**7-2 Project Two**

James Lightner

Southern New Hampshire University

CS-320 Software Test, Automation QA

Professor Ramsey Kraya

12/13/2024

Ultimately, there was a lot that went into testing for project one. I wanted to ensure that my approach to unit testing was aligned with the specific requirements set forth to us. The tests that I created were designed to validate the functionality of many critical features, including contact management, how tasks were scheduled, and how the user created appointments. These tests ensured that all the features met the clients’ expectations. I wanted to make sure that all the test cases were mapped according to specific functional requirements to see if it was adequately running.

The quality of the JUnit test was important to the overall success of the application. The goal was to achieve a 90% coverage but that is not always so easy to achieve. I believe that 90% is the best one can do because I think in a test you cannot account for everything, but it gives the development and testing teams the confidence this is a strong product. This kind of coverage is an in-depth analysis on any edge case scenarios whether it is a surprise or expected.

This is the first time I have ever written JUnit tests, so I was definitely navigating unfamiliar waters. To ensure my code was technically sound, I just wanted to make sure that it was simple and straight forward. I wanted to make sure that proper testing structure was used and that it was logical to the requirements we were required to do. On top of being technically sound I wanted the code to be efficient too. I did this by making sure that parameterized tests were utilized to reduce any test duplication. This allowed multiple inputs to be tested within in one method.

I applied various software testing techniques to ensure that it is tested and functional. For each of these milestones I utilized unit testing, integration testing, and boundary testing depending on each milestone requirement. All of these allowed me to make sure that the components worked together and identify any issues that needed to be addressed. Although I used many techniques to test the software that was developed, I also did not use many either. For instance, I did not use regression testing. This could have provided additional insight into how the software works and its efficiency.

Each testing technique has obvious implications for software development. For example, unit testing has quick identification of issues within the code. Doing this can save a significant amount of time debugging it later. Integration testing makes sure that the modules communicate effectively. Omitting other kinds of tests can potentially put the future of the software at risk of being deployed with unknown flaws.

In terms of adopting a certain mindset for developing testing I definitely leaned more on the cautious side. I think that played a crucial role in me trying to identify any issues between the three different modules. For example, making sure that a task that was associated with a deleted contact would also be removed too. This took a slow methodical approach to make sure all of these things were considered and tested. I also believe that a cautious mindset also aides in the maintenance and overall integrity of the application too.

Limiting bias can be extremely difficult, especially if you are testing your own code. This is why it is best to have a developer and a tester. I do not believe that it should be a dual role kind of situation. The way I tried to approach this was view the application as being created by another team all together. Attempting to separate myself in this way allowed me to be impartial to the code as best as I could and lead to a critical evaluation.

Maintaining discipline while testing an application is paramount. To maintain discipline, you want to make sure the code is meeting up to date coding standards and practices. You also want to ensure that it is written clearly and maintainable test cases have been established. For example, you want to make sure that shortcuts were not taken. You do not want to skip test such as error-handling or hardcoding values for a specific test being executed. Being disciplined will only improve the quality of the test. Not only does this make the test of the application reliable but also the test will establish a professional and reliable reputation as well. This kind of mindset will only aide in future tests as one will gain more experience.

In conclusion, the unit testing approach utilized for this project was in depth and as closely aligned with the requirements set forth for it. By deploying a multitude of testing strategies and maintaining a disciplined mindset, we were able to ensure a robust examination of the application. Although it is recognized that the addition of more advanced testing techniques will only enhance future projects. The lessons learned in this project have been invaluable as I look to strengthen my skills as a developer.