Summary and Reflections Report

Throughout development I tried to ensure that my testing followed the requirements set for the project. Focusing on requirements allowed me to create specific tests for the features expected of the product. An example of these tests would be testing for variable length and ensuring that variables are not null. When checking the quality of my JUnit tests one easy factor to look at is coverage. Coverage percentage tells us what percentage of the code is being testing by the JUnit tests. Other factors that I kept track of included failures and test run time. Run time tells us if the code is running efficiently based on how long it takes to run the JUnit test. Tracking failures allows me to see how many tests failed, indicating a problem in the code.

To ensure my code was technically sound I ran tests early and often. I also reviewed my code in detail for syntax errors. I tried to keep my variables and code simple and efficient to reduce run time and prevent any issues. I have found that keeping variables simple and consistent allows for easier review or collaboration on projects. Focusing on requirements and functionality allowed me to ensure my code was efficient. Each test was created for a specific requirement and did not include any unnecessary code. I also made use of comments to make reminders or explanations for myself if I ran into any issues.

Throughout the development of these milestones mostly black box testing was used. Black box testing was used to test if functionality met the development requirements or not. It also allowed testing of performance errors. The tests used in each milestone focused on factors such as variable length, description length, and variables being empty. The program was also required to be able to update and create new contact, tasks, or appointments as needed. Tests were run early and often during development to prevent and issues later. Throughout development I have found early testing to be largely beneficial to time management and productivity.

During development of these milestones, I did not use interface testing as the program being created does not yet have a user interface. I also did not make use of cause-effect graphing, boundary value testing, or random (Ad Hoc) testing. As the development was focused on implementing specific requirements early testing focused on functionality and variable length seemed the most important. I have found that using an Agile method of testing is beneficial as it promotes continuous testing as development proceeds.

While working on this project I had to adopt a more detail-oriented mindset. I was more cautious when creating my code as I was focused on meeting the requirements of the project and ensuring that my tests worked properly. Learning to implement JUnit tests allowed me to realize how complex code can become when it all works together. When reviewing my code, I split it into different sections and checked for syntax errors and other issues one section at a time. Not reviewing the code in the order I created it helped me see more errors and limit my bias. It is important to not cut corners and test properly because doing so can result in expensive and difficult problems to fix later and can be brand damaging.