Daniel Jones

CS 320

Project 2

December 15th, 2024

Summary

Unit testing is important when it comes to successfully building software solutions. It helps reduce the risk of completing sections with coding errors and improves the overall quality of the code by quickly finding errors and helping the coder address them. The unit testing I used for the three features was similar as I made it align with best practices and within the software requirements. A major example I can think of was during the contact object. It had several restrictions to what an acceptable parameter like was being 10 characters long, shall not be null, not be updatable, and to require a firstName and lastName. It took me having multiple appropriate test parameters to be able fully understand how much time and energy the tests were saving me. I could also have these tests in a repository so that I can use them in future projects so that I have a faster and easier time testing in future projects.

The contact object has a service that does some crude operations that were all specific requirements for one of the three features. All other features had similar requirements so that the software must be implemented with those requirements to pass the unit tests. The overall quality of my unit tests I feel like improved as I referred to each of the previous tests when creating new tests. Each field and operation were checked, and the validity of the test cases were over 90%. I feel like my testing was thorough and collectively improved my though process when designing code in the future.

My experience with the Junit test was at times frustrating as I referred to previous tests and what worked for one project did not implement the same on another. That being said, I think back at it as a positive experience because I know I still have a lot to learn, and I tried to implement what I knew ensure the code was technically sound.

A screen shot of a computer code

Description automatically generated

The line of codes above was checking the validity of the parameters for the contact class, and I also created a test case that was checking to see if the testID was too long. This was an effective way of checking if the code was efficient.A screen shot of a computer program

Description automatically generated

Reflection

I used the Dynamic testing method which involves testing the behavior of the software. This technique was used to check the various dynamic variables to find the weaknesses of the logic in the code during computation and runtime. The other software technique that was barely used was static testing. This technique is a way of testing code without compiling or executing it. The major difference I have noticed between the two is that static testing makes sure the logic makes sense while the dynamic testing method is used to make sure the code AND logic makes sense by testing what is allowing to be stored in the objects.

The mindset I took while working on the various projects was repetition and verifying that all the testing conditions I needed had been met. I thought of it as a check list for each object making sure each parameter that the client required was met and that my logic was sound. If there was an issue or something I missed, I would refer to the checklist that I had made so that I could maintain the high level of polish on each project. I did have to thoroughly understand the client requests for the code and how each variable was being passed through to properly implement the restrictions and tests properly.

I would regularly take breaks during my code for review to help avoid bias by the famous sunk cost fallacy. I mean that when I took regular breaks, it prevented me from being exhausted or refusing to go back and revise and rewrite code because of the time I had spent. This also comes into play with self-discipline. Discipline is a must for any computer scientist. It is important to not cut corners by having a high level of discipline that will ensure that you are always able to put another foot forward and can also move to tackle any problem that may arise. Writing and testing thousands of lines of code per professional project can be extremely taxing, but if it is not handled correctly it will lead to the loss of revenue, loss of clients, and loss of work.