**Summary and Reflections Report**

Carl LaLonde

Southern New Hampshire University

CS-320: Software Test, Automation QA

Professor Samuel Haney

April 19, 2025

While working on Project One, JUnit tests were created for the Contact, Task, and Appointment services. The main approach that I followed was creating tests for each class’s main features such as: creating new objects, getting and setting input values, and making sure that the requirements for each field were not being violated. For example, in the ContactTest class, I tested to make sure that the phone number field had to be exactly 10 characters long.

A screen shot of a computer code

AI-generated content may be incorrect.

A screenshot of a computer error message

AI-generated content may be incorrect.

This feature was taken from the requirements showcasing my understanding and testing based on what the customer needed.

My testing approach was based upon the software requirement that were given. The requirements for all three services said that the ID should be unique and should not be able to be edited. Other requirements stated that names and descriptions should not be over a certain number of characters, and for the appointment services, appointments should not be scheduled in the past. I wrote test cases the check for those constraints. In AppointmentTest class, I wrote a test that if an invalid past date was used, it would throw an error.

A computer screen shot of text

AI-generated content may be incorrect.A computer code with colorful text

AI-generated content may be incorrect.

These kinds of tests prove that my code is doing what it is supposed to do according to the requirements set before me.

I believe the quality of my JUint testing was strong. I made different test cases that checked for valid input as well as invalid input. In TastTest class, I created separate tests that tested for a valid name, null name, and a name that would exceed the character limit.

A computer screen shot of a computer code

AI-generated content may be incorrect.

A screen shot of a computer code

AI-generated content may be incorrect.

A screen shot of a computer error

AI-generated content may be incorrect.

I did not use any specific tool for calculating code coverage such as EclEmma, I believe that I tested all of the methods and fields in each service.

To make sure that my code was technically sound and efficient, I kept my tests simple but focused. For each method that I tested, I created different tests to check for different circumstances. For example, in TaskTest class, I created different tests to check if the description field was either null or too long. This also ensured that when a test failed, it was known why that test failed.

A screen shot of a computer code

AI-generated content may be incorrect.

A screen shot of a computer

AI-generated content may be incorrect.

I also used a clear and understandable naming system for each of my tests so I knew what each test was checking for.

For this project, I used unit testing through JUnit for my software testing technique. Unit testing primarily focuses on testing individual pieces of the code, or individual methods, to make sure that they work properly. I wrote JUnit tests for the Contact, Task, and Appointment Classes to make sure that setter methods and input validation was correct. I used an AssertThrow() to make sure that when trying to set a null value to a required field, an exception would be thrown. A computer screen shot of a program code

AI-generated content may be incorrect.

Unit testing was helpful because it allowed me to detect bugs or make sure that the code was doing what it was supposed to do early on. It also allowed me to easily find any piece of code that might have been incorrect. Unit testing is good to use during development, when creating and fixing individual parts of the code.

Some techniques not used for this project include integration testing and system testing. Integration testing checks to make sure that different parts of a system work well with one another. For example, if Appointment Service were to utilize a database, I would need to check to make sure that information is correctly stored, updated, and pulled from the database. System testing would test the whole application in real world scenarios. This testing technique is used to make sure that the system meets user requirements. Since my created services were back-end for this project, systems testing was not necessary.

Thinking like a software tester, I had to be cautious because small mistakes, such as letting an invalid phone number be accepted, might cause issues within the program later. In testInvalidPhoneNumber() method, I tested what would happen if the phone number was too long. A computer screen with text

AI-generated content may be incorrect. Not including a test such as this might have allowed bad data into the system.

When writing one’s own code, it’s easy to assume that the code that you created is going to be correct. This mindset can lead to costly errors. I tried to think about how a user might put incorrect information into the required fields and tested for both valid and invalid input types. I created tests to check for invalid input to make sure that the program handled any mistakes correctly. I think it is a challenge, and requires a special mindset, for a developer to test their own code without assuming it will be correct the first time.

If a developer begins cutting corners just to finish their work earlier, such as skipping tests because they assume everything is fine and dandy, it could cause costly errors in the development cycle. Any bugs that are not found early could take longer to find and fix later on, as well as adding on additional costs. Users, as well as any stakeholders, might lose any trust in the software and/or the development company. Integrating these testing techniques will allow me, as a developer, any technical debt by writing clear, testable code. Writing tests while I create the code, instead of waiting till the end, will allow me to catch any mistakes immediately.