**Summary and Reflections Report**

1. **Summary**
   1. Describe your unit testing approach for each of the three features.
      1. To what extent was your approach **aligned to the software requirements**? Support your claims with specific evidence.    **My unit testing approach was to follow the requirements of the customer. Name lengths and text descriptions were kept at the required length and the unit test looked to ensure that is what was done. I generally tried to stick to exactly what was asked so that it would align with the software requirements.**
      2. 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 feel my Junit test were effective because every function that was present inside of my code had a test associated with it that would test it functionally. Ultimately the functions play a major part in the performance of the application with receiving input data. So having a test for each will lend to producing a quality application.**
   2. Describe your experience writing the JUnit tests.
      1. How did you ensure that your code was **technically sound**? Cite specific lines of code from your tests to illustrate. **I ran the debugging tool to ensure that my code was technically sound. Although this did not clear all my errors. For explain in appointmentserviceTest line 61 this is still an error I could not fix. For myself taking my error list and removing as many as I could was how I keep tried to ensure my code was technically sound.**
      2. How did you ensure that your code was **efficient**? Cite specific lines of code from your tests to illustrate. **Structuring of the code and checks was done in the most efficient way I could think of. For example, in Appointment line 20 if statement if the input equals null then the cycle will stop and throw a illegal argument. This was placed before the check for character length because why check character length if nothing has been entered.**
2. **Reflection**
   1. Testing Techniques
      1. What were the **software testing techniques** that you employed in this project? Describe their characteristics using specific details. **Unit testing was the software testing techniques most utilized during this exercise. Unit testing primarily isolates smaller sections of code like functions or methods and verify their function. These tests can often be automated and ensuring these smaller sections of code function properly this will ensure the application process inputs correctly.**
      2. What are the **other software testing techniques** that you did not use for this project? Describe their characteristics using specific details. **Another software testing technique is system testing. During system testing you evaluated the overall software system. This test checks that the overall system meets requirements and functionally. This test would usually be performed after integration testing and before acceptance testing. For this test the software must complete and able to be executed.**
      3. For each of the techniques you discussed, explain the **practical uses and implications** for different software development projects and situations. **The practical use of unit testing is it is far easier to handle faults when the code is broken down into smaller section. With unit testing the focus is on functions and methods. This is important because these functions will handle user inputs. Stopping the user for inputting a value that could crash the entire system will save on potential errors once the application goes live. Also, unit testing can be done earlier in the software development cycle correcting errors early. The later in the software development cycle an error is found the more it will cost a company to correct. System testing is done on the entire application. It will be done later in the software development cycle. System level testing detects and identifies system-level problems before the acceptance test. Correction of these errors found here will improve system reliability and scalability in the future. Also, it will create user confidence in the quality of the software.**
   2. Mindset
      1. 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 acting as a software tester, I did not employ much caution being as I was also the developer, but I understand why this is important. As tester we must appreciate the work that someone has done and not just bash it once it has failed a test. Coding takes a very high skill set to do it efficiently and as tester we must report our findings and leave out the bias attached. In manufacturing I often get involved with products failing quality test and trying to fix the problem. During these situations nothing would upset me more than when the tester judges my work due to a failed test. It lowers moral and causes tension within the team.**
      2. 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. **When creating testing for code that you created, I can see how bias will be a problem. Knowing the nature of the code you created a lot of the tests could subconsciously be made to ensure that the code will pass. Only checking section that we know are perfect and leaving the others alone. Much like selling a housing during the showing you put emphasize on showing the finished rooms, but speed pass the ones not completely put together. This could occur will self-testing code. Sections that we a hundred percent correct will get the bulk of the test were the portions that a little weaker will only get the minimum.**
      3. 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. **Throughout this course we have been presented will multiple articles showing how important it is to be disciplined in creating quality code because the consequences could be great. For my discussion this week I spoke about how deleted files during an upgrade cause a plane to crash killing the crew on broad. This is a simple mistake that caused loss of life and high financial toll on the company, but it highlights the importance of being detailed and not cutting corners during work. Where I do not plan on being a software developer or tester in the future, I will use the lessons learned to remember that my work does affect other people. Keeping this in mind with help to drive the point that each and everything I do needs to be up to standard because the outcome could not be favorable.**