Project 2 Submission

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

My testing approach aligned with software requirements by first testing early and often. After each requirement was believed to be fulfilled, I added assertions to ensure that the code is working as intended. I put effort into ensuring at least the production of the system was done in a way that facilitated the best results. More specifically, after completing the task object I created assertions that tested for expected outcomes according to the software requirements. The requirements stated that the ID must not be null and must be below 10 characters, so my expected value would be an IllegalArgumentException if those parameters are not met. The assertions would tell me the results of that query.

Defend the overall quality of your JUnit tests. In other words, how do you know your JUnit tests were **effective** based on the coverage percentage?

I believe my Junit tests are thorough because I have continued to test each parameter individually. That way, I can make sure that all requirements are not only met but tested as well. Missing coverage should only happen if I missed the function entirely, such as if I were to forget to add the ability to update the name and description in TaskService. I also believe the exception handling also helped my Junit testing cover more because I did not have to individually test for what is out of bounds, but rather what was in bounds. That means less surface area for possible outcomes and more expected outcomes, to include exceptions.

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

Outside of testing early and testing often, I also implemented some exception handling and a minor amount of encapsulation. My objects throw a IllegalArgumentsException whenever the parameters are not met to prevent unexpected values. I made an effort to use private variables and data type conventions such as specifying a variable as a string (although Java pretty much forces this) which also contributes to technically sound coding. Finally, I maintained proper coding etiquette and naming practices. Without those I would not have been able to keep my code concise and proper, and surely would have made mistakes I could have missed despite having the tools to fix it.

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

To ensure my code was efficient, I made sure that all values were used. In the case that certain values needed to be used many times, I did my best to ensure that they were variable values that were accessible only when required so the code was never running too many extra steps to receive information that should be already available. All of my getters and setters reference the base case rather than redefining the requirements set by the client. Finally, I made modifications to some of the original renditions of the code I had made because I was being inefficient about using indexing to reference the arraylist with the ID rather than a loop. With the loops for each assignment, the efficiency of locating objects based upon ID became so much better.

What were the **software testing techniques** that you employed in this project? Describe their characteristics using specific details.

For most of the project, I implemented similar software techniques. My intention was to test early and test often. I added assertions to ensure that the code is working as intended after each section of code so that I know approximately where my code started having an issue with unexpected outcomes. For each milestone we were required to test null and longer than expected values to ensure the intake of values was within an expected range, and otherwise throw an exception so that as many values as possible had a reasonable value both when parameters are met and otherwise. With that in mind I referenced the rubric often to ensure that my test coverage was satisfactory regarding all the features requested.

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

Other software techniques I did not use for this project consists of different forms of vulnerability testing as well as any development or deployment methodology. Any code that would be in use that stores information, especially personal, could be exploited and/or stolen. I have the minimum amount of vulnerability protection since this is not a system that will go live. I also did not need an API, nor do I have a team of people or a long period of development time, so I did not go through the process of agile, DevOps, Waterfall, etc. to produce the software.

For each of the techniques you discussed, explain the **practical uses and implications** for different software development projects and situations.

Vulnerability testing is useful for prevention of data loss and exploitation of an otherwise defenseless system. In this case, the data being stored is not personal, so the expense of losing such data is not near as high, however it would still be costly to lose appointment data should the system be breached. I also mentioned software development methodologies, which are more practical for longer or larger projects, especially with a team of people that must communicate to create the product. Following one of these methodologies sets the pace for production and attempts to ensure that no step is forgotten, leading to a more whole product with minimal missing features in accordance with the client’s request.

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.

As a software tester, I employed caution my employing the most amount of coverage possible with my Junit tests. The more tests I run that receive expected results, the worse chance there is that something will go wrong, especially in areas I have tested for. These complex interrelationships that were being tested are incredibly important because systems that do not work as intended can be more than just an inconvenience… they can cause harm. In the specific case of the system I have developed, improper use or errors in the system could result in client data being mixed or lost. A specific example could be if the IDs of any part of the system was lost or manipulated in an unexpected way, users may have other user’s information available to them as well as not have access to their own when conducting business with the client. A business that has problems managing client data would not be considered trustworthy. My faulty system could ruin a business’s image.

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.

To limit bias, tests were run even for problems I didn’t anticipate happening. I ran assertions for even the base case for each milestone not only to increase coverage but to ensure that even areas I am confident in due to my bias were still covered objectively. Rather than testing only the areas where problems are most common to arise, or even just areas I was not confident in, objectively testing everything is a much more grounded methodology for tests since there is no personal involvement in testing each function. Bias is a realistic concern when I am responsible for my own testing because bias can get in the way of test coverage. In addition to this, confidence and expectations in my own coding could affect how I conduct my tests as well, leading to less effective and efficient testing overall.

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.

Being disciplined in my commitment to quality is important because a lack of attention to detail and cutting corners leaves room for errors to be made, and in some cases go unnoticed. Some errors are small inconveniences that are quick to fix but cut the corner large enough and the business may end up with a very costly mess where some may be hurt or killed. To avoid technical debt as a practitioner on the field, I hope to thoroughly test my systems. In addition, when creating systems, I will provide proper escapes and fail safes to prevent worst case scenarios. While these will depend on the system in question, if my system is going into the medical field for example, into x-rays, it is important that only a safe number of x-rays can possibly take place. Otherwise, radiation poisoning, injury, and possibly death may follow.