Josh Cantu

CS-320

4/20/2024

Module 7: Project Two

The normal approach that I took before this course was testing code as I wrote it. Even now, I still test the code to ensure everything is running correctly but I’ve got more tools under my belt to test code properly and professionally. During this course, I learned the type of testing I was doing was aligned closely with dynamic testing. I did try to do some static testing which is applied before developing the code. I would plan how I envision the code running begin the framework and look over the base of the build before continuing further. After running the test code, it would provide me with the results. The higher the results more parts of the code ran successfully. I’m not the best coder and I still have a long way to go to be the type of developer I see myself, but I like to think I did a decent job creating this.

I don’t consider myself a very good developer. I did enjoy creating the code for testing units. All the test codes required “@test” and “@DisplayName”. You could create the requirements for the test that would exist in the code. Run the test code and get the results. I would do my best to create in a way that would make sense to me and for the code to run. I also wanted to make it effective. I wanted to keep it short and only run what it needed to run. I ensured I hit part of the code that was required in the rubric. Parts like ensuring part of the code wasn’t more than 10 characters long etc.

I would focus more on the dynamic testing portion of the code. I wanted to ensure everything was running smoothly as I went through development. I tried static testing since I never really took the time to plan my development in the past. I found it helpful to come up with a game plan to ensure the code looked universal and had a solid foundation. I did take some time out of the actual development of the code, but it was worth it. I like to plan most things in my life, and it only made sense to apply it to something like this.

I did not try black box or white box very much. I didn’t see a reason to implement something like this into this project. Since it was a solo, this type of testing isn’t required. Well after reflecting, it could be. I could have acted as the testing group and tried to replicate a real-life scenario. Of course, I realized this after project one was submitted and I’m writing this reflection.

The listed techniques have their place. I think static and dynamic will always have a place regardless of the software being developed. Since static testing is used more for planning and building a foundation there isn’t a reason to not use it. Having a standardized code will make developing and reading the code easier. Dynamic testing should be used when creating the code. The code is run for functionality and to try to break it. It finds shortcomings early and allows you to patch them before you get too far into development. The white and black box could be used in a solo project, but you would need to get creative.

Since this testing was all new to me. It’s safe to say that I was cautious about developing everything. I’ve never stepped into the shoes of a tester, nor have I seen their workflow. This was new territory for me and I like to think I did everything a tester did, or at least closely. This was all my interpretation anyway. I do find it interesting how the code and the test code work together. I don’t think the code I generated was overly complex and it is easy to follow. When including multiple things, I find this best. Especially so since working with testing plugins I’ve never used.

One of my biases is not being very good at this craft. I may review my code and think I did a halfway decent job. It could turn out that I did not follow any standards. My bias is that I did a good job regardless of my confidence or skill level in this development. This especially applies to the testing portion. I’ve never done any sort of testing code development. With that being said, I could see how this could affect me if I was testing my code. It would be hard to set aside my own biases when testing my code.

Maintaining professionalism when getting test results is important. It should not be taken to heart what the testers find. I know that could be challenging because you could feel that they are attacking you, but they are looking out for your best interest. I would like to think the testers would be able to tell if you are cutting corners. Cutting corners would just hurt you and the client. Cutting corners allows holes and bugs to be made which can turn into vulnerabilities. The foundations should be the base and follow any standards when developing code. Don’t take what the testers have to heart, just reevaluate and address what they find.