7-2 Project Two

CS 320

Camrin Stilwell

8/13/2023

**Summary**

Unit testing is an important aspect of developing software. This reduces the chances of problems and errors affecting the software's overall functionality. It can also improve the code quality based on any issues that are found, which can be dealt with. Regarding the software requirements, my unit testing approach for the three features had similarities. An example was from the Contact service where the contact objects had unique requirements, including the Contact ID, which cannot be updated, first name, and last name, which must be no longer than 10 characters and cannot be null. In addition, there was also the phone number that must be 10 digits long and cannot be null and an address that cannot be longer than 30 characters and not be null.

The Contact service also had features that allowed the contact object to be added based on a unique ID, deleted per ID, and could be updated per ID with the following being updatable: first name, last name, phone number, and address. These requirements reflect the features that the contact service must use to meet quality and functionality expectations. To address these features and their entirety, unit testing was needed. My unit testing went over the requirements of the Contact service by testing each requirement with their test cases based on what is inputted in the contact object. Overall, my junit test went well, as I did have issues in the beginning with how my environment was set up, which made it a challenge to get testing to work. However, after doing similar programs with testing, I learned from my mistakes and fixed what was necessary.

A screenshot of a computer

Description automatically generated

The lines of code above create an object allowing contact methods to access it. I ensured that my code was technically sound by making sure it was used to test contact.java as it calls for numerous methods. This was a great way to test if the code was functioning as intended.

A computer code with black text

Description automatically generated

The lines of code above test the functionality of adding a new contact with contactservice.java. This displays that the code was efficient at testing a service requirement.

**Reflection**

A software technique I used for this project was Dynamic Testing, as it involves testing the dynamic behaviors of the code. An example of Dynamic Testing is giving an input and getting an output for each test case based on the service requirements of the contact service. This technique also allowed me to check inconsistencies during the software's runtime. A software technique I did not use for this project was Static Testing. This testing involves testing a piece of code without executing it. When comparing dynamic and static testing, overall, static testing prevents defects, while dynamic testing finds and fixes defects with test cases.

The mindset I adopted when working on this project was ensuring I included the necessary test cases based on the requirements and how the services would function. It was important to understand how the code reflected these requirements and what the methods do within the class. With this understanding, it would be easier to relate test cases to code that may have issues or know what the code does.

Bias can play a factor in reviewing code. However, it is important to accept failure, no matter how confident in how the code is written until the proper testing is conducted. When creating variables for a class, one might feel confident that the code written will be practical and functional. This mindset should be used cautiously as the focus should be on testing the software, which will reflect how well the code will work in the end.

Being disciplined plays a significant factor as a software engineering professional. When creating software programs, the initial coding process must contain few to few errors or bugs. When writing and testing code professionally, time and money are also considered. To avoid losses, testing code ensures that the program works as intended before meeting deadlines. An example of this is the Contact service was ensured with each unique contact ID that cannot be updated so that each contact is unique and does not cause any errors with other contacts.