Software Architecture

We are using **3-tier software architecture** because our prototype class structure is based on three semi-distinct tiers of functionality. This allows us to divide up the large application into separate pieces that can work relatively independent of each other. Similar to the decision of using object-oriented programming, this structure allows the work to be effectively divided up between teammates, while not having so many tiers that it causes unnecessary confusion.

The bottom tier is the presentation tier: the front-end user interface of our application. We created a Bootstrap powered web interface which the user sees and interacts with to create their graduation plan. It allows users to select their major and starting semester, the drag-and-drop courses into the desired semesters. It renders the prerequisite arrows between courses and displays the total credit hours in each. It informs the user of any potential issues such as when prerequisites are not met. There is also a navigation bar for quick access to useful locations like the KU's schedule of classes and enrollment.

The middle tier is the logic tier, which links presentation and data tiers together. The logic tier receives events from user interaction in the presentation tier, such as dragging and dropping a course. It then acts upon them, making any necessary changes in the data tier and informing the communicating back to the presentation tier as to how it should update its display. One of its important roles is validating the plan the user creates by checking prerequisites, credit hours per semester, etc.

The top tier is data tier which stores two different kinds of data. The first is static data: the list of available courses and majors (which only includes Computer Science in our prototype), and is bundled with the source code. The second is dynamic data: the plan created by the user, the semesters the plan contains, and the courses each semester contains.