Demitrius Peoples

CS-320

Summary and Reflections Report Project 2

**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 believe that the testing that will be performed aligns with the requirements of the software well. Task service is a value that should not be null. And many other listed requirements. Addressing the software requirements, I was able to align my test with all of them. My code has met the requirements for ID’s not being updateable after they have been set. Also, requirements for ID’s to be unique I put in a check before assigning a new ID that the random ID generator creates does not match any previously existing ID.

A computer code with text

Description automatically generated

**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?**

I know my tests are effective because I utilized tutors that assisted me with these assignments. Not only that, but I also performed and created specific lines of code to test the requirements given. I tested every function to ensure that all requirements were met.

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

My experience in writing J Unit tests has grown to become extensively knowledgeable. I believe I have learned the foundation of J unit tests and have acquired some novice skills in the overall functionality of the testing systems.

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

I created variables that are set with all the requirements in mind. This shows being technically sound because for all my tests I must call these variables instead of writing the value into each test. This also allows for quick updates to the test if requirements change in the future allowing for longer/shorter versions of these variables. Citation of my specific code is:

A computer code with text

Description automatically generated

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

One of the ways that I ensured my code was efficient is to run it in repeated tests. Instead of writing a test multiple times to check for unique ID, I used a repeated test. This makes the code more efficient and less busy. I ensured my code was good by testing adding a description, deleting of tasks, adding tasks, and many other things. I also avoided writing long methods, and if else statements. I avoided creating big objects and primitive types of code. I understood the task and kept the code precise and focused. The sample of my code to illustrate is asfollows: A computer code with text

Description automatically generated

**Testing Techniques**

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

I used unit testing to test my project. I did this to ensure that all the requirements given were met. Unit testing breaks a project down into its base units to ensure they are functioning as intended. Doing this provides clearer and concise debugging that prevents issues resolving the bugs.

**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 spiral methods, extreme programming methods, and rapid development methods. The verification and validation model method are similar to the waterfall method but looked at as an evolved form of it. Testing happens after each development phase is completed. Simultaneous execution alongside coding speeds of the detection of bugs.

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

Agile methods have a practical use of discovering bugs early on, and recurring testing. This prevents the extensive risk of costly fixes. Waterfall methods divide the project into five key stages with each prior stage relying on the completion of the previous one.

**Mindset**

**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.**

In the case of this project, I only employed a small amount of caution. I did this because the requirements are simple. One of the many reasons for caution is over testing. Complexity can be a large factor when doing testing, more complex code can easily lead to over testing, or a test could introduce a bug unintentionally. Both aspects are also true when code has more interrelationships.

**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.**

The simple nature of the code ensures that review is accurate and unbiased.

With more complex code ego could get in the way and cause bias in review. This can lead to unintentionally covering up a flaw in the code due to belief that tests cover a section of code that was overlooked.

**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.**

Being disciplined in quality as a software engineer is very important because quality helps provide better user experience, better reputation with clients, and finally and possibly most importantly save time and money having to redo work with errors. Cutting corners can lead to defects in the code that can cause security or user experience issues which can sometimes be serious. For example, for medical software cutting corners in code could lead to a security breach that could open the company to costly legal action on top of having to pay to debug code to fix the oversight and the loss of reputation with clients for quality work. Ways to avoid this by trying to keep ego out of the equation via keeping my own biases away when reviewing my work. Also, this part will be somewhat out of my control, working with a good team that communicates and cooperates can make a big difference to ensure quality code in a system with multiple authors.