## Architecture

Our group chose to use Model-View-Controller as the architecture for our system. Using Model-View-Controller allows us to separate data(model) and user interface(views) so that data handling is not affected by the user interface, and that data can easily be changed and moved without the need to change the interface, through the use of the controller. The model represents the knowledge or data contained in our system. The views are the visual representation of the system, mainly the graphical interface. The controller is the link between the user and the system, it works with the model and views to control the user input.

The main reason we decided to use Model-View-Controller for our system is that it always that it allows you to easily separate the different components of your system, which allows for increased ease of re-use for classes particularly in the model aspect of the system. It also allows for easier maintenance and testing as each component in the system can be separated and independently tested or altered without affecting the overall system. It is a great way to keep our code functional and maintainable as having everything separate allows you to see exactly what each piece of the system is doing and not have everything mixed up and doing multiple things.

Another reason we chose to use Model-View-Controller is that it is the architecture we as a team are most familiar with and felt we would be the most comfortable using in our system. This plays a major role in our decision as being more familiar with Model-View-Controller we are able to implement our system more easily and more effectively than if we were using another architecture with which we are unfamiliar. We also feel as though Model-View-Controller is very fitting for our system as it is highly UI based and this allows us to easily handle the interaction between the UI with the data through the controller.