Development plan

1. Process raw data
    a. Calculate the torque of the motors with raw data
    b. Include gravity to make it more accurate
    c. Have an easy to read a chart to show data

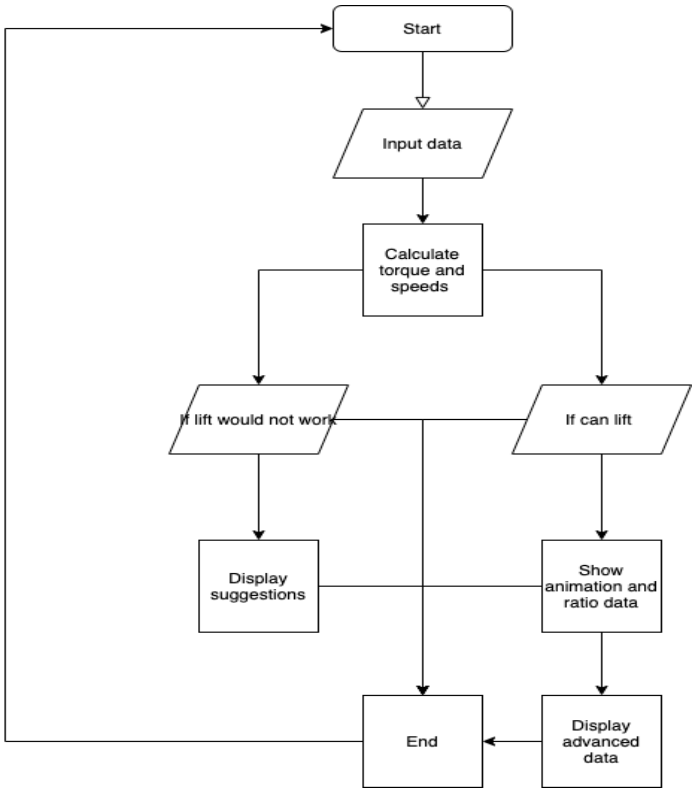
2. Dynamic data
    a. A secondary mode that shows motor running in real-time
    b. Data can also be manipulated in real-time
    c. Updating data display

3. Suggestions
    a. Offer advice if the motor would not work
    b. Offer advice to make sure the motor works predictably
        i. Simulations rely on ideal circumstances which don't always reflect real life

4. Clean up and optimize program
    a. Make the UI more user friendly and professional
    b. Get rid of unnecessary tools made for production
    c. Make sure there are no input errors

5. Finalize testing
    a. End-user testing and final program

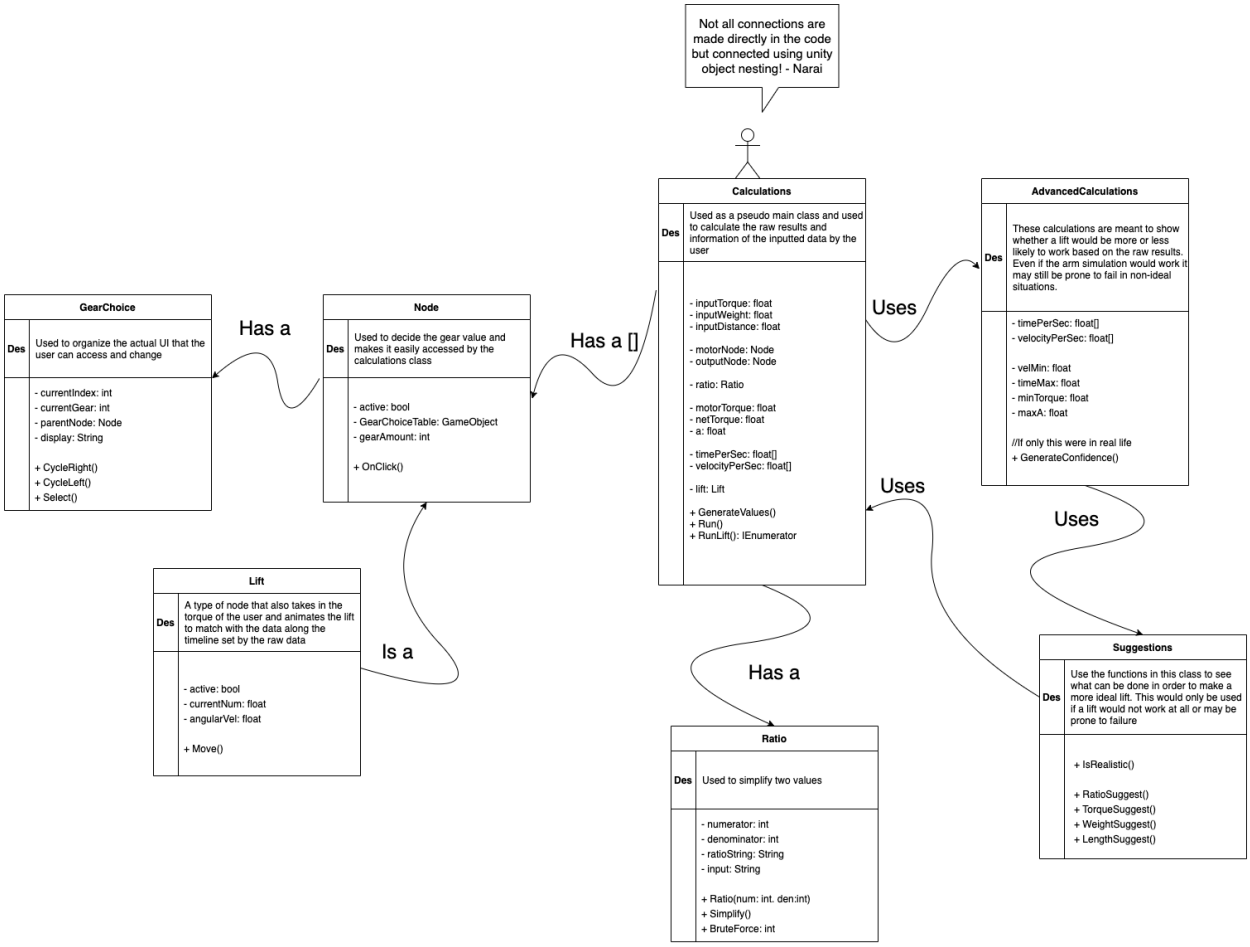
To the right we can see how we would originally want the program to run and process however through the development of the applications my client and I discussed a new way to make what they wanted. Instead of having a static calculation the motor would constantly spin and could be changed during runtime to see live changes in the motor which would be a better representation of how motors would work in robotics.



When starting the program there are technically now two different modes you can select but they both operate in almost the same way. The only difference is that one updates in runtime while the other

makes predictions and calculates based on unchanging variables.

The diagram to the right would show how the flow of the unchanging data during processing would work. If it were dynamic then the ‘input data’ node would be accessible from any other point in the process.



The main method that we use to calculate the torque added to the motor is combining the induced torque from the motor itself and then get add the force added by gravity (including vector direction).

Torque equation: $T = rFsin\Theta$

This means though that we need to recalculate the nettorque for each update to the rotation of the motor. The main difference is that in the dynamic calculations this is constantly updated while the alternative only needs to be put through a for loop to calculate the change in torque over time.

The issue for needing to input the force caused from gravity came to my concern because if the weight was not lifting something significantly heavy then we could neglect the weight but since we need to make sure that it not only can more accuratly simulate reality this program is also used to test motors and gears so that it has a higher chance of working in real life. A major cause for them not working properly in real life is subsequently the weight which needs to be taken carefully into consideration.

A concern for that came to mind within this project was also that user in real life can program their vex robots to work based on velocity rather than force. This however was typically used for wheels instead of lifts and didn't need to be tested before use for physics based problems. Since we use torque we are able to make easier calculations for the user. This mainly though comes down to the fact that we do not know the exact method VEX uses to update their motors using velocity.

Most of testing will come down to situations that have random inconsistent data input. For staitc data we need to make sure that no input can break it so there are the correct catch catches but also that it catches impossible situations and uses the suggestions class when prompted. The dynamic simulation would be similar but also for quickly changing the variables in runtime and making it sure it does not break because of that.