Christina Roki

When it came to unit testing for each of the three features, they all had a pretty similar core approach. Of course, as each one was submitted in a different module, this allowed me to change my testing approaches depending on the grade I received. For example, when I wrote out the tests for Contact Service, I kept referring back to the code following the tests to a similar style. This sufficed for Contact Service as I received a decent grade, so I decided to apply the same approach to Task Service, but received a poor grade because that was not the testing approach needed for this feature. I believe that my JUnit tests were effective for the basis of the code. I believe that I could have done a better job on coverage like ensuring edge cases were taken care of. I would create a list of scenarios/possibilities that I would type out as tests and constantly review the code with the tests to see if they align.

I believe they highlighted many of the components targeted; however, in terms of coverage, I do not believe they covered a majority of cases that it intended to do. Something I would do to ensure my code was technically sound was by going back to the requirements and checking off every component I believe I completed. For example, one of the requirements was “The task service shall be able to add tasks with a unique ID.” So, the line of code I added was Contact c1 = new Contact("C1", "Sara", "Chinna", "9976578987", "Ashofrd Gables, USA"); This supports that each is linked with an unique ID that allows tasks to be tied with this. One way I would ensure that my code was efficient was by making sure there was no redundancy code. For example, I created a function called toString() which would list all the contact’s information instead of doing this to each contact by a written out string.

One of the main software testing techniques I picked up during milestones was to look through the main code/functions section by section to ensure that I am highlighting all test points. When I did not do this, I would forget certain test cases that for example, may throw null. Usually, these are edge cases like a function that takes a number and looks through an array list to output a string may not account for negative numbers. And of course, with testing this will remind you to write that in the program. Of course, this technique may apply more heavily on certain projects than others than involving a lot of edge cases with array lists and such.

Some software testing techniques I did not pick up during milestones are efficient commenting. This may sound simple but it is very effective. For example, I have a tendency to comment a lot during the development of the program. But, when it comes to testing the files, I always seem to forget to jot comments and then when I go back to look over the tests, I forget what I was testing and the reason behind it. I may be able to get away from commenting depending on the situation. For example, if I was the only one looking at this code and may be confused on what I was testing, I can end up figuring it out even if I;m not being really efficient on time. However, if I was working with a team on this project, I would never get away without comments since that is also what other developers rely on to understand the other developer’s POV.

In all honesty, the extent in which I employed caution taking on this project was very little. I say this because of course it is not as heavily pressured as a job on the line or money. However, it was important to appreciate the complexity of the code I was testing or else I would not be understanding it. For example, the Appointment Service project is a real life scenario that could even lead to the Y2K bug. I could 100% see bias being a concern if I were testing my own code subconsciously thinking I have covered all requirements needed. Of course, it is very important not to cut corners when being a software developer as this leads to many more issues down the line making the whole process extremely inefficient. Also, discipline is very important especially for time management, balancing work and life and progressing the project each day.