Tierney Shaughnessy

10/19/2021

Professor Cross

CS 320

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.
         1. The unit testing aligned with the software requirements by helping ensure the requirements were met and the input is correct. I tested the code by applying various inputs to make sure that it caught anything invalid.
      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?
         1. I don’t know if I fully understood this part, truth be told. I forgot to turn it in, but even when trying to do it – I cannot seem to get it to Run. It must be user error, but this is an area I don’t feel comfortable defending in all honesty.
      3. 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.
         1. I used objects to validate the input and used proper industry standards.

Example:

**if** (index < 0 && validateAppointmentID(appt.getAppointmentID()) && validateAppointmentDate(appt.getAppointmentDate()) && validateDescription(appt.getDescription())) {

*appts*.add(appt);

**return** **true**;

}

* + 1. How did you ensure that your code was **efficient**? Cite specific lines of code from your tests to illustrate.
       1. I tested each aspect of the input separately to make sure that the input was according to expectations. There is no point in uploading the input to the database if the date, for example, is not correct.

Example:

@Test **public** **void** invalidDate() {

Appt appt = **new** Appt("2222222222", "05/23/2022/23/24/25/26/27 filler", "name is Joe");

addAppt(appt);

System.***out***.println("Size = " + *appts*.size()); }

1. **Reflection**
   1. Testing Techniques
      1. What were the **software testing techniques** that you employed in this project? Describe their characteristics using specific details.
         1. Unit Testing
            1. Tests the smallest unit of code written by providing various inputs. Practically, this can be used for testing objects to ensure that said objects do what they are supposed to do.
      2. What are the **other software testing techniques** that you did not use for this project? Describe their characteristics using specific details.
         1. White Box Testing
            1. Testing where all internal system design is known and taken into account while creating test cases.
         2. Black Box Testing
            1. Unlike White Box testing, Black Box testing does not consider internal system design for testing. Tests are primarly based on the requirements and functionality and are used to ensure that the product behaves in an expected way.
         3. Integration Testing
            1. Testing used to see how the various parts of the code integrate into the system, best used to make sure that all the parts work in the way they are expected to within the system being built.
         4. Functional Testing
            1. Rather than testing small bits of code like Unit Testing, functional testing tests the functionality of the code to see how a module is behaving and ensure it behaves in the way you need it to.
      3. For each of the techniques you discussed, explain the **practical uses and implications** for different software development projects and situations.
         1. (Included in descriptions of techniques)
   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.
         1. I struggled in the beginning of the term due to some family troubles and illness. As a result, I had a lot more to combat than my usual anxiety. I employed caution wherever I could because of this, because I knew I had a lot more on my mind than usual. I would take more breaks, etc. But, I think this also allowed me to appreciate the complexity of the code I was testing because I had to truly stop and take a look at it in a way I might not have had I not been so very stressed.
      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.
         1. I try to limit bias in every aspect of my life. But, when it comes to my work as a future software engineer, I put a lot of work into understanding the ways that Bias can often manifest in coding. Even something as innocuous as the input I put in to test could have bias to it – as if something like say, non-eurocentric names were used as invalid input – the code could not work for people whose names are not Eurocentric. As a result, yes – I do believe bias would still be a concern if I was providing my own code.
      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.
         1. I’ve examined this a lot, as with my ADHD I have to be mindful of things like task management and staying on point. In my career as a practitioner in the field, I intend to apply the methods I am learning as a student. The usual methods to remain disciplined and on task don’t always work for me, so the problem solving I have to do for my assignments is invaluable and irreplaceable. School is giving me time to truly learn through trial and error what works best.
         2. It’s important to not cut corners because while it may be easy in the moment – those little ways you cut corners can and will add up. And then at the end of the coding, you will have to go back and fix a mountain of potential errors.