Bryan Chan

CS 320

Prof. Norman

23 June 2024

Project Two

Summary

For all three features of the project, I have developed my unit testing in a manner to ensure the highest test coverage possible. This was needed in order to satisfy one of the assignment's requirements, which was to have a testing coverage above 80%. Each of the services had unit cases which would ensure that every avenue of the service would be tested. For instance, in the AppointmentTest.java file, the Appointment.java file was tested to look for correct values such as a non-null string input with the right length. In addition to this, the file was tested to look for incorrect values such as null values and inputs of the wrong length. The same design for unit cases is used throughout the project, for the other services such as Contact and Task. By going with this design, the Appointment and AppointmentService unit cases were given a 100% code coverage, 91% code coverage for the Contact and 100% code coverage for the ContactService, and 100% code coverage for both the Task and TaskService classes. Because of the high code coverage of all three services, the program was shown to have an effective unit cases.

In order for a program to have the highest quality of code, special attention needs to used during development. This was present in the program throughout the development. For example, in every

service class, a Java hash map as storage. The reason why a hash map was used as opposed to a simple array, is due to a hash map's features and efficiency. This is seen in the fact that with a hash map, the service's ID can be used as the key, which makes it easy to search for the key. In addition to this, a Java HashMap has special methods such as the containsKey() method, which allowed the program to search for potentially duplicate objects. Not only does the containsKey() method allow for easy lookup, but it also increases the efficiency of the program by eliminating the need to use for-loops for searching through the hash map. Exceptions, which were created and called using the MyExceptions class, are also implemented in the service classes in order for to aid in input validation. These exceptions are later monitored in the unit cases for the services, in which the validity of the input is determined by whether or not an exception is thrown.

White box, security

Reflection

Throughout the development of the program, there were some testing strategies that could have been employed. For example, the following testing strategies, which are found in the GeeksForGeeks article, "Software Testing Strategies" by user madhurihammad (2023), include white box testing and security testing (madhurihammad, 2023). The GeeksForGeeks article (2023) defines white box testing as testing for the code's internal structure and logic, while security testing is when the software is tested for vulnerabilities and that it meets security requirements (madhurihammad, 2023). The white box testing strategy contributed greatly to the overall project, since it deals with the code's structure and logic, in which quality and bug-free software is created as a result. In addition to this, by implementing security testing as opposed to only testing for logic and performance, the software will be much safer since it will be tested for vulnerabilities. However, I did not utilize security testing. Security testing

would have been much of a concern if the program contained external dependencies and any other APIs, which could have compromised the integrity of the program. The dependencies would have to be tested by a dependency checker, in order to ensure that safety of the installed dependencies.

As the software testing of this project, it was important for caution to be demonstrated throughout the testing phase of the project, in order to ensure that the code would be accurately tested. To do this, I utilized custom exceptions via the MyExceptions class, which allowed for the project to test for invalid inputs. For example, in the testAppointmentDateNull() method for the AppointmentTest class, an assertThrows() methods is used to determine whether an exception is thrown due to a null appointment date input. This allowed for the high code coverage in the program.

In addition to being cautious, it was also essential for the tester to eliminate as much bias as possible when it came to the testing phase of the project. Bias can diminish the overall quality of the code since it will leave the tester with the impression that the code has been thoroughly tested, when it reality is has not. To eliminate this, I sought to utilize a non-biased standard such as the values for the code coverage. The reason so, is that these values will determine whether or not the program has been efficiently tested. Because the code coverage resulted in high values, it became clear that the program had been thoroughly tested.

Because testing is often overlooked in software development, it is easy for the tester to cut corners when it comes to it. However, a disciplined testing approach is vital for the quality of the program. This was demonstrated in the AppointmentTest class, in which unit cases were used in a way to ensure that every avenue of the input was tested to ensure validity. The end result of this approach

was a high code coverage, which demonstrated that the program had been successfully tested for efficiency.

References

Madhurihammad. (2023, February 6). *Software testing strategies*. GeeksforGeeks. https://www.geeksforgeeks.org/software-testing-strategies/