Project 2

Unit testing has various different approaches available to use in order to test input within the code being written. Unit testing is a form of testing in which test cases are based on the internal structure. For example, we would choose inputs for the code to check the different outputs for our inputs. The main goal in unit testing is to check different classes of code to ensure they function correctly and output what they were implemented to do. Regression testing goes with unit testing. When using regression you can check to make sure that a new fix or update in code doesn’t change functionality or mess it up by checking the changes to the code. Code will always most likely change at some point updating things adding features and so on therefore regression is very important to use for those reasons.

Junit testing also known as Java unit testing is one of the best regression test methods to use. It is used to write and run repeatable automated tests for your code. This was our testing approach in the features of our program we built through this course. We can tell that our code was tested effectively through what is called branch coverage. This will let us know how many decision outcomes were tested on a percentage basis, so the higher the coverage percentage is the more outcomes were tested effectively. To ensure our test code was written effectively we only test a single code unit per time, make each test separate from the others, and naming consistently throughout the tests. Ensuring the test code was efficient throughout test coverage this tells us we effectively test what needs to be tested through getting a high test coverage percentage output of our code.

Throughout our project we used a few different testing methods, some being unit testing, regression testing, and junit testing. These are all forms of testing how our program functions but each technique is designed to find a different type of defect. Junit is the one we used the most and it basically runs automated inputs through the code multiple times to check for errors and tell us how much of the code was covered in our test. Practical uses of junit testing is to create a set of unit tests that can automatically run when any change or updates are made to the software. This can ensure our changes or updates do not break how the software ran previously and keep our current functionality.

I had a problem solving mindset when working on this project because I am trying to solve any problems in the code as a tester in order to get functionality to where it is needed. Employing caution when needed for example fixing a problem may create different problems so use caution when updating or changing the code to make sure you aren’t creating more work for yourself later on. Limiting bias in code can be a little harder when it comes to code that we wrote and tested ourselves, but to help overcome as much as we could we reviewed the code numerous times in order to make sure it was correct and functioned as needed. In most cases you would have developers and testers separate therefore you would have multiple people reviewing the code to ensure it was correct. In this case we only had a developer that was testing their own work so we did try to limit the bias in our review as much as possible.

It is super important to be disciplined when it comes to being a software engineer because it takes time and patience to have a successful piece of software. You are given requirements and goals for the software to meet and you have to figure out how to implement each and ensure they all function together properly. Once all your code is written a whole other job comes into play of testing the code written which can be tedious because you may have to go back and make some quick fixes without damaging any of the other functional parts of the code. Being a perfectionist helps in this field because you don’t want to take any shortcuts when it comes to this because this may cause errors in the near future that could have been avoided with just investing a little more time to make sure it was done right the first time. This will also reduce technical debt because you won't be having any additional work later if you did it right and didn’t choose the quick and easy solution over the proper more time consuming solution. Also making sure your work is done right will make you look more professional and of course the consumer will have a much better experience with the software.