Project 2 George Kaline III

My approach to testing the software requirements came down to following what was required in the initial assignment. In all three features the requirements for testing were laid out. All three features all followed similar testing methods. With the task test dot Java file, I created I simply created a new class called task test and then created a new task with a get ID, get name, get description entry for each test. I used a valid ID number I used my daughter's name and a simple description set up. I wanted to keep it simple and understanding to not get confused with a complicated test. The next test tested the class length and performed the test to bounce the entries against their requirements laid out in the project. For the contact test I set forth and followed a similar process I created a contact test class then followed up with entries for a get ID, get first,name, get last name, get phone and get address. All the examples used were again simple, very easy to follow and understand. This was to eliminate any erratic testing procedures to not complicate the solution. Next, I tested the test contact name length to validate that all of them followed the requirements set forth in the project. The last one was the appointment test. This test followed a very similar testing method and was set up to create a new appointment. I used the get ID get date and get description in order to test all the required content. I used simple numbers and a description and followed up with the date library class from the import Java dot utility dot date. This allowed for the date to be formatted in a certain way that the app user wanted to see. Next, I created the test appointment named length. This pulled in the testing entries that are used to test against the requirements set forth in the document.

The J unit test created was effective and successful. The J unit test created was effective and successful. If the J unit test came back failed that means my entries were not valid. Retesting with valid entries proved that the coding was correct.

I shared the code that I wrote was technically sound by running multiple tests using correct and incorrect entries for each of the testing procedures. This allowed me to test my own code and test the testing code to validate that I wasn't fact getting valid responses. By keeping the testing method simple by giving them very generic names. It allowed me to simplify a code. For example, in the appointment test dot Java file naming the package appointment and then keeping the appointment theme of appointment test for the class as well as calling the appointment dot get ID, appointment dot get date, appointment dot get description all allowed me to follow a simple method of testing and naming of the class items. I followed this and all of the testing for the appointment contact and task Java files. It's also allowed for easy copying of my code to create feature J unit test files. Once placed and copied I was able to use the find and replace method. This made the overall process of creating the J unit tests much more time efficient. It allowed me to focus on the coding aspect instead of worrying about typing mistakes.

I ensured my code was efficient to minimize the amount of code needed to be written. This helps keep everything organized and simplistic. Over complicated code can cause issues down the road then followed up by J unit testing can complicate the matters if the code is not well written. In the contact test Java file I use the assert true function the contact.getID().equals(“”) method. This allowed me to have a copy and paste method for each of the entry lines required to test. This would allow for simple straight to the point testing and allow for someone to open the file and understand the code I have written.

For project one I used functional software testing techniques. At a high level this is a very straightforward yes or no answer to a test. This does not leave any room for error. Like a true false statement this allows for the answer to be true or false. And this and since we wanted to make sure that everything came out true and that the testing used and with the entries were what the client wanted in the mobile app. I also used a manual testing method. This manual test required me to type in the entries into the program of the test contact class and other two very similar. I could have written a UI to allow for a user to enter the values themselves instead of having to open up the code or Java files directly.

I did not utilize any sort of security testing or performance testing techniques while writing the code. While important for all applications to have security implemented. This was not a direct requirement of this project. I assume this part of the project would be handled at a different step along the way. Functional testing is a testing technique used to test the requirements are met of a certain software package. This is important for any project to follow the requirements written by a client. At its core our main objective is to provide a working solution that is tailored to a client. Nonfunctional testing while important handles mostly performance security and usability of an app or program. Nonfunctional testing is more about the feeling and responsiveness of an app. If your app is slow and uses contrasting colors the user experience may not be as desired by your client. That is why it's important to have proper functional and nonfunctional testing in place when creating an app nonfunctional testing may look different and may not require J unit testing. It may require more manual testing by a user or a review board 4 proper color palette selections.

The use of caution in this testing project was rather simple. The caution used in this testing project was validating the entries used within all three testing dot Java files was to validate that my entries passed all the tests. The caution used to validate these tests was further testing to input false data that I knew would fail. This allowed me to remove any doubt about the entries being correct. By throwing false data in it allowed me to make sure that even false data would not come back true due to a coding error or mistake. It was very important to understand the relationships between the class files and the testing files to make sure that they were accessing the correct class information. If typed incorrectly the testing file would not register from the clash file. For example, making sure that the get ID, get first, get last name, get phone, and get address information from the class contact was referenced by the contact test class during the testing phase.

During the testing phase I made sure to follow the requirements listed within the project. This allowed me to be unbiased towards my own testing because I was not coding for the responses that I put in rather the requirements listed in the project. Understanding the testing of your own code can be tricky when you know the answers, so you are trying to write the test to prove your answers. Having a third party write your testing procedure based off a requirement and then running the test can always be beneficial and should be best practice for any environment.

Being disciplined when testing code will allow you and your team to put out a better product. Having a predetermined standard testing procedure written for any project can be helpful. These will act as guidelines in testing. These are usually written by senior programmers who know what to look for. These may not be specific to a certain project and will have to be modified but these guidelines can help in testing to ensure the testing is similar for the entire company that employs you. Cutting corners and writing and testing code can lead to disastrous consequences. Releasing code for an app or a program that can be consumed by millions of people could lead to issues poor reviews on app stores which could lead to quality issues in the future. In the future I do plan on running J unit tests on all Java code. This will help ensure that the code written will be back checked against the requirements put forth.