue

# Project Two

1. Summary
    1. Describe your unit testing approach for each of the three features.
        1. To what extent was your approach aligned to the software requirements? Support your claims with specific evidence.

For each of the tests that I wrote up, I focused primarily on the specific requirements stated, those being the length of the ID strings, and all of the contact info of the contacts created. I made sure that none of these fields may be left empty or null using an if statement like this
if (name != null && name.length() <= 20) { this.name = name).

2. Defend the overall quality of your JUnit tests. In other words, how do you know your JUnit tests were effective based on the coverage percentage?

The JUnit tests that I wrote up had 100% coverage and met the expectations of what was required of me for this task. I was able to test different lengths of IDs or other contact information for known exceptions.

2. Describe your experience writing the JUnit tests.
    1. How did you ensure that your code was technically sound? Cite specific lines of code from your tests to illustrate.

I made sure to keep my code as simple and clean as possible asserting the expected throws and creating the right try/catch statement such as in the ContactServiceTest class on line 27.

        2. How did you ensure that your code was efficient? Cite specific lines of code from your tests to illustrate.

While writing my test cases I made sure to only test exactly what I wanted to test like in line 33 in the TaskTest.java class where I test only if the task ID is too long.

2. Reflection
    1. Testing Techniques
        1. What were the software testing techniques that you employed in this project? Describe their characteristics using specific details.

In this project, the software testing technique that I implemented was Unit testing to ensure that each class was working as intended.

        2. What are the other software testing techniques that you did not use for this project? Describe their characteristics using specific details.

I did not use Integration Testing which verifies the interaction between different components or Load testing which tests how a program performs under specific loads as they were unnecessary for this program.

        3. For each of the techniques you discussed, explain the practical uses and implications for different software development projects and situations.

Integration Testing would be better utilized in a program with many different components that need to communicate with one another such as a video game where a signal might need to be received by many other nodes. Load testing might be used if we were working on a cloud service and needed to test how the program would function with 10,000 - 20,000 users on at once.

    2. Mindset

1. Assess the mindset that you adopted working on this project. In acting as a software tester, to what extent did you employ caution? Why was it important to appreciate the complexity and interrelationships of the code you were testing? Provide specific examples to illustrate your claims.

I had to ensure I was thorough with my tests to cover all the exceptions and other requirements specified. While creating the initial class and the service classes I had to also ensure that these two were correctly inheriting variables and exceptions.

2. Assess the ways you tried to limit bias in your review of the code. On the software developer side, can you imagine that bias would be a concern if you were responsible for testing your own code? Provide specific examples to illustrate your claims.

As I am the one assessing my own code I tried not to overlook possible issues or favor my own solutions to problems. The tests I wrote were extremely simple and I tried making them as clean as possible but I'm sure they would not suffice under professional scrutiny.

3. Finally, evaluate the importance of being disciplined in your commitment to quality as a software engineering professional. Why is it important not to cut corners when it comes to writing or testing code? How do you plan to avoid technical debt as a practitioner in the field? Provide specific examples to illustrate your claims.

I believe cutting corners is just cheating yourself, though doing things the easy way may work our initially, in the long run will only come back to bite you. Doing something properly will take time and effort, but will also reward you in the future for the effort you've put in. I know I won't be able to create a JUnit test blindly still and will need to research and take time to create a proper test, but in the future, it will be much easier to understand the concepts and create more elaborate tests.