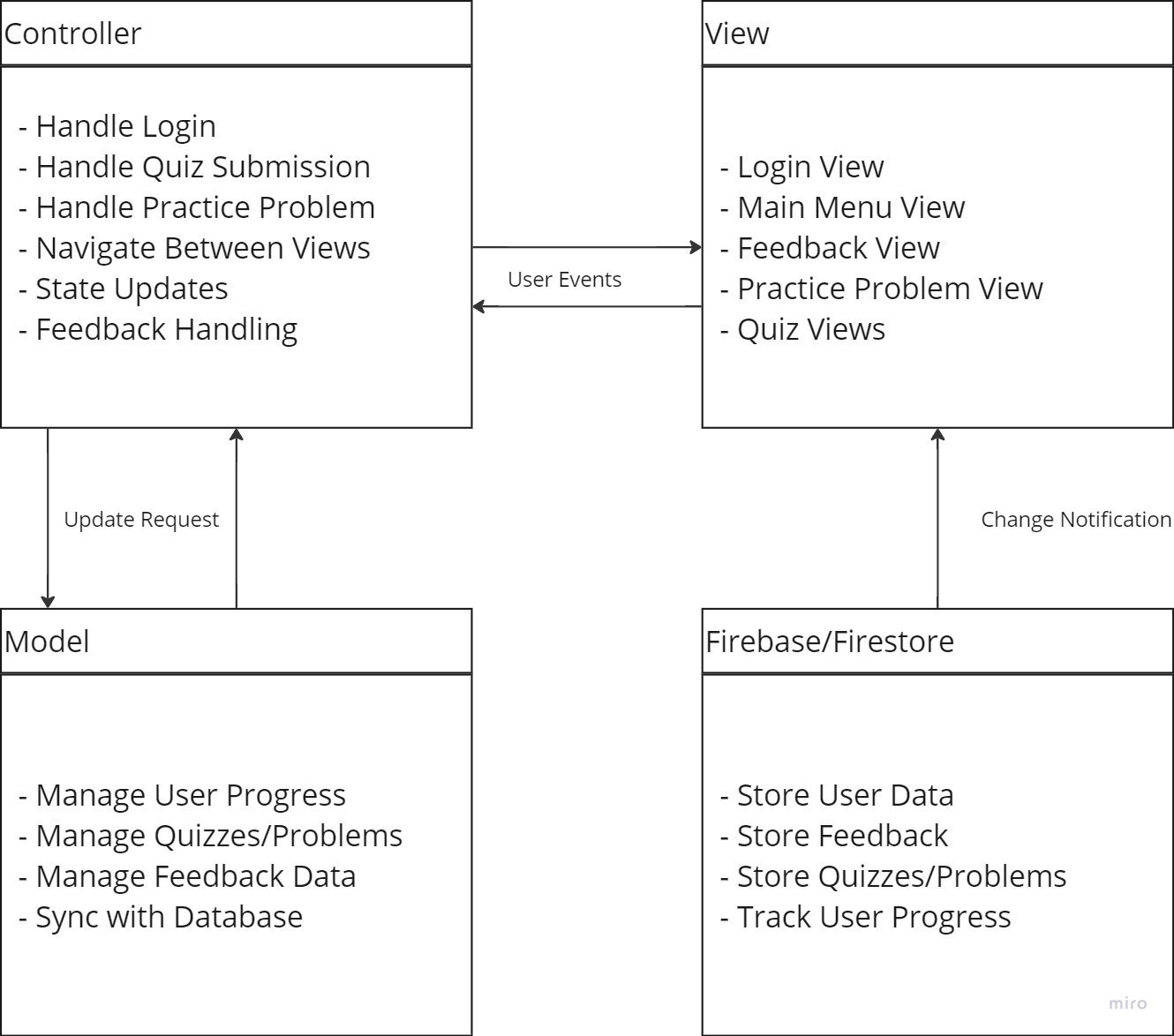
**Team: Brilliance**

**Architecture Design:** [**https://miro.com/app/board/uXjVLPyDaEw=/**](https://miro.com/app/board/uXjVLPyDaEw=/)

*The decision to use* ***MVC*** *is justified based on the following considerations:*

* ***Separation of Concerns****: It allows distinct separation of business logic, UI, and control flows, making the app easier to scale and maintain.*
* ***React for UI****: React is suitable for rendering dynamic views, which is essential for an app that frequently updates its content (e.g., quiz progress, feedback).*
* ***Firebase for Data****: Firebase provides real-time data syncing and user authentication, which is critical for user progress tracking and providing instant feedback.*
* ***Scalability****: As the project evolves, the MVC pattern allows adding more features (like new quiz types or user feedback functionalities) without major rewrites.*

**Architecture Design Diagram**:



**Design Description:**The architectural pattern for **IntuiCode** consists of three main components: the view, the controller, and the model.

**View Layer:**The view for IntuiCode handles the **user interface (UI)**, primarily using **React**. React enables the creation of dynamic, component-based UI elements, which respond to user actions in real-time. **CSS** is used for styling, ensuring the app is visually appealing and responsive across devices. Key views include:

* **Login View**: Where users sign in or create accounts.
* **Main Menu View**: Giving users access to various learning paths, resources, and feedback.
* **Quiz and Feedback Views**: Allowing users to solve coding exercises and receive instant feedback.

We also plan to use **React Router** for handling navigation between views, and **Axios** for making asynchronous requests to fetch or send data to the backend.

**Controller Layer:**The controller manages the interactions between the view and model, processing events triggered by the user and updating the UI accordingly. For example:

* Selecting a programming skill from the main menu.
* Submitting code exercises for feedback.
* Navigating between different resources like videos, glossary terms, or example code.

The controller handles this logic, updates the state, and coordinates changes to the model when necessary. **React Hooks** such as useState and useEffect will be used for managing component state and side effects, while **Context API** or **Redux** could manage the global state of the app.

**Model Layer:**The model layer stores and manages the app's data. This includes user profiles, progress data, coding challenges, and feedback records. We plan to use **Firebase** for user authentication and **Firestore** to store real-time data like user progress, quiz attempts, and feedback responses. The model communicates with the controller to send or retrieve data and ensure that the view reflects any updates in real time.

By adopting this MVC architecture, **IntuiCode** ensures a clear separation of concerns, allowing for easy debugging, future scalability, and maintainability. Using **React** for the view and **CSS** for styling ensures that the user interface remains intuitive and responsive as the app grows.