CS320 Project Two

Creating the Contact and Contact Service classes for this project required substantial testing, as did creating the Task and the Task Service classes in this module. My testing aligned with the software requirements because I made sure to test the elements of this project that were explicitly laid out as requirements for each class. For example in the Contact class there were requirements that stated that the first name of the contact could not exceed ten characters. So in order to test this we created a test that would input a value in the first name field that had over ten characters. When running the JUnit tests for this class it failed as expected and passed the tests. Likewise in the contact service class we tested that the first and last name, as well as the Number and Address fields were updateable as to match the requirements stated in the rubric.

The quality of my JUnits for the contact service and task service classes are sound. I believe they cover a high percentage of the packages and ensure that all the requirements are met within the scope of the classes. I know this because each portion of the tests run touch on a corresponding block of code within the Contact Service and Task Service classes, nothing is left untouched by the tests that I ran.

I ensured my code was technically sound by ensuring it was free from errors and double checking that the syntax was accurate when assigning values within my code. For example in my “Setters” block of my Task Class I ensured that “TaskName” followed the correct formatting based on my private String that was set to TaskName at the beginning of this class. Similarly I made notes throughout my code to be sure that this was following the requirements of this project. For example the task name could not exceed twenty characters so I made a note in my setters section to note that the TaskName could not be longer than twenty characters.

I made sure my code was efficient by utilizing nested if else statements in my constructors and Setters blocks. For instance when I created my public “Task” I use if else statements to set the parameters of the task name and description so that it would meet the requirements of the project without excess code being written.

Each milestone that I have completed utilized Black Box testing primarily. This is because testing has been entirely based on the requirements of the program without knowledge of the internal program structure. Getting even more specific into the types of testing methods I used, I focused heavily on using systematic testing and sanity testing. Sanity testing came in to play when making sure that the basic functionality of the application worked as expected. Systematic testing was used to ensure that each and every requirement was met based on specific parameters that were laid out in the rubric’s for each assignments.

Another type of testing that I did not utilize throughout any of the milestones was Non-functional testing. Non-functional testing is more geared towards the performance of the application without focusing on the functionality. I did not focus on this type of testing because at this point in time the assignment is not about getting the best performance possible but instead, meeting the application requirements as needed. Non-functional testing is still important especially when the application that will be used will require a lot of heavy lifting. If the application does not have high performance then this could lead to a slow product or even a product that crashes regularly.

Systematic testing and Non-functional testing are two very key techniques in the software QA testing world. Systematic is key because this requires you to test each functionality of the application logically and functionally to ensure that you have met every requirement of the customer so that you are absolutely sure that you have turned in a finished product that the customer will be pleased with. Non-functional testing is also vital and has different implications for the software development lifecycle. This allows you to be certain that the application is performing well and not at risk of crashing due to performance based errors.

The mindset that I adopted while working on this project and acting as a tester was to focus on primarily meeting the basic requirements of the project itself first and foremost. Then second was to focus on generating tests to ensure the quality of the code was up to the standard that it needed to be in order to be successful. I employed caution pretty heavily throughout the project to make sure that any changes I made did not regress the project in any way. I also was very cautious to be sure I tested every element of the code so that I knew it would work as expected when used by the client. It is very important to appreciate the complexity and interrelationships of the code I was testing because the more you appreciate it and understand how it flows together, the better equipped you are to make changes or additions in the future to that same project if needed.

I tried to limit my bias in code review by doing in depth J-Unit testing. This was a pretty clear cut way of determining if the code I had written was accurate and working as expected. However I will admit without J-Unit testing it would be much harder to harshly critique your own code if you were running tests manually. Discipline is absolutely key in the development process. We cannot cut corners because when we do bugs are introduced to the code, poor quality of work is produced or deadlines are missed and all of that will result in bad feedback for yourself as a professional.