Danielle Sousa

CS 320 Software Test and Automation

4/11/23

**Project Two**

**Summary:**

**a. Describe your unit testing approach for each of the three features.**

**i. To what extent was your approach aligned with the software requirements? Support your claims with specific evidence.**

My unit testing approach aligned with the software requirements. For example, In Contact Service for ID length, if the ID is over 10 characters or is null then an exception is thrown. The same logic was applied to the rest of the requirements in Contact Service.

Text

Description automatically generated

The rest of the requirements for Contact service are for throw exceptions.

Graphical user interface, text, application, email

Description automatically generated

**ii. 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?**

According to coverage testing. For my contact class, there was an overall of 79.7 percent lines tested and passed. Which is just shy of the required 80 percent.

Graphical user interface

Description automatically generated with low confidence

When digging in further as to why it was only 79.7 percent. For example, get ID test. You can see the line in green tested pass instructions. However, the succeeding line are in yellow which means that the test has run through these instructions however not all possible cases have been covered. Which helped to bring my test percentage to 79.7 percent. My J-unit test was effective but needed improvement.

Graphical user interface, text, application, chat or text message

Description automatically generated

1. **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.**

In the Appointment test, you can see that when setting an appointment date if the date is null or before the current date an exception is thrown for the Invalid appointment date.

Graphical user interface, text, application, Word, chat or text message

Description automatically generated

By looking at the coverage for the Appointment test you can see that 100 percent of the lines passed the test which by these testing standards suggests this class is technically sound code and adhered to the requirements for this class.

* + 1. **How did you ensure that your code was efficient? Cite specific lines of code from your tests to illustrate.**

The way I ensured the code was efficient was to test the code. The main goal was to ensure that all requirements were met and that inputs validated the unit tests.

Graphical user interface, text, application

Description automatically generated

**2. Reflection:**

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

The software testing techniques that I used in this project were unit testing, which used J-unit testing is an example of such test. It tests individual units or components of the software to ensure they are functioning.

Same for coverage testing, which is a method of testing which lines/methods. Classes are executed during a particular session.

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

Integration Testing was a technique not used in this project. This test looks for defects and failures. Basically, tests the behavior of the components of the system together. For example, the appointment, Task, and contact classes all work together to take the input and put it into output, however, there is no interface to make this happen.

User Acceptance helps to decrease the chance and severity of new and regression defects. Encourages closer collab between developers on the one hand and customers on the other which ensure business requirements are met.

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

For integration testing its practical use would be used on any software project that is developed to make sure all code works together to adhere to requirements and works as intended.

For acceptance testing, since there is no interface and essentially no customer there was no need to do acceptance testing since this course is only meant to understand the testing process at its basics. However, in a real-case scenario, this test has a binary result pass or fail. A failure suggests a defect in the product. Acceptance tests are used as a main form of a functional statement and formal articulation of the business requirements.

1. **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.**

The mindset I took going into this project was one of openness to making mistakes, which since this is all new to me I knew I would be making these. Making mistakes allows me to learn from them and not do these in the future. Not that this is practical in any way outside of this project and if this was a real-world scenario then the mindset, I would use is to stay focused and stick to the requirements. Everything else can be tweaked and fixed. Keep an open mind to research, if necessary, an understanding of anything that I don’t know the answer to. Make sure that all aspects are being considered such as a pass and failed metric when testing and not just testing for passing.

* 1. **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.**

I cannot sit here and say that I was trying to eliminate bias in this project. My focus was on getting the tests to pass the J-Unit tests and fixing any issues and the coverage tests trying to get that percentage above 80 percent. I would have to say that Bias would most likely be an issue if I was to test my own code because by nature, I am a perfectionist, and it would irritate me not to get something correctly. I may not look through my code and critique it as hard as I would someone else.

* 1. **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.**

It is important to be disciplined in my commitment to quality as a software engineer because a small mistake in code can be life-altering and could cost the company millions of dollars. When cutting corners when writing code is not a very good work ethic and can be noticeable by employers. In my opinion, the only way to avoid technical debt as a practitioner is to continuously practice and continue education and reading. Keeping up with the current coding practices and companies’ code of ethics.

**References**

*Unit Testing and Code coverage in Java*. (2021, December 7). Wyssmann Engineering. <https://wyssmann.com/blog/2021/12/unit-testing-and-code-coverage-in-java/>

*Software Testing Techniques*. (2021, February 22). GeeksforGeeks. <https://www.geeksforgeeks.org/software-testing-techniques/>

*What is Acceptance Testing?* (2015, December 6). Agile Alliance; Agile Alliance. <https://www.agilealliance.org/glossary/acceptance/#q=~(infinite~false~filters~(postType~(~>

*What is Integration Testing? A Complete Guide to Integration Testing*. (n.d.). Www.practitest.com. Retrieved April 12, 2023, from https://www.practitest.com/qa-learningcenter/best-practices/what-is-integration-testing/

‌

‌

‌