Tashi Anderson

CS-320

August 18, 2024

Project 2

Summary and Reflections

My unit testing approach was closely aligned to the software requirements by constantly checking and verifying that my code worked as needed, where for example if a description couldn’t be over a certain number of characters, I made sure that the character limit was set in place as well as output when the limit was surpassed. Another example of being in line with the requirements was in evaluating all program files, which included the, for example, appointment test file and appointment service test file. This did prove to be, however, quite time-consuming as many of the test requirements were like each other. The quality of my Junit tests was in-depth and high-quality, which is based upon the extent of which the testing covered. In all test files, I made sure to test every possible input and I even made sure to break the program by having null values.

I ensured that my code was technically sound by creating both getter and setter functions for all files. This helps in ordering and keeping track of which variable should be updated and which variable should not be updated. This is especially beneficial when dealing with both public and private variables, such as in the appointment java file which allows for an appointment id to be set, but not updated. I ensured that my code was efficient by having all files be brief, which means that debugging is easier, and the code can be maintained for the future. This is seen in my appointment service code, where it had only about thirty-six lines of code!

Some software testing techniques I have employed for each of the milestones were dynamic testing, static testing, and even manual testing, where I went into the code and hand tested each line myself. Static testing is where you test software without executing code, and dynamic testing is where the code is executed. To help with these tests, the Junit resource was used.

A particular testing technique I did not use in any of the modules was a testing platform named TestNG, which is inspired by Junit and NUnit, that is said to be more powerful and easier to use. (“TestNG Documentation,” testing.org) It helps in test automation and the organization of test cases. I believe if I were to expand on the several types of testing classes used for my code, I would be able to have a deeper understanding of the different software testing techniques and why it is so important.

One practical usage for both static and dynamic testing is that it ensures the quality of the software is up to par, eliminating any errors down the road that may cause financial issues. These testing techniques can be used in any step of the software development process, which in turn allows for the code to be easily maintained and updatable.

When coding for any large-scale project, you must always use caution. This is because the technical requirements can change in an instant, which can push back any progress made. Also, it is immensely important to understand exactly what is being evaluated and why. For example, assessing the appointment and appointment service code only makes since when it is tested from the perspective of someone trying to make an appointment for something. Understanding that there is a possibility of human error, as well, can affect why a program needs to be tested. For example, what if someone wanted an appointment in 2025 but they accidentally input 225? For that possibility, there needs to be a fail-safe within the code to catch this error and allow for a revision. This issue also works hand in hand with understanding potential bias within testing the code, as not everyone that is interacting with the code that you made understands what the code is supposed to do and how. Understanding this bias helps when evaluating your own code.

Finally, ensuring that you as a tester, as well as a programmer, are disciplined in your commitment to making quality code is important when releasing and/or aiding in the release of content to your userbase. Making sure that no corners are cut during the development process or overlooked helps with the trust you have with your clients. If a client does not trust you, then they will not use your software. This discipline can also extend to the user, as the user should be able to submit feedback on the program they are using. This feedback can help in ensuring that the code you make is accessible to all users and can be updated to reach a wider audience.