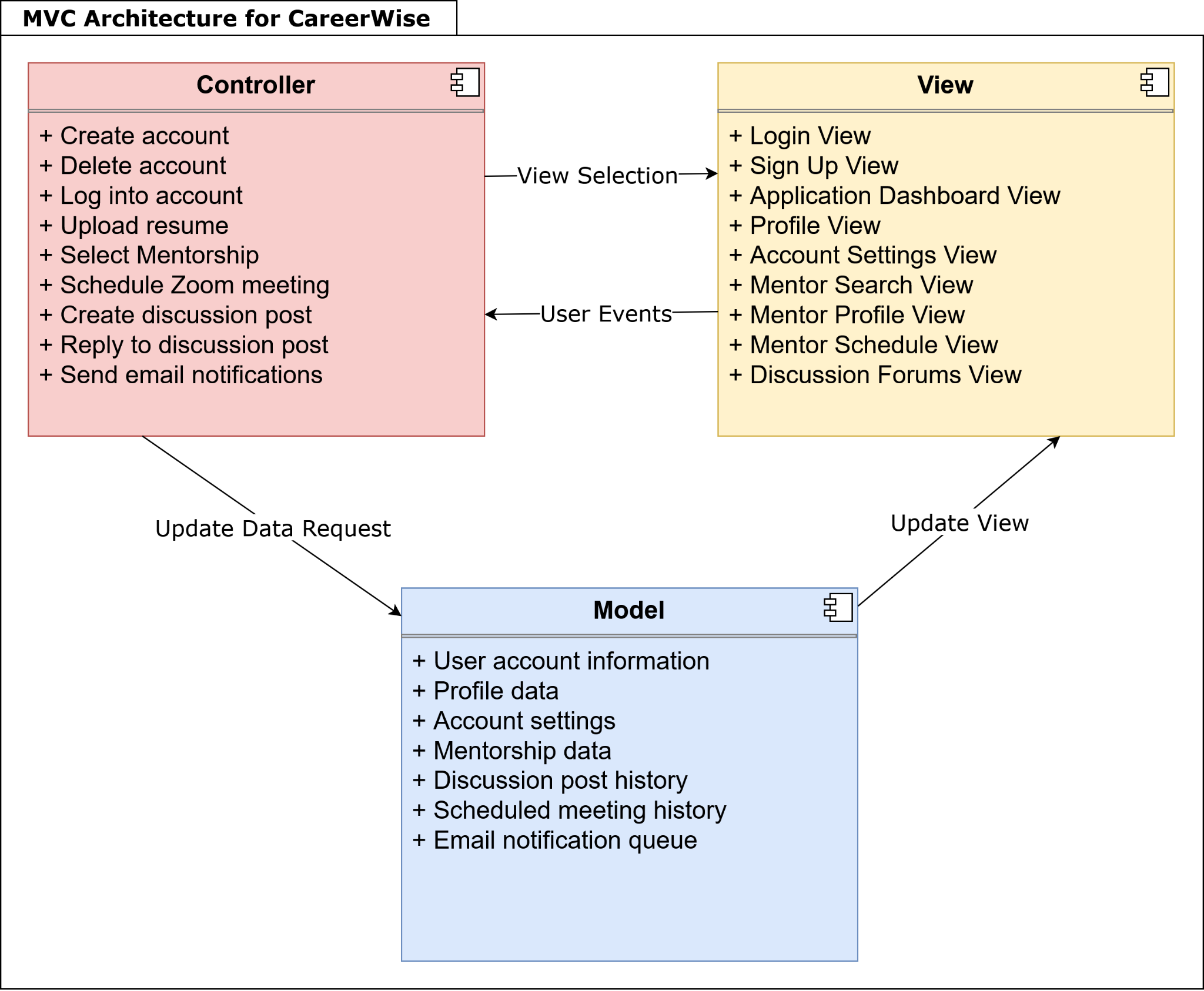
**Team A (Asynchronous)**

**Architecture Design:**

For designing the architecture of CareerWise we designed a Model-View-Controller architecture. This architecture was chosen because it provides an organized method to present the user with the most recent information regarding user’s mentorships on the application dashboard. It also provides an effective method to receive and store information from a user, such as their questionnaire answers and their resume. This architecture will simplify the process of storing user information with Firebase and displaying information back to them.

**Architecture Design Diagram**:



**Design Description:**

The architectural pattern consists of three components, the view, the controller and the model.

The view for CareerWise handles the UI of the web application. This is mostly handled by React and Bootstrap. The application will have many views that can possibly be displayed. The Login View will display when the users first open the application. Upon a successful login, an “application dashboard” will display that provides a user with shortcuts to the application’s vital functions. Any interaction with the UI can trigger user events sending information to the controller to handle the inputs. Mantine will be used for the React components that it provides, including text fields and date pickers. Chosen dates and times, such as for scheduling meetings, are also sent to the controller to be handled.

The controller handles the various events from the view. Most events are simple button presses, such as for a user to upload their resume, answer multiple-choice questions in the questionnaire, choose a mentor, or choose a date and time to schedule a mentor meeting. When an input is detected, the controller then notifies the view of any changes if necessary and then notifies the model of any state changes.

The model for CareerWise handles the data from the app. Here the view can access any data for specific elements in the app to update the UI. Any state changes notified by the controller will be stored in Firebase and a notification of a change, if necessary, can be sent out by the model to the view controller. This will allow for the view to be updated when a new item has been added or a new message has been sent out. This communication between the database and the view controller is imperative in updating information on the application. Day.js will be used alongside Firebase when storing date information such as meeting dates. Finally, the model is also responsible for making API calls when notified by the controller. Zoom’s API will be used to schedule virtual meetings. PayPal’s API will be used to assist in processing payments between mentors and mentees.