How I used a unit testing approach for the three features in project one. This was alinged with each of the software's required to that I needed to write a test for all the required functions. I used a variety of techniques to test that including equivalence partitioning, Boundary value testing. This ensured that my software requirements were met. I wrote my JUnits test in a modular way to make it easier to maintain and debug if there were to be any errors. Then I used various assertions in my tests to verify the results were correct. The percentage of my Junits tests did not show errors in the testing, but in the percentage of coverage did not reflect how it was supposed to be. To that I would still need to work on to make sure that my code is more accurate and effective

The Software Testing Techniques that I employed during this Project were Unit Testing, Equivalence Partitioning, Boundary value testing. The Units testing technique is that it involves the testing of individual pieces of code, That is essential for ensuring that code is correct and functional. The Boundary value method is a Black-box Testing technique that involves the testing of boundaries of both input and output values, this is what help ensure the code can handle the values correctly. The Equivalence Partitioning is also a Black-box testing method that divides the input space into equivalence classes, this would help make sure that the code can handle all possible inputs. The Decision tables technique is that it uses a table to specify the possible combinations of both input and output. This would help the code ensure that it handles all possible combinations.

The other types of software testing technique that I did not use during this project were System Testing that is a Black-box testing technique that involves the testing of the whole system, that is done after integration testing. This would be done after integration testing is complete. The Acceptance testing is a testing technique that I did not use during the project that involves testing by users. This would be done after System Testing is complete.

How each of these testing Techniques that I have discussed have practical uses and implications for different types of software development projects. That Boundary value is a good choice for when project where input or output are critical. The Equivalence Partitioning is also a good choice for when a project the input space is complex. The Decision tables are also a good choice for when a project has a large number of possible combinations of input and output. That Unit Testing is also a good choice for any software development, regardless of complexity or size.

The Mindset I adopted while working on this project was caution and precision. This that I was aware the complexity of the code that I was testing, and to test possible combinations of inputs and outputs. But also, to try to limit my bias in review to my code that of all possible scenarios and not personal experience. That to also have more help in having other software developers review my code for possible errors that may cause the JUnits to be a low percentage when there was no errors. For the software developer side, a bias can be a concern if that developer is responsible for testing their own code, that they are more likely to overlook their code. To limit this bias, would be willing to have help from other developers that can help overcome the bias.

It is important to be disciplined in your quality as a software engineering professional. This does mean no cutting corners when writing or testing code. This also means that to be willing to invest time and effort into improving the quality of work. This also means to review your work to check for errors and having other software developers review your code aswel to not have a limited bias on your work. This being disciplined in your commitment to quality means that it can help ensure that your software is reliable, and the needs of your users are met.