Kayvon Ghadiri

Southern New Hampshire University

CS 320 Final Project

August 14, 2022

CS 320 Project 2

Unit testing always plays a pivotal role in software creation. Reducing errors, fixing potential bugs, and strengthening the code are all very important pieces that are often overlooked. In a world where deadlines rule all decisions, often times testing and editing is overlooked and seen as something that can be fixed via an update. In this class and this project, we worked on unit testing and emphasized its importance in the world of software solutions. A good example of this was with our Contact Service code. In this code, there were specifications that had to be met like a unique contact ID string that cannot surpass 10 characters. The contact ID string also needed to require a firstName and lastName. All of these objects could not be more than 10 characters and could not be null.

The Contact object also contained various requirements necessary and if not implemented would not pass the unit test. In the end, my unit test was good but I struggled with ensuring my code was in the correct format as I do not believe they were able to be tested. I fixed this error in my second assignment and later in my project as well. At the end of the day, although it was difficult and sometimes frustrating, I had a good experience with the Junit testing and I learned a lot with even more space to learn more. I ran some tests to ensure the code was effective and efficient, which thankfully mine was.

All in all, I took many things from this project that I will store in my repertoire and continue to use in the future. One of those was dynamic testing. A good test for dynamic behaviors within software code is always valuable and I was also able to use this style of testing to check dynamic variable behaviors and identify any weak areas within the software. Another helpful technique that I used to help my adventure in a few cases was static testing of code without the execution of the code. This was a different style than I am used to, but I see how it can be useful. In the end, I felt that dynamic testing was easier to use and identify weak points while static testing was a little more difficult. Using these techniques, I was able to scan for dependencies and inconsistencies within the software model.

How you look at code and interact with code is pivotal in understanding it and within this project my mindset while looking at this code changed. After looking at the test and the results, it drastically changed my mindset and made me realize what I needed to work on. The code that was tested was far more convoluted that I originally thought on my first glance. The class object requirements and other specifications really changed the aspect of the code and this was all revealed within the tests. Without understanding what each part of the code is doing, the tests are not as successful in reaching their end goal.

When it comes to reviewing your own code, the first thing that comes to mind is bias. The person who wrote the code will always be confident in what they wrote and believe that their code is correct with no errors. That is why I always believe a third party should review the code. That was the code is reviewed by a fresh perspective. Not only can dependencies and inconsistencies be seen, but the 3rd party reviewer can also look at implementing new techniques into the code that the original coder never thought of.

Discipline within code is also important as this environment is always meant to be professional and with as little amount of errors and bugs. Although sometimes software engineers see the need to cut corners to save time and meet the deadline, they should not jeopardize the quality of the code. At the end of the day, pushing a deadline or release date will always be better than releasing an unfinished or bug-filled product.