* How can I ensure that my code, program, or software is functional and secure?

I would have to continue to use the best practices that have been teach to me and to not try and cut corners suing code that I don’t know what it might do, I say this so that others and myself all have the same concept of the same code if another person use’s different code that is not known that could hurt the project and or slow the project down since it's code we don’t recognize. The same basics that I have learned other before me and after me will used the same code in our to maintain quality since the codes is the same we can all be on the same page as what coding language we are using to build the system and to keep it running.

* How do I interpret user needs and incorporate them into a program?

I would incorporate their needs and only their needs in the code, in order to stick to the plan. As before I wanted to be different and add personal touches. I was advised not to, which I did not think of is a good thing since they would be a waste of time, and since the request was not advised by my team leader the time wasted could have been used elsewhere, where I could have used my time more wisely and built a better system and better code usage and have a better understand of what is needed of me to deliver in the code. I best way that I have found during my time as a student would be to write down the needs and wants of the designer and this way I can revert to them as a list on a note pad to refer to the notes to see if I have answered all the questions or points the person wants me to bring into the system and what points the person wants me to bring to the forefront to build the system they are needed to solve there problem.

* How do I approach designing software?

Keep it clean, using code that I know and that other can pick up where I left off is great practice since we all should be writing the same code since we are all being teach the same way to make code. But some code might seem easier but if we don’t know what the code does the code might be more complex than our understanding and that code could bring problems to the code and to the problem being built. Keep it simple is also a good understanding since we want others to know what is going on in the code and what should be the outcome and to be able to pick up where we might have left off and build after us and build upon our code to better it than to erase it and start fresh since the code is to bad that it can't be used.