Project Two

Eric Schoelen

6/19/2022

The full extent of my approach was to meet the requirements of the software and the rubric, this was done by the use of testing. The functionality of the features was checked to see if there were any runtime or other errors, this was done with JUnit testing. Coverage percentages let you know that the JUnit tests were effective if the percentages are high because it takes the total lines of code and how many were involved in the tests and division is done to give you a number to compare. Ensuring the code is technically sound comes from testing, testing, testing. Making the code efficient is done by making it as simple as possible but still making it do what needs to be done.

Testing and quality of testing is very important. We need to test to see if the results are in line with the software specifications and match what the requirements are. Quality testing involves running multiple tests while testing many different types of input to see if the results still come out as predicted or as expected. Testing with multiple types of input lets you see if there are any errors or bugs that could be used to exploit the software or cause it to run incorrectly. Clean code and simple code help with this as well because if there is an error you can easily change something rather than having to dig through unnecessarily long and excessive lines of code. If the program returns the expected results with extensive testing, then you can feel confident that you have produced good software.

Some of the software testing techniques used in this project were structural unit testing, JUnit testing, Automatic testing, checklist testing, and exploratory testing. Structural testing was used to check the integrity of the code. JUnit testing was integrated into the code itself. This was done in the IDE itself by including the JUnit library in the project. Then the test case was run, and results were produced to show if the conditions were met and if it was what we intended or not. As for automatic testing, codes and scripts test the software. Exploratory testing was used to inspect the code to make sure that it followed standards and was functional.

There are many testing techniques not used in the project, some of those are manual testing, Ad Hoc testing, Static testing, and Dynamic testing. Manual testing is just as it sounds and should only be used in small projects. Ad Hoc testing is a method that does not follow any set plan and is really only good for small projects. Static testing is done on the code without execution. Dynamic testing is done while executing the code and is great for complex code.

For this project I tried to keep an open mindset, one that considered quality, caution, and logic. I employed caution because it is important to look at all aspects of the code and what could go wrong if there were errors or bugs that could allow others with malicious intent to cause harm in many ways. Logic was a big part of my mindset. This is because we must think logically when writing code and testing, all aspects of a project must be logical. Quality is of the utmost importance when working on a project. It is important for the code, testing, and project as a whole is of the utmost quality so that when you present it to the customer, they are happy with it and that it does what they have asked. The quality of your work is not only important to your customer, but also to yourself, you will be judged in your career based on what you produce and producing poor quality work will affect you negatively.

In trying to eliminate bias I tried to keep an open mind to different ways of coding. I might feel like one particular way of doing something is the best or correct way, this however, could be incorrect for this project and could possibly cause the software to have unintended consequences even though it works properly, such as the case with Amazon and their trained AI bot.

Being disciplined in the commitment to quality is of the utmost importance. It means that you care about your finished products, your career, your customers, as well as your reputation. Cutting corners is never good when it comes to quality, you will end up with problems that you did not foresee as well as leaving you responsible for future issues with the software and in part damaging your reputation as a designer/programmer. When you ensure your software is always quality work then you are helping to eliminate the technical debt in the future, and you will have less to fix down the road and you will have a much better reputation for your work.