**Project 2 – Summary and Reflections Report**

**Section 1: Summary**

**To what extent was your testing approach aligned to the software requirements? Support your claims with specific evidence.**

My testing approach aligned with the software requirements to the extent that the approach was based directly off the requirements document. For instance, the task class requirements started with “The task object shall have a required unique task ID String that cannot be longer than 10 characters. The task ID shall not be null and shall not be updatable.” Therefore, my code included tests that make sure the task ID is as expected and that it isn’t too long, with two separate tests “getTaskIdTest()” and “TaskIdTooLongTest()”.

**Defend the overall quality of your JUnit tests for the contact service and task service. In other words, how do you know that your JUnit tests were effective on the basis of coverage percentage?**

Because the given parameters and requirements were all given out in bulleted form, it was not difficult to ensure that our service classes were coded in a way that prohibits invalid inputs. From there, it was just a matter of making sure every key point from the bullet point is tested. I believe by doing this, my JUnit tests have practically 100% coverage for each of their respective service classes.

**How did you ensure that your code was technically sound? Cite specific lines of code from your tests to illustrate.**

My code was ensured to be technically sound in two ways. First, I used best coding practices and explained via comments anywhere that I veered away from them. Second, I incorporated feedback from the first assignment module (Contact Service) into my second assignment module (Task Service). For instance, per the comment on the testing classes in my Task Service assignment (“//per feedback tried to make use of @Test on a more "singular" level”), I made sure to separate my tests out more so in the Task Service to make sure it’s easier to find errors that are created within the test. It’s better to have many tests that will fail for a specific, easily identifiable reason than it is to have an “assert all” statement that tests a pile of things and it’s harder to find an issue within. This is something I’ll continue to work on moving forward as I was initially drawn to the “assert all” functionality as making use of it allows the use of a title field, making the actual tests look better when they’re run in my opinion.

**How did you ensure that your code was efficient? Cite specific lines of code from your tests to illustrate.**

Similarly, I did this in two ways. I made sure to follow standard efficient code practices whenever possible and explaining via comments any time I didn’t. Second, I was fortunate to receive some feedback from the first assignment (Contact Service) that also related to efficiency, so I made sure to do my best to keep that in mind when completing the second assignment (Task Service). For instance, in the TaskService class, I have to run a check to make sure the task exists before returning the taskList. Instead of using a while loop, I was informed in the feedback that the for loops are more efficient, I therefore used a for loop in searchForTask(String id) to catch my exception that the task doesn’t exist (line 16 of TaskService.Java).

**Section 2: Reflection**

**What were the software testing techniques that you employed for each of the milestones? Describe their characteristics using specific details.**

As with any software testing, I started each class with identifying the scope of the testing. For these assignments that was fairly easy as the exact requirements were given from the start. Then the testing type was identified to verify that requirement – typically Equivalence Partitioning in the case of these assignments. Based on the requirements given, we know was constitutes a valid and invalid input so we test each and verify that it works for the valid and throws exceptions for the invalid.

**What are the other software testing techniques that you did not use for the milestones? Describe their characteristics using specific details.**

I rarely if ever used Boundary Analysis. Because the bounds of the item themselves is set within the constructor (say a string length of 10 for example), if the constructor and setters are tested with items of any number under 10 and any number over 10, it is safe to assume in this case that the boundary items do not need to be tested. Additionally, the requirements in these assignments had only whole-number boundaries on lengths of strings which are not typical to cause issues at the boundaries. This technique is far more effective for technical item testing when dealing with floats against a system that can be very specialized (like measuring the resistance of a board/system or the throughput of a network device).

**Assess the mindset that you adopted working on this project. In acting as a software tester, to what extent did you employ caution? Why was it important to appreciate the complexity and interrelationships of the code you were testing? Provide specific examples to illustrate your claims.**

In testing, caution is most easily practiced by being thorough and calculated. I employed caution in this project’s testing sections by being meticulous about making sure each statement and item within the bulleted requirements document was tested to do (or not do) exactly as stated. I did this using an iterative technique and going through them one by one. You can see this in the construction of the testing functions, the last few “words” of the functions will describe what they function is testing and the first few “words” describes which function or class it is testing. By setting them up like this, it was easier to cautiously proceed through the lists and make sure the testing coverage is beyond adequate. As an example, there is “TaskIdTooLongTest” and “TaskIDNotNull”, etc.

**Assess the ways you tried to limit bias in your review of the code. On the software developer side, can you imagine that bias would be a concern if you were responsible for testing your own code? Provide specific examples to illustrate your claims.**

It is imperative to combat bias at all stages in the software development life cycle. As training developers, we have been working on and grappling with this task since the first class we took. I always try to limit my bias as much as possible by following the guidelines or requirements document that is always given or generated at the beginning of the project. I try to assess different ways of seeing the data and different interpretations of what that data could *mean* to different people. Within this project the potential impact for bias was minimal and the mitigation of it was relatively speaking easy – we were given a bulleted list of exactly what the code was to do and what about the code was to be tested. As a result, I tried to break my testing down into each component that was bulleted so that no bias would be introduced – an example is the way I named the functions for each of the tests. For instance, on any given constructor, you should see a function that will test it to be not null if that was a requirement and the function itself will end with “NotNull” in the name.

**Finally, evaluate the importance of being disciplined in your commitment to quality as a software engineering professional. Why is it important not to cut corners when it comes to writing or testing code? How do you plan to avoid technical debt as a practitioner in the field? Provide specific examples to illustrate your claims.**

A commitment to quality is of course a requirement to be a good software engineering professional – or any professional, really. Within the software engineering profession, we are fortunate to have testing tools to validate, check, and assess our code before it is all completed. As we learned early on in the course, and investment in testing early and often saves money on an orders of magnitude scale for a software project. Therefore, the testing must be as good as can be done and it must be thoroughly completed, to make sure the return on investment is as good as can be made. Keeping up to date with knowledge will help myself as an individual avoid technical debt - far more common though is technical debt within the system. It needs to be updated, tested and verified on a regular basis to ensure that technical debt doesn’t hinder it’s performance or security and as a practitioner within the software development field I can do my best to ensure that’s done when possible.