Project Two

Jordon Burris

Apr. 18 2024

SNHU CS – 320

My code and testing aligned with software requirements by following the requirements that were presented. Like, making unique ID’s, being able to add and remove ID’s, and making sure the ID’s have a certain number of characters and some form of validation. I saw great effectiveness in the JUnit tests, the tests would verify my code and make sure the parameters were set correctly according to the customers’ requirements.

Looking at my code in AppointmentTest.java, lines 20 – 24, this is testing the parameters of the code looking at the description and date. These lines set the description length too long to verify you cannot add too long of a description and also uses a date that is past date.

The main forms of testing I used were of course the JUint tests and I also did some manual testing, I feel that manual testing will give a great sense of how the program will run and being able to manually step through the code is very effective. The JUint testing tests individual units of the code to make sure it is functioning properly, doing these tests are great for very detailed testing such as setting and testing parameters of the application.

Some of the tests I did not do were security testing and performance testing. These types of tests, I feel would be done in the later part of the development process and would not be applicable to this application. Security testing would be used for a program that would be near end of base development and would have user interaction that could possibly put information out for unwanted users. Some ways of security testing would be some sort of pen testing or stress testing, and some ways to combat that would be keeping things up to date and implementing secure code. I did not do any performance testing because the program was still in the initial development phase. Some ways to ensure performance are to find and eliminate bottlenecks and make sure the application can handle the number of expected users.

I kept a very open mind doing the testing, not really knowing what to expect, but I feel things went better than expected. I employed a decent amount of caution because I did not really understand JUint testing before this class, the only forms of testing I have done was manual testing. It is very important to appreciate the complexity and relationships of the code because it will help you understand exactly what you are testing and will allow you to understand that simple changes in one area of the code may affect something else in the code as well.

Bias can be an issue when testing your own code because if you spend a lot of time on something and work hard for it, you might be less inclined to say it has issues. I kept my bias away by understanding there are code requirements, and they need to be testing to make sure the entire code will function properly. A good example of this is we have some equipment at my job that needed an update, and someone wrote the code for the update and did not test the code entirely because they felt like if they did not report any current issues with the code then they would look better to their boss. Which was not the case when the production server came on and the application did not work as planned.

Being disciplined in regard to quality is crucial, this will ensure your code works properly and no short cuts were taken. Cutting corners will only cause issues down the road, not only for me as a developer but also for other people on the project, having the integrity to not cut corners will help everyone in the long run. The main ways I intend on staying out of technical debt would be to verify my code, not cut corners, and stay up to date on best practices. This will ensure my code is effective and not rushed to save time and money and to be a more effective developer.