Shanna Warrington

CS-320 Software Test Automation & QA

10/13/2021

**Project Two**

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

For each of the three features I tried to ensure that all the features that were a part of program was covered. That meant understanding what each feature was being coded to do. Enable to cover each feature and have good coverage during testing I needed to know what each feature was asking the user to do but also figure out what the user might input that was outside the features of the program. In other words, what errors or mistakes might the user input.

**To what extent was your approach aligned to the software requirements?**

Many of the requirements were similar for each of the features. It required features to be a certain length and not be null or blank. Some also needed to have a unique ID that could not be changed. Several requirements also needed to be able to be updated so they could not be permanent. By making use of if-else conditions and while statements I was able to cover the requirements that were been asked. I was able to work through whether a condition was too long or blank then do this or if it was correct do that.

**Defend the overall quality of your JUnit tests.**

Beings I still had some syntax errors in my code, I was unable to have it run green. When running the Junit test for the features it did show that there was significant test to cover the program. I believe with some more work and correction I would have full coverage. With more experience I feel that I would be able to overcome any problems I had with the program.

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

At the beginning of the term, I had no experience with Junit tests. I had never even heard of them. Now that I have used them and seen how they work it makes sense to use them. It allows the developer or tester to ensure that their code is accurate and more importantly works. Being able to test the product all throughout the development stages and up to release is a crucial part of developing a sellable product. Junit testing allows the developer to test many different case scenarios that could happen when the product goes live. This enables them to be preemptive with their code.

**How did you ensure that your code was technically sound? How did you ensure that your code was efficient?**

While my code does still have some errors, I believe that it is still modular and concise. By using if-else statements and while loops I was able to create clean and effective code. With the use of best coding practices my code is clear and easy to understand. This also makes it efficient code when it is easy to follow. Adding in comments to explain what each section does also helps it to be efficient for myself and others. I have included some screen shots of my code as examples.

Graphical user interface, text, application, email

Description automatically generated Graphical user interface, text, application, email

Description automatically generated

**What were the software testing techniques that you employed in this project?**

I felt that the best way to go about the testing was with a structured base. By using sequence, selection, and iteration (Hambling 2015) I was able to get a good coverage on the testing. Using sequence allowed the statements to be exercised one after the other making sure that each one was covered. The selection makes the computer decide if the statement is true or false. While the iteration has the computer running through sections of code more than once. This ensures that each scenario is tested.

**What are the other software testing techniques that you did not use for this project?**

I could have used specification or black box techniques to test my code as well (Hambling, 2015). This technique deals with equivalence partitioning which tests chunks of input together that has similar characteristics. By making use of boundary value analysis which shows that errors tend to group together around boundaries it can test errors more efficiently. There is also experience based technique which uses error guessing and exploratory testing based on a developer’s knowledge and experience. I am still learning, and I did not feel that gave me the years of experience it takes to go this route.

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

In my limited experience I would say that black box techniques would work better and more efficient on large projects. By grouping code together, you do not need to test every scenario. The white box technique that I used would be better for small projects. It allows for 100% coverage due to the developer testing each statement in order. The experience-based technique I believe would let you know better what technique to use and how to incorporate experience into the testing.

**Assess the mindset that you adopted working on this project. To what extent did you employ caution? Why was it important to appreciate the complexity and interrelationships of the code you were testing?**

My mindset to begin with was timid. I wanted to have everything perfect and be at 100% coverage. As I was working on the project, I realized that at my level of skill and experience it wasn’t going to be perfect, but I was still determined to do my best. I tried to be cautious and make sure that I was covering every scenario I could think of. While this would work in a small project such as this, I do not believe it is achievable with larger projects. Appreciating the complexity and interrelationships of the code enabled me to better understand the code and how each requirement could play off each other. Looking back instead of testing each little thing separately, I could have grouped pieces together to make it easier to follow and code. For example, instead of coding the length of a name and if it is null separately, it could have been coded and tested together.

**Assess the ways you tried to limit bias in your review of the code.**

I tried limiting bias in my review of the code by not being so hard on myself. I am more likely to tear apart any code I create and not feel that it is good enough. For developers testing their own code though, I imagine they would think the code was fine and not put enough effort into testing it. That would allow defective code to be released live.

**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?**

Having a high standard of quality is very crucial to being an effective professional. Without those standards you would release defective, subpar products and more than likely not make it very far in the industry. When you cut corners or not finish the product quite up to standard, you are showing that you don’t care about the development world or your clients. It also shows a lack of pride in your self and your work. A way to avoid technical debt is to keep the client involved in what has been done and where the product is in the timeline. I feel that open lines of communication can save a lot of headaches. It may also be able to work with the client for an extension if they know how much work you have been putting into the project and that the extension will allow a better product to be released.

**Cites:**

Hambling, Brian Morgan, Peter Samaroo, Angelina Thompson, Geoff Williams, Peter. (2015). Software Testing - An ISTQB-BCS Certified Tester Foundation Guide (3rd Edition) - 4.1.2 Self-Assessment Questions. BCS The Chartered Institute for IT. Retrieved from

https://app.knovel.com/hotlink/pdf/id:kt00UC2IY1/software-testing-an-istqb/test-desig-self-assessment