Project Two

Ray Cooke

Southern New Hampshire University

Testing

The testing approaches that were used in this project aligned to their software requirements by giving them input that should either fail or pass depending on their specifications. The results of the Junit tests proved their effectiveness. Quality testing guaranteed that the programs match the specifications of their software requirements. The Junit tests all failed in places where they would be expected to fail. For example, in the “Appointment” class we define that the date of an appointment must not be before the current day.



The test case “testSetters” attempts to insert an invalid date. Once the test is complete this is the result.

Text

Description automatically generated

While there’s a failure, it’s in an expected place and suggests that the code is working as intended. This also ensures that the code was technically sound and efficient. We have proof that the software requirements are satisfied using modular code. With proper testing and code review, we can be confident that this code will mesh well as part of a larger object-oriented programming project.

Manual code design, development, and review was used to create the classes, and Junit tests were used to verify that the classes are set up to properly address their software requirements. White box testing methodologies were used to test these projects. J unit tests were performed with the source code in mind to ensure that the rules laid out in the classes are enforced while the code is being executed. Since the testing was done in an early phase of the software development lifecycle, back box testing was not used. These testing techniques involve testing without thinking about the source code. Since I was responsible for the source code, I decided that Junit testing paired with manual code review would be sufficient because it is much faster and less labor intensive than inputting each test case. In most cases it seems as if white box testing and code review would create sufficient testing coverage. For a very large and/or security minded project, black box testing methods should not be overlooked to ensure proper inheritance structure and security protocols.

I adopted the mindset of a software engineer for this project. I ensured that software requirements are met. I thought of every test case as a fast way to pass input to the variables to double check that they are working. It is important to appreciate the appreciate the complexity and interrelationships of the code because the projects were all object-oriented. For example, the “AppointmentServiceTest” file runs tests on both “Appointment” and “AppointmentService” classes.

Text

Description automatically generated

Thinking about this through the lens of a software engineer assisted me in designing and implementing the test cases.

To limit bias, I kept things as simple as possible. With the help of a checklist, I ensured that code performed the tasks that were required by the customer. This was verified by astute review of the pass or fail unit testing method that was used. An example of this is that I made sure that every software requirement was tested and verified before submission.

A commitment to quality is essential to create great software. Discipline to do the job correctly will prevent bugs and result in less time spent on a single project or task. When it comes to writing or testing code, simple mistakes can amount to tremendous financial consequences. In some cases, loss of life or property can also stem from software bugs. These reasons emphasize the importance of thorough software testing and the need for discipline in all phases of the software development lifecycle.

Using best practices and naming conventions are among the simplest ways to remain disciplined in software engineering. Proper annotation is another simple method to help peers understand the intended purpose of code, for this project I used in line comments. Keeping syntax as simple as possible also upholds the integrity of a program that has multiple people involved. Of all the ways that technical debt can be avoided, I believe that software testing is one of the most important. Understanding software testing from the perspective of a QA and software engineer was very helpful in understanding this.

References

*Software testing : An istqb-bcs certified tester foundation guide - 4th edition*. (2019). BCS Learning & Development Limited.