As we started brainstorming for what we wanted to do for this project, we began to think of what we work with every day in our classes. We gathered a list of ideas, including balancing reactions, Van der Waals equation of state, matrix calculations, mole to mass calculations, and more. Although we do use all of these, we wanted to choose what is most applicable to what is used every day in our classes, homework, tests, and in industry. As a result, we came to a final idea and proposal that was approved by the instructor, Quinn Lanik, during her office hours.

When approaching this project, we wanted to improve the speed of a process that is used to solve nearly every problem in our industry. Chemical engineers work to improve and optimize chemical process that are in both everyday life such as water treatment and more specialized such as nuclear power plants. While optimizing these processes we are often presented with a reactor or multiple that have a chemical reaction occurring inside. In order to determine the deficiencies in the process we must first convert multiple inputs or outputs from moles to mass. A mole is a universal unit for the number of particles in a substance, this may be atoms, molecules or ions. In one mole there are 6.022 x 1023 particles. Converting to mass allows for the outcome or origin to be determined from its opposite. Most often, the amount of each reactant is given in mole fractions, then later converted to mass of each separate component in the problem.

With the steps of the process being pretty straightforward, the structure of the code was able to be loosely determined by breaking functions down into each step. First, the molar fraction and molar mass are required for the next steps. These are numerical values that are determined by information given in a problem and molar masses derive from a chemical formula. To allow more general coding, we designated each part with letters A to H instead of writing explicit components. This is to avoid lack of application in many different conditions, the user will just need to be aware of which is which. Also, if there are to be less than eight components, the user just needs to leave all unused fields as zero. Once these are inputted, the fraction and molar mass are multiplied in order to find the mass of each component. A basis of 1 mole allows the program to keep the molar fractions as a decimal, whereas if we set a basis of 100 moles, the fractions would look like percentages. Although it may look more appealing, doing so causes more work to be done in our later calculations as chemical engineering majors. The calculated masses are then summed, and each mass is divided by the sum to find the mass fraction. Others may think that mass and molar fractions are the same, but they are different values, as molar masses vary greatly.

For the functions that require the user to input values, there were not many obstacles. Slight difficulty came with finding the total of the mole fraction column, but once it was figure out, we were able to then implement some error notices. Inputs that would not allow calculations to be run correctly include values less than 0 or nonnumeric inputs. There are notices in place, including an error popup and a general box under the column that prompts the user when they have correctly entered their numbers. Overall, the total must equal 1.00 for the values to be correct and applicable in real life applications.

The true difficulties of this code occurred when the inputted values needed to be multiplied in order to find the mass of each component. Because we had already used them in a different callback, help was sought out from LA’s and the instructor after research into the issue was of no assistance. Five separate LA office hours were attended for this program, but sadly each one did not help us resolve the problems that were occurring. It seemed as if the issue we were running into was one that they had never seen or were not sure where for the program was encountering obstacles. We tried to use two callbacks for one component, but when that was done, previous functions were no longer running correctly. Finally, one of us was able to meet with Quinn in her office hours and we were able to fix the issue, along with completely finishing the code with some assistance. In hindsight, what we thought would be a pretty straightforward and simple task ended up being surprisingly difficult and, in Quinn’s words, “a very ambitious project.” However, as a result, our understanding of GUI and its applications in MATLAB increased greatly. Additionally, we now have more knowledge of how to manipulate GUI’s many capabilities and troubleshoot in ways that were not discussed in class. A final issue that was encountered was that one of our member’s git bash was not working as intended for her, as she is on a Mac computer, so the other updated the repository after getting her work by other means.

There seems to be a multitude of ways that we may be able to implement GUIs in the future. With reactors, separators, distillation columns, combustion reactions, and many more aspects in our field, we believe that GUIs and overall MATLAB will be very applicable in our studies. By modeling these processes and, in turn, coding programs to lead to calculations with them, we believe that it would help us substantially and lead to efficient problem solving that we currently have to do by hand. During this semester, we worked a lot with the systems listed above and would have to solve each one independently rather than have an easy was to be able to balance all components across many streams, outputs, and inputs. If we were to make a GUI for this, theoretically, it may be of use and end up with less time spent on repeating the same process many times manually.