**Summary and Reflections Report**

Alex Mizway

Southern New Hampshire University

CS-320: Software Test Automation and QA

Professor Joseph Rangitsch

February 19, 2023

**Summary**

My approach to the JUnit tests for the Contact and Task classes were aligned with the requirements given within the rubric. For example, the ContactService requirements are as follows:

1. The contact service shall be able to add contacts with a unique ID.
2. The contact service shall be able to delete contacts per contact ID.
3. The contact service shall be able to update contact fields per contact ID. The following fields are updatable: First Name, last Name, Number, Address

These requirements were the first thing I worked on as without a class to base the tests off of, there is no tests available to run. I worked forwards and backwards when programming the Contact classes as I initially had a hard time figuring out the JUnit words like the annotation @Test and the line fail(“example”); When I first started working on the programming, I had to really dig into what keywords JUnit used and what they produce when properly implemented. By doing that I can guarantee that the testing requirements are met.

I can guarantee the quality of my JUnit tests as I personally ran them each and every time I made an edit to the code while writing it. I did not submit the projects until I saw that the JUnit tests were successful in every aspect and then honed the code down to the simplest I could possibly make it.

The best example I can show that ensures my code is efficient is how I kept the projects within 100 lines of code from top to bottom with the average length of code being 71.5 lines of code. Another way I can ensure efficiency of the projects is to continuously learn new and efficient ways of coding. As an example, I recently learned how to use UUIDs or universally unique identifiers as part of a way to generate unique taskIDs for the TaskService class:

private String newUniqueId() {

return UUID.randomUUID().toString().substring(

0, Math.min(toString().length(), 10));

}

This snippet of code shows how with 4 lines of code, I can generate 10-character IDs for tasks that are generated.

**Reflection**

I constantly employed caution in my development and testing of the code that I was working on throughout the last 8 weeks. I am always hesitant to push code through as if there is an error that I miss, I will be liable to accept the consequences.

Luckily, due to my overzealous caution, my bias is acceptable as my bias advocates for my coding projects being buggy and unacceptable leading me to work harder and more consistently on the projects.

It is extremely important to be disciplined in our commitment to quality as software engineering professionals because it will lead to better coding practices being pushed from teacher to student and helps keep the projects more uniform.