Jeffrey Thornton

CS320 Project Two

10/18/2024

**Summary:**

For the development of the mobile application, I utilized unit tests for the contact, task, and appointment services and to ensure my tests were effective, I employed a combination of testing techniques which involved entering data to ensure the correct output was produced, and examining the code to check that the logic is sound. I wanted to also ensure my code was efficient and used a combination of methods, such as refactoring and removing redundant code. This approach, aligned with the software requirements outlined in the rubric and ensured my code met all the required functionalities and ensured that my tests were effective.

In addition to my unit test, I also used JUnit test cases to examine the code for errors. My experience writing the JUnit tests was overall positive and it allowed me to ensure that my code was technically sound by verifying that the logic was correct and that the expected outputs were produced.

Practical uses of the software testing I utilized is that they can be used to verify the correctness and efficiency of the code. They can also be used to uncover errors in the code or find and identify edge cases and scenarios. The dangers of not using software testing are that potential errors and security risks may go unnoticed making their way to end users or corrupting code further down the line.

Some software testing techniques that I did not use for this project included performance testing, integration testing, and user-acceptance. Performance testing is used to measure the speed, scalability, and responsiveness of a system which is impossible to do without access to the customer’s network. Similar to performance testing, integration testing also requires access to the customers networks and host environments. User-Acceptance testing is a vital step but must be completed by users actually executing the job which we do not have access to.

**Reflection:**

Different software testing techniques utilized:

* **Unit testing: This** is a testing technique that involves testing individual units of code. This is the most basic level of testing and is essential for ensuring that the code is correct and functioning as expected.
* **Boundary value analysis: This** is a testing technique that involves testing the boundaries of input and output values. This helps to ensure that the code handles extreme values correctly.

Software testing techniques that I did not use:

* **Integration testing:** This testing technique involves testing the interactions between different units of code or code between multiple platforms. This is typically done after unit testing has been completed.
* **Performance testing**: This is a testing technique that involves testing the entire system. This is typically done after integration testing has been completed and stresses the environment to ensure it meets customers average demands.
* **Acceptance testing**: This testing technique involves testing the system via users or customers. This is typically done after system testing has been completed.

Each of the testing techniques listed above have practical uses and implications for different software development projects and situations. However, unit testing is always a good choice and should be performed regardless of project size and/or complexity.

**Mindset:**

My mindset while working on this project, was to be very cautious and aware of the complexity of the code while taking my time and being cognizant of the requirements outlined in the rubric. I used code reviews and unit tests to thoroughly examine the functionality of the code to ensure I had not overlooked anything. I was maintained awareness of the complexity of the code I was testing, and I took care to test all possible combinations of inputs and outputs outlined in the rubric. Developers may be more likely to overlook errors in their own code, so it is important to have a critical eye and to be willing to ask for help from other developers if testing uncovers errors or mistakes.

Lastly, I focused on quality when writing and testing my code. I avoided cutting corners and used code reviews and unit tests to thoroughly examine the code and ensure that it was free of errors. I invested the required time, patience, and effort into improving the quality of my work while maintaining a commitment to quality. This ensured that my code was reliable and meets the needs of customers.