**Project Two: Summary and Reflections Report**

Scott Mansfield

Department of Computer Science, Southern New Hampshire University

CS-320: Software Test Automations & QA

Kalysa Wilson

June 17, 2023

**Summary**

For project one I was tasked with creating three different objects and three different services to manipulate those objects. Each object and service had a distinct set of rules they had to adhere to, and it was my job to produce code that respected those rules. My approach to unit testing the objects themselves was straight forward and the unit tests were able to directly align with the assignment parameters. If you look at the JUnit test files for each object (TaskTest.Java, ContactTest.Java, and AppointmentTest.java) you will see I was able to code and run a test for each parameter of each element within each object. Compare the tests run in those files to the Project One Guidelines and Rubric (n.d.), and you will see a one-to-one correlation to the software requirements. Writing the unit tests for the object services was different upon first approach, as I originally created one unit test that encapsulated all the requirements for each object service. However, in the end, I wrote an individual test for each requirement of the object services that aligned directly with the assignment objectives. If you look at the TaskServiceTest.java file you will see a test to add a task, update a task and remove a task as requested in the Project One Guidelines and Rubric (n.d.). Overall, my JUnit tests were effective in verifying the overall functionality of the code and its alignment to the assignment objectives. An individual test was run for each parameter of the assignment and if you were to run coverages as JUnit tests, you will see that my JUnit tests obtained 100% coverage of my code.

During my time coding these objects and services, and prior to writing my unit tests, I verified that my code was technically sound by adding code to create an ArrayList, and then Add, Update, and Remove objects from the ArrayList. Throughout this code I used “System.out.println()” functions in order to “show” my work as I went. That mock testing code was then removed when it came time to write the unit tests, however I wrote similar code in my unit tests. If we look at TaskServiceTest.java again and see lines 16-20, you can see that I create a new ArrayList object, then I create a new Task object, I add the Task object to the ArrayList and then verify that it was in fact added to the TaskService object. During the initial writing of my unit tests, I also added “System.out.println()” functions to “show” my work as I went along but those were removed prior to submitting my finished code. As far as efficiency is concerned, if you were to review the same lines of code, you will see that it only takes three lines of code to add a Task object to the list. Furthermore, if you look at the rest of the code in the TaskServiceTest, you will see that once the list is created and an object is added to the list, it only takes one line of code to remove or update that object. Looking at the TaskService.java code, the actual functions written to add, update or remove an object only consist of two to six lines of code. I was able to write this software with minimal code and therefore, minimum processes to get the job done right.

**Reflection**

There were three types of testing techniques that I employed while working through the code of this software. Those techniques were functional testing, unit testing, and manual code review. Functional testing is a high-level of testing used to verify the application will output a specific value given a specific input (Pittet, n.d.). When I went through, created mock objects, and added, updated, and removed them from the arraylist’s, as well as printing the results to the console, this would be considered functional testing. Think of it as making sure the application “functions” as desired in a semi real world scenario. Functional testing also implies you have enough of the code written to perform these more global functions. Unit testing, however, is considered a low-level type of testing and is used to test individual functions, methods, classes, components, and modules (Pittet, n.d.). Unit testing can be done much earlier in the development phase, and it is what we used to test the various parameters of each element within our object and service classes. Because we had not coded any type of user interface or any way for a user to input data to the application, this allowed us to thoroughly test our functions and objects. Manual code review is often bunched together with the security testing side of things, and it is used to verify that coding best practices were used, code is clean and free of errors. Once I finalize the functionality and efficiency of my code, I go back through and delete unnecessary things, make the “whitespace” and indentations consistent and overall make sure the code looks clean. It should be noted that although I did not specifically mention integration testing, some of the unit and functional testing I did could be considered integration testing.

There are many distinct types of testing that I did not implement for this assignment. End-to-end, acceptance and performance testing are a few notable ones. Although these three tests are aimed at different aspects of an application, they all require the application to be more or less complete with a user interface. End-to-end testing is meant to replicate a user’s behavior and workflow through an application (Pittet, n.d.). Acceptance testing is done to verify the application meets all business needs (Pittet, n.d.) and this usually encompasses local/federal regulations as well. Performance testing is stress testing the application to weed out performance issues such as responsiveness or other bottlenecks (Pittet, n.d.). There are also various testing techniques used to verify security and compliance that I did not use as well, however these are out of the scope of this software project and report.

For this assignment I really had to juggle the mindsets of both a software developer and a software tester and to keep reminding myself that often these are separate jobs in a professional software development scenario. When I first started creating the first part of this project (the contact service), I did not really know how to handle the fact that we were not tasked to create a UI or any way for a user to input data to the application. This hindered my code at first but after reminding myself that a software developer in a professional setting will often be tasked with writing just a single object or a single class and nothing more. For that first object and class, because my coding was hindered, my testing and testing mindset was also hindered. In fact, I was not originally able to write a unit test for the Contact Service, and my only testing for that code was functional testing. When trying to fix my code to unit test it, I had to be cautious and eventually gave up on unit testing it altogether for fear of breaking the code completely. Soon after that, while working on the task service, I was able to get in the mindset of a software developer as well as a tester. I could write more direct and efficient code that only did what was required and nothing more, then write complete unit tests to test it thoroughly. Although my code functioned as expected and was tested thoroughly, I still had not obtained 100% coverage. By the time it came to finalize the complete project code and test it, I was able to clean up my code completely, and write more thorough tests in order to get 100% coverage across the board.

In this situation limiting bias is especially important and I attempted to limit my bias in a couple diverse ways. The first was my usual way in which I try and find bugs through functional testing. Instead of just accepting the code works with expected input, I attempt to feed it unexpected input to break the application. The second way I attempt to limit bias is by not accepting that my code works the first time around and doing everything in my power to make it more efficient and cleaner. This goes for the testing as well because I have learned the better the code is, the more thoroughly I can test it without it breaking. If I am going to be the one to test and quality check my own code, I must be more disciplined and diligent when verifying and testing my code's quality and functionality. Spending more time and effort in the testing and quality assurance of my own code will also limit technical debt in the future.

**References**

Southern New Hampshire University. (n.d.). *Project One Guidelines and Rubric.*

<https://learn.snhu.edu/d2l/le/content/1311909/viewContent/24289735/View>

Pittet, S. (n.d.) *The Different Types of Software Testing.* [https://www.atlassian.com/continuous-](https://www.atlassian.com/continuous-delivery/software-testing/types-of-software-testing) delivery/software-testing/types-of-software-testing