**CS 320 Project Two**

1. **Summary**

In my role as a software tester at Grand Strand System, I've employed diverse strategies to ensure the quality of our mobile application. This involved rigorous JUnit testing across all software components, including contact, task, and assortment classes. Each feature was developed in accordance with specified requirements. For instance, the contact feature required a 10-digit phone number and the ability to update contact fields for each unique ID, such as first name, last name, phone number, and address. My code demonstrates the application's adherence to these requirements.A screen shot of a computer code

Description automatically generated

In the task feature, another requirement is that the task service should be able to add tasks as needed to the Task map list. This allows the program to add tasks continuously provided the required input. This can be demonstrated through the following code in the program:

A computer screen with text

Description automatically generated

In the appointment service feature, one requirement is that the appointment service should be able to delete appointments as needed using a unique appointment ID. This allows the end user to delete appointments that are no longer required. This specific feature is also demonstrated in the following code as meeting the necessary requirement:

A screen shot of a computer code

Description automatically generated

In my Junit testing for each application feature, I focused on achieving at least 80% test coverage to ensure thorough testing and sufficient quality. Throughout the testing I employed diverse methods and scopes and illustrated good testing techniques by testing for various inputs and exceptions. During each stage, tests were conducted that ensured a quality review throughout the entirety of the project. Lastly, I tested boundary cases to prevent potential errors or exceptions, while also enhancing the quality of my tests.

Ensuring the technical integrity of my code meant adhering to professional industry standards. For instance, in the Appointment Service feature, when adding appoints, I test to see if there are existing entries using if-else statements. Inside the statements, I check if the appointment ID already exists isn’t in the map list and if so, then I throw an exception. Otherwise, I use ‘put’ to add the new appointment to the map. This exemplifies adherence to coding best practices and technically sound principes.

A screen shot of a computer program

Description automatically generated

I also ensured my code was efficient by changing the initial structure of my code by incorporating the use of a hash table as opposed to an Array List. While an Array List can certainly be useful, the use of a hash table decreases the runtime complexity when adding more elements to the list. This helped in making sure my code prioritized efficiency in place of data structures that may not necessarily be suitable for large quantities of data.



1. **Reflection**

Regarding testing techniques for the application, some of the techniques that were utilized included white-box testing. In which, test cases were created that tested the functionality and structure of the program. For instance, when outlining the requirements for each component of the program, unit testing was applied under white box strategies that would test for the integrity of these requirements. For example, in our Task Service Test class, a test is created that checks that the Task that is created is stored correctly. This aligns both with our initial requirements and tests for the structure of the program. In this case, the test passed, ensuring validity in our test scenario and creation of Tasks.

A screen shot of a computer screen

Description automatically generated

Another testing technique applied is unit testing, in which the individual components of the application are tested to help validate the combined units of the program and ensure they adhere to the requirements. In each of the test classes, unit testing was applied that ensured enough coverage was present, while also testing for each method within the main classes. A prime example is that of the Contact Service Test class, in which I tested the method of deleting a Contact. While this is only a single example, its application to each method from other classes is what provides the quality of unit testing for a program.

A black background with text

Description automatically generated with medium confidence

Some of the software techniques that were not used included stress testing, where the program was not put to the limits of its efficiency. Neither was it run in a real environment where the applications speed could be tested either.

In practical software development, unit testing is a key technique used by developers to assess individual parts of a program while other components are in use. This method allows for the isolation of code sections to ensure their integrity and early detection of bugs or vulnerabilities. Whie box testing, for example, aids developers in evaluating code performance and quality, helping toe stablish sufficient coverage for the application. For example, in testing a loop of size ‘n’, boundary conditions can be applied to assess code integrity in different scenarios (GfG, 2024).

1. **Mindset**

When evaluating my mindset as a software tester, I did my best in exercising caution by testing pieces of code that would both test the specific method and validate the requirements. For example, in the AppointmentTest class, I created a testInvalidAppointmentId method. The reason I decided to do this one was because an Appointment Id was being used in several other methods throughout the program. Caution was also used by implementing try-catch blocks that would catch expected exceptions for the test.

A screen shot of a computer code

Description automatically generated

Limiting bias is certainly difficult, especially when testing your own code. However, in

this example, I tried limiting bias by testing for specific edge cases that might have been overlooked otherwise. By exploring these edge cases, my aim was not only to reduce bias but also to enhance the overall test coverage, ensuring that the code’s robustness was thoroughly evaluated. I used the testAddDuplicateTask method as an edge case because it was not a specific requirement, but it did allow me to test a method that may have resulted in an error had a duplicate task been entered or overwritten unexpectedly.

A computer screen shot of text

Description automatically generated

Cutting corners in software development or testing can lead to technical debt, where fixing issues becomes more costly than initial development. While shortcuts may seem timesaving, they often result in more errors and unexpected consequences. Maintaining discipline is crucial for software professionals, enhancing long-term success. For example, within my ContactService class, I've introduced an updateContactFirstName method. This function utilizes the contactId parameter to specifically update the first name of a contact. By allowing users to focus solely on updating their first name rather than inputting all contact information, the process is streamlined, reducing errors, and ensuring system clarity.

A computer screen shot of a code

Description automatically generated

GfG. (2024, January 30). *Code coverage testing in software testing*. GeeksforGeeks. https://www.geeksforgeeks.org/code-coverage-testing-in-software-testing/?ref=lbp