Class Diagram

[Miro design class diagram (View)](https://miro.com/welcomeonboard/cm04K0M3QTZ0elVRdStCTTVBelFMb2hVejFSZFZ0T0pBcktEQ2diM2xmUnVON0cwaGNoWFdSNVBRTlJjWXB6SnFaNktJd3lQVVY4K2N2SjdiTXR5Nlg3QldxT0NGOWNJOTQ2OTRwR25ubllrbCtva0x1WXU1NXUyOGRZZ1NMekghZQ==?share_link_id=12683504397)

The IntuiCode application's many React view controllers are connected to Firebase, React Router, and data structures to facilitate its operation, as shown in the diagram below. Every view controller has a defined function within the application and is designed as a React functional component. React Router handles page transitions for the application, while Firebase Auth handles user sign-ins.

The primary component of the application is the App component, which links key elements like the LandingPage, AuthPage, and course-specific sites and reroutes users to the appropriate pages based on whether they are signed in.

Core Features and Connections

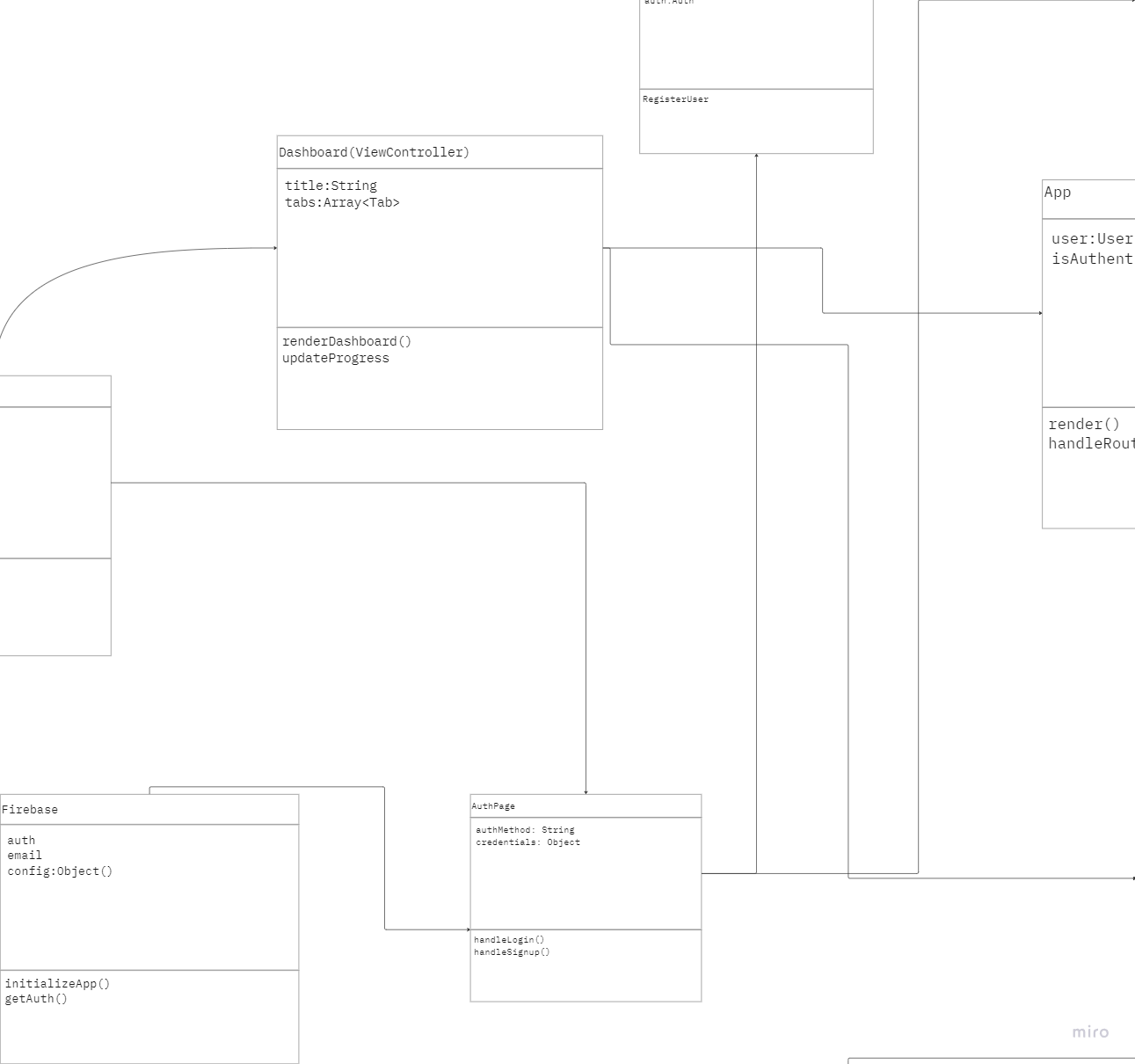
1. App Component:
   * Serves as the root of the application and manages the user's authentication state using Firebase Auth.
   * Routes authenticated users to the LandingPage and unauthenticated users to the AuthPage.

Relies on Firebase Auth to handle login/logout via the handleLogin and handleLogout methods.

1. LandingPage Component:
   * Displays a dashboard for logged-in users, with tabs for:
     + Interactive Exercises
     + Progress Tracking
     + Quizzes
     + Feedback
     + Streaks
2. AuthPage Component:
   * Manages user login and signup using Firebase Auth.
   * Includes methods such as handleLogin and handleSignup to authenticate users.

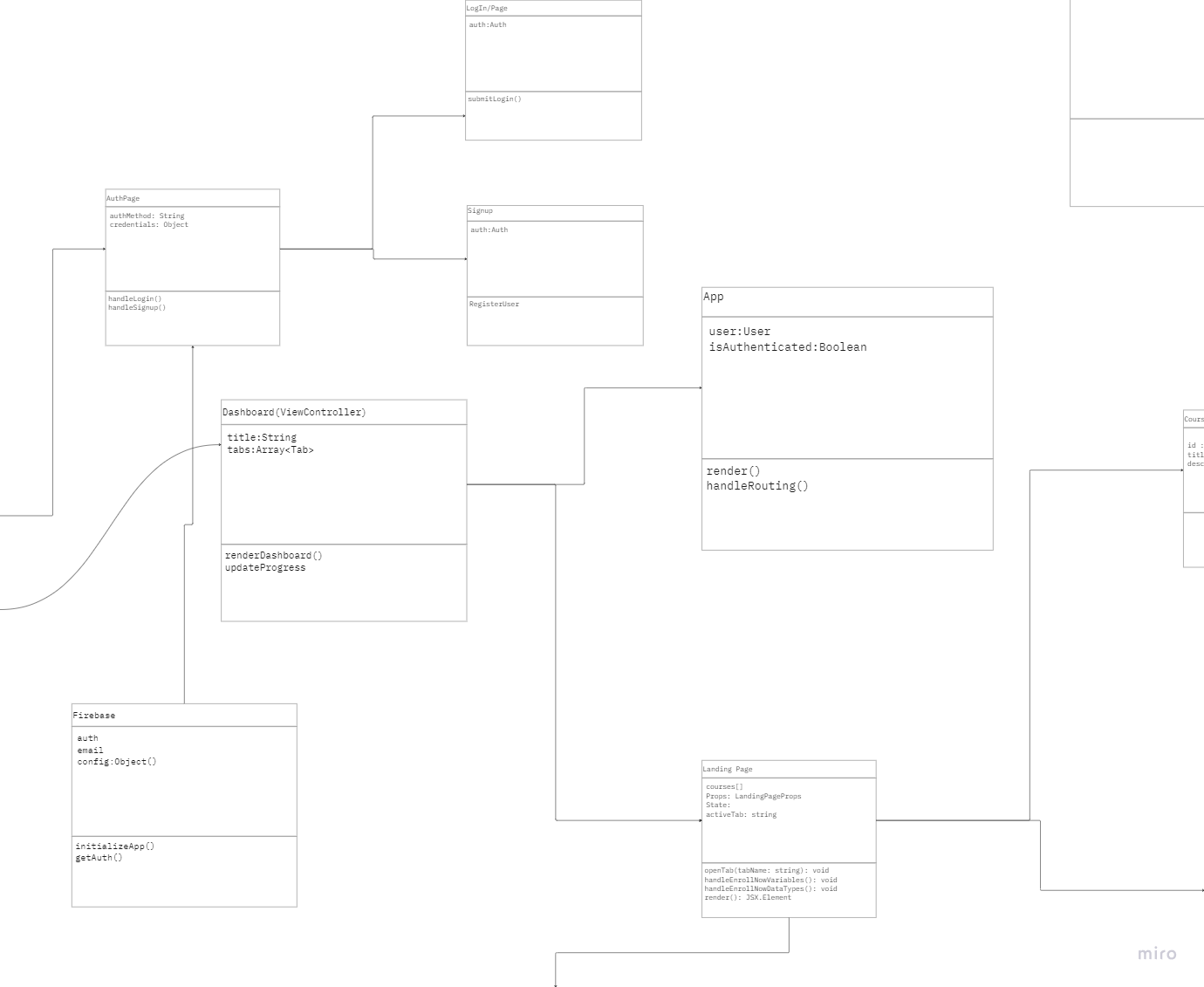
etc…

Database Tables



The diagram of the IntuiCode application shows the implementation of Firebase as the main backend support for user authentication and a possible future database for storing user information. Authentication with Firebase Auth enables secure user login and session management in the app.

Design Patterns



[**Miro link for full view**](https://miro.com/app/board/uXjVLLZ8KDg=/) **of DCD**

**The IntuiCode application is presented by this DCD, following the MVC design pattern with clear divisions of tasks. A simple explanation is presented here:**

**1. Model**

The Model would be the data layer responsible for maintaining data storage and business logic:

**Firebase** -

The main backend system where the initializeApp() and getAuth() methods are called to manage user authentication and configuration.

Maintains user state (auth) and may interface with a database for storing user progress, user courses, and user streaks in future enhancements.

**2. View**

The View is the UI component that is responsible for rendering data to the user and processing user input:

**LandingPage (ViewController):**

This is a UserControl's dashboard for signed-in users; it renders tabs such as Exercises, Progress, and more.

It contains attributes like props and state for managing user interactions and updates dynamically the view.

Methods include the openTab() for tab switching and the handleEnrollNowVariables() for course enrollment.

**AuthPage**:

This page shows the login and signup UI to unauthenticated users.

Communicate directly with Firebase Auth to take care of user login and signup actions.

**Course-Specific Views (e.g., Variables)**:

These views display content and exercises for the individual courses, linking to the navigational structure coming from the LandingPage.

**3. Controller**

The Controller controls the logic that ties the Model and the View together. It processes events that come from the user, communicates with the Model, and updates the View:

**App Component**:

CENTRAL - Controller: routes and maintains user state.

* **Attributes**:

user: User: State of the currently logged in user.

isAuthenticated: Boolean: Whether a user is logged in or not.

* **Methods**:

handleRouting(): Routes user to different views- for example, logged-in user to LandingPage, and unauthenticated user to AuthPage.

handleLogin() and handleLogout(): Updates the authentication state of users using Firebase Auth.

**Dashboard**:

A controller to render and manage tabs such as ExercisesTab and ProgressTab.

* **Attributes include:**

title: String - the dashboard title

tabs: Array<Tab> - the list of tabs present on the dashboard

* **Methods:**

renderDashboard(): Renders the active tab's content

updateProgress(): Updates user progress dynamically

* **LandingPage:**

Has methods like openTab() for tab switching, effectively binding user interaction with the UI and changes in state.