**Project 2: Reflection**

* **Summary**
  + **To what extent was your approach aligned to the software requirements?**

All three features of the mobile application met the software requirements. The three features of the application included, contact service, task service, and appointment service. The contact service required me to support storing contacts. This involved adding, deleting, and updating contacts based on their contact ID. The task service allows the system to support storing tasks. This required the system to allow adding, deleting, and updating the tasks based to a task ID. Finally, the appointment service allows a customer to make an appointment. The system needed to be able to add and delete appointments with a unique appointment ID. For each feature in this application, I had to ensure that if any inputted information was NULL or exceeded the allowed character length an error would be caught. After creating each of these classes for the application I tested them using JUnit tests.

* + **Defend the overall quality of your JUnit tests.**

I ran the JUnit test on each class of my project and saw that there were no errors or failures detected. Also, the coverage on each class ranged from 81.5% to 100%. The classes that had a coverage in the 80s were my classes that contained my getters and setters. Since the setters are not tested, they lowered my percentage. Seeing that every testable case in my project was covered ensured that my JUnit tests were effective.

* + **How did you ensure that your code was technically sound?**

I ensured that my code through the project was technically sound. I reviewed the requirements of the programs many times to ensure that I knew what to test for. After, I added every test case that could cause my program to fail. As an example, on my task service test class, I added five ways the update task service could fail. In my task service class, I printed hard coded messages for every way my code could fail. When I ran the test, I saw all the printed messages were correct. I also ensured that my code was technically sound by creating an array list in multiple classes in my application. This allowed me to add items, iterate through the list, delete from the list, and update items within the lists. As an example of this the following code is from my task service class.

Text, letter

Description automatically generated

* + **How did you ensure that your code was efficient?**

When writing my code, I ensured that my code was as efficient as I could make it. An example of this is in each class I created I defined my variables inside my loops. An example from my appointment service class follows.

Text

Description automatically generated

* **Reflection**
  + **What were the software testing techniques that you employed in this project?**

While working on this project, I completed my functional testing using unit testing and some integration testing. I performed unit testing by ensuring each individual component of my program worked appropriately. Unit testing is the first level of testing done before the different units are integrated. Within each milestone I would test add functions, delete functions, update functions, and more. The integration testing was done by inserting add cases into my delete test cases. I would see if they worked well together. If something was added, such as an appointment, I would use the delete test on that added appointment.

* + **What are the other software testing techniques that you did not use for this project?**

There are many different software testing techniques that I did not use. These testing methodologies build on each other. This app I am working on is in the early stages of development. Therefore, I used the first two testing techniques. For function testing the next steps would by system testing and acceptance testing. System testing is a black box testing method used to evaluate the completed and integrated system. The project would have to meet all technical, business, and functional requirements. Acceptance testing is the last phase of functional testing. This testing ensures that the project meets the end user’s needs. This testing will include beta and alpha testing. This will put the product in your user’s hands to look for any errors on a massive scale.

* + **For each of the techniques you discussed, explain the practical uses and implications for different software development projects and situations.**

As stated previously, each of these software testing techniques build on each other. Unit testing should be done in every project. Unit testing will make debugging easier because errors are found very early in the project. Ensuring all individual components work before integrating them is very important. Integration testing is also important in every project. These are the tests that will be used with user stories. Such as, logging in or opening files. The individual components must be able to work together seamlessly without any errors. System testing is very important as well. The project is tested from beginning to end in many different test cases. The final test phase is acceptance testing. This testing is very important to see if the users will accept the final product or find errors. This testing phase should also be used for every project. Performing beta testing will allow crucial feedback from potential customers.

* + **Assess the mindset that you adopted working on this project using caution.**

I ensured I was very cautious when developing this project. As I was adding my different test cases, I would run my tests. If there were failures, I would look at my code and adjust it until the tests succeeded. After one section of my code passed, I integrated it into the next section and would test it. Testing your code every time you change it or add to it, will result in less errors to debug. Attach is an example of different parts of my code integrated. As you can see, I am testing my add and remove of a task service working together.

Text

Description automatically generated

* + **Assess the ways you tried to limit bias in your review of the code.**

There are many ways to have bias when developing and reviewing your code. Every developer wants their program to work properly. However, it is important to test for all possible failures. This is called debiasing the confirmation bias. It is important to avoid testing what you know will work in a program. During this project I insured I tested every possible outcome to make sure my program worked properly. An example of this is below:

Text, letter

Description automatically generated

* + **Evaluate the importance of being disciplined in your commitment to quality as a software engineering professional.**

It is very important to be disciplined when writing code. Cutting corners when writing or testing code can result in many errors. Software is used in many modern-day devices. This may include, aircraft, missile navigation systems, satellites, vehicles, and many other devices. If the programming within these devices contains an error the device could fail resulting in injury or death. I plan to avoid technical debt by making sure I test every piece of software throughout a project. As I finish a small part of code, I will test it. After testing individual pieces, I will integrate them and test them. I will use automated testing programs to expediate the process.

**CITATIONS**

Changyi. (2020, April 14). *50 efficient code samples for Java Programming*. Alibaba Cloud Community. Retrieved December 11, 2021, from <https://www.alibabacloud.com/blog/50-efficient-code-samples-for-java-programming_596122>.

*Technical debt*. ProductPlan. (2021, September 2). Retrieved December 11, 2021, from https://www.productplan.com/glossary/technical-debt/.

*8 cognitive biases in software development*. The Valuable Dev. (2020, May 27). Retrieved December 11, 2021, from https://thevaluable.dev/cognitive-bias-software-development/.