Fabian Rodriguez

10-13-2022

CS-320 Software Test Automation & QA

Project Two

The approach was aligned to the software requirements to a great extent. The software did not have a lot of features which made the process of ensuring compliance easier. It is hard to measure the quality of JUnit tests, but there are some ways to tell if they were effective. One way is to see if the developer who wrote them understands the requirements and knows how to implement them. Another way is by seeing if the unit tests are well-written and give a clear idea of what they are testing. I can tell that my JUnit tests were effective based on the coverage percentage. The coverage percentage is a measure of the amount of code in your project that has been executed by your JUnit tests. To have an effective and robust test suite, it is important to have high coverage percentages. If you don't have high coverage percentages, you are not testing all the necessary functionality of your code. The code coverage for the project is currently sitting at 100% for methods and classes, and 92 percent for lines of code tested.

Writing tests with JUnit was quite simple. Like most testing frameworks/libraries assertion was a strong element that help the developer create concise test that can help find bugs or enforce requirements.

When developing code, the developer should always keep in mind that they need to write it in a way that is technically sound. This means that they should not make mistakes when programming and account for all possible scenarios. The best way to ensure that your code is technically sound is by writing unit tests. Unit testing is a type of software testing that checks the functionality of a small part of the software. Unit tests are written by programmers and executed by the computer at any time during the development process, even before the code is committed to version control. To mention some specific part of our code we have ensured that the id was unique by following two practices. We used UUID, we reduced the size to 10 to meet the requirements and then we wrote a test to ensure that the ID was indeed 10 characters long. To be more efficient the ID is automatically created every time a new object is created.

The main software testing technique employed in this project was unit testing. Unit test is a type of software testing that is used to test the smallest units of code. The purpose of unit testing is to make sure each individual unit of code performs as expected. Unit tests are written before the code is written, which allows for bugs and errors to be found before they happen.

There are multiple other testing techniques that could be used for this project, however, due to the scope of the project were not necessary. Integration testing is a technique that is used to test end to end software where two or more software units are involved, this technique was not used for this project since it does not have a separate user interface. System testing is somewhat like Integration testing except that it focuses on ensuring that the complete build aligns with the functional and non-functional requirements. Lastly, we can mention acceptance testing which evaluates the system’s compliance with the business requirements. Testing like many other processes of the Software Development Life Cycle can be divided into different stages. It is also for that reason that we mainly employed unit testing in this project. When the complexity of this project increases, namely adding a second software unit like a frontend with React, Vue or any other UI framework, we could leverage the other stages of software testing. With a client/interface we could leverage the integration tests to ensure that the backend and the frontend are connected appropriately. Similarly, when the product is nearing the minimum viable product, acceptance testing can be used to ensure that all the business specifications are being met.

I have been able to work on this project with a mindset of curiosity, openness, and acceptance. I have been able to work on this project with the mindset that every single bug is an opportunity for improvement and not a reason to be frustrated. The code review process is a vital part of software development, but it is also the most vulnerable to human bias. The reason for this is that as the person who developed the code you can meet high percentages of coverage but that doesn't mean that the quality of the tests is good. Although code coverage is a very helpful tool it is not perfect. If a test touches a line of code the coverage will raise, however, a developer can simply assert null or create non effective test that might increase the coverage of the test but not the quality. Thus, it is very important that the test is not simply written to cover majority of the code base, but the quality of the test should be up to standard in order to follow best practice and have good high-quality software.

Davu, S. (2020, April 13). *The four levels of software testing*. Segue Technologies. Retrieved October 13, 2022, from https://www.seguetech.com/the-four-levels-of-software-testing/

*Software testing stages of agile and traditional methodologies*. UTOR. (2022, August 30). Retrieved October 13, 2022, from https://u-tor.com/topic/software-testing-stages-explained#:~:text=Generally%2C%20four%20main%20stages%20of,System%20Tests%2C%20and%20Acceptance%20Tests.