Southern New Hampshire University

Alan Palmer

Project Two

12 April 2023

**Summary**

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

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

I approached this project strictly based on the software requirements. I did not write an application based on what I thought the customer wanted. I then formed the classes and methods as directed. There is always a different way to achieve the same result, but the customer defined their requirements for the application that they want delivered to their customers. It is always the best policy to deliver what the customer wants, or they will find services elsewhere. For example, my method structure was as follows:

add()

update()

delete()

**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 trick to providing effective coverage is to force all the failure criteria and then finish with a pass. This allows for all the logic to be tested. My JUnit tests provided for excellent coverage with the lowest being 85%.

**Describe your experience writing the JUnit tests.**

**How did you ensure that your code was technically sound? Cite specific lines of code from your tests to illustrate.**

Building technical sound code requires the developer to accomplish one step at a time to achieve the overall outcome. Each class identifies the required members and the proper access. The getters and setters are formed properly. My experience prior to this course with JUnit tests was zero. I was able to read the tutorials and through trial and error I discovered that specific failures must be observed with a pass at the end. This provides maximum logical coverage.

public void deleteAppointment(String id) {

if (appointment.getId() != id) {

throw new IllegalArgumentException("Appointment ID must be valid.")

}

this.appointment = null;

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

I ensured that the code was efficient but providing what was identified in the software requirements and nothing more.

testTaskService()

testAppointmentAdd()

testAppointmentDelete()

**Reflection**

**Testing Techniques**

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

The software testing technique that I employed was to test each specific requirement individually. For example, I checked for null values and values that were either too big or too small.

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

I did not include processes to test for bogus characters in my testing technique. I also did not test with a GUI which was identified in the software requirements as not a requirement. Both methods would have been useful to fully test the logic from an end user point of view.

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

Both techniques would have practical uses and implications because an end user will ultimately use this software for the purpose defined in the software requirements. If the user experiences errors, the software does not deliver the result.

**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 employed caution at every step. I checked each individual requirement to ensure that the code was efficient and delivered the desired outcome. It is important to appreciate code because you will be looking for work if the product does not deliver the desired software requirements. For example, if the software requirement is identified to check for null values and you overlook this requirement as a developer, the software will accept null values and the customer specifically did not want that outcome.

**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 think it is important to understand that the developer is the code expert. The customer is not but they are an expert in what they want and are paying for. This must be understood by the developer. The customer may have a budget that they are sticking to. Testing your own code is dangerous because you make certain compromises, mentally, on why you code things in a certain way. In the end if the software does not deliver the desired result, it does not matter how efficient it is.

**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.**

Discipline is critical because so many things depend on it. If you are hired as a developer and you produce a bad product, your reputation is tarnished, and the customer will seek out another company to deliver a solid product. If the customer has a customer of their own, their own customers lose faith in the product that they are paying for. This outcome causes your customer to lose their customers. From another angle, what if your product is supposed to monitor someone's vital signs and it fails? Someone could lose their life this way which would most likely result in a lawsuit.