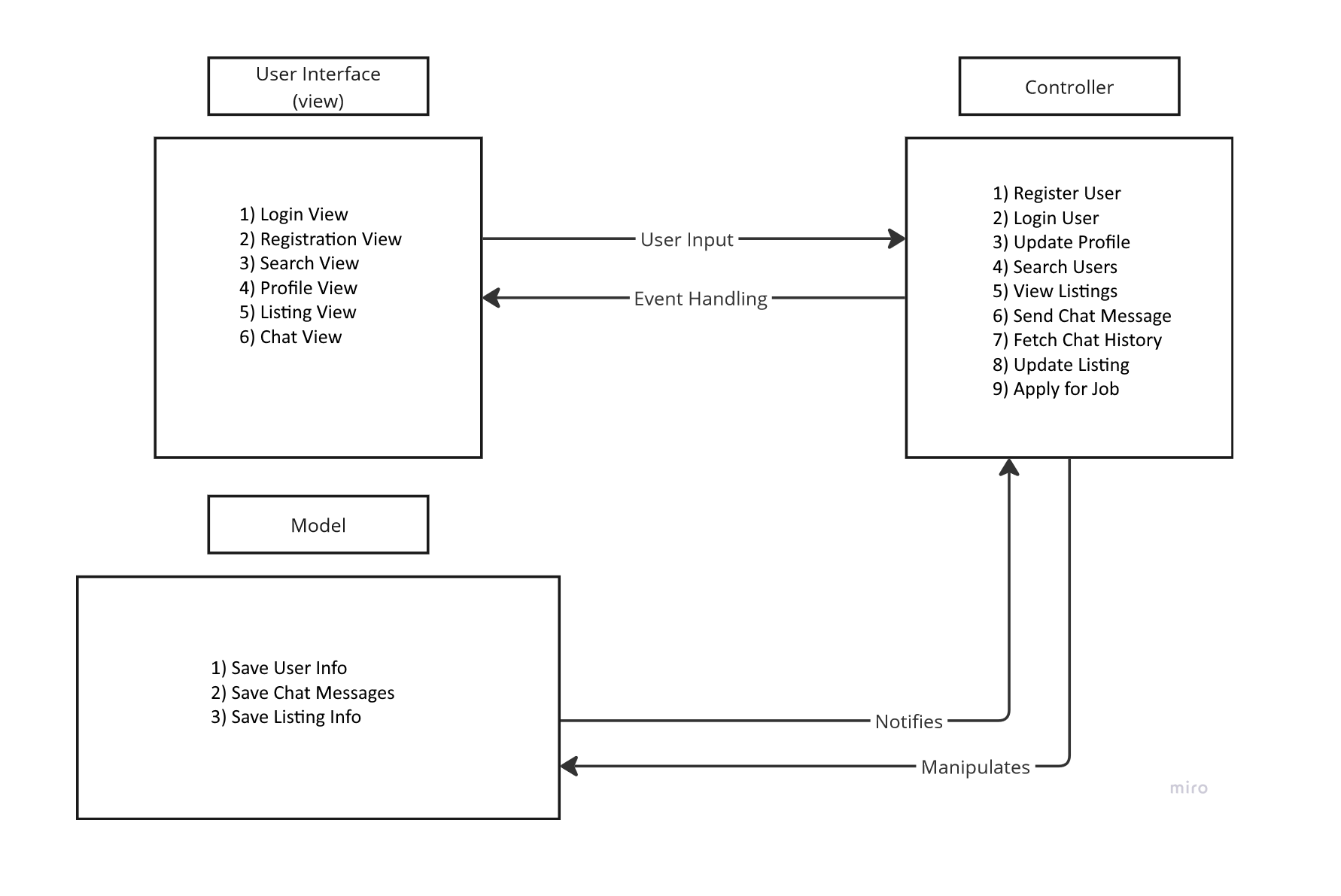
**Team: Ace**

**Architecture Design:**

For designing the architecture of EmpowerMaine we employed a Model-View-Controller architecture. MVC was chosen because it is an industry-standard for creating web applications and is a logical separation between server logic, client logic, and database operations. This separation not only improves the reusability of our various components but improves the testability of our application as each component can be tested in isolation. MVC also increases the scalability of our application by allowing the router to route API endpoints to their respective controllers.

**Architecture Design Diagram**:



**Design Description:**

The architectural pattern consists of three components: the view, the controller, and the model. For storing database information, Firebase was chosen as it reduces the need to host your own database and provides a console for easy access and management.

The view for EmpowerMaine handles the UI of the web application, which is mostly handled by React. The application will have many views that can be displayed. The Homepage View will display when the user first opens the application. Any interaction with the UI can trigger user events, sending information to the controller to handle the inputs.

The controller handles the various events from the view. These events can be a simple button press to inspect a user profile, log in or out of an account, submit a job application, or for a recruiter/mentor to create a new listing. When an input is detected, the controller is invoked with data sent by the client, enabling it to interface with the model to save, fetch, or update data as necessary, and return server response information to the client in the form of JSON.

The model for EmpowerMaine is responsible for managing the application’s data. The controller interacts with the model by calling functions that communicate with Firebase to access or modify specific data elements. When the model processes any changes, it can notify the controller, which in turn informs the view to refresh the UI with any new updates, as needed.