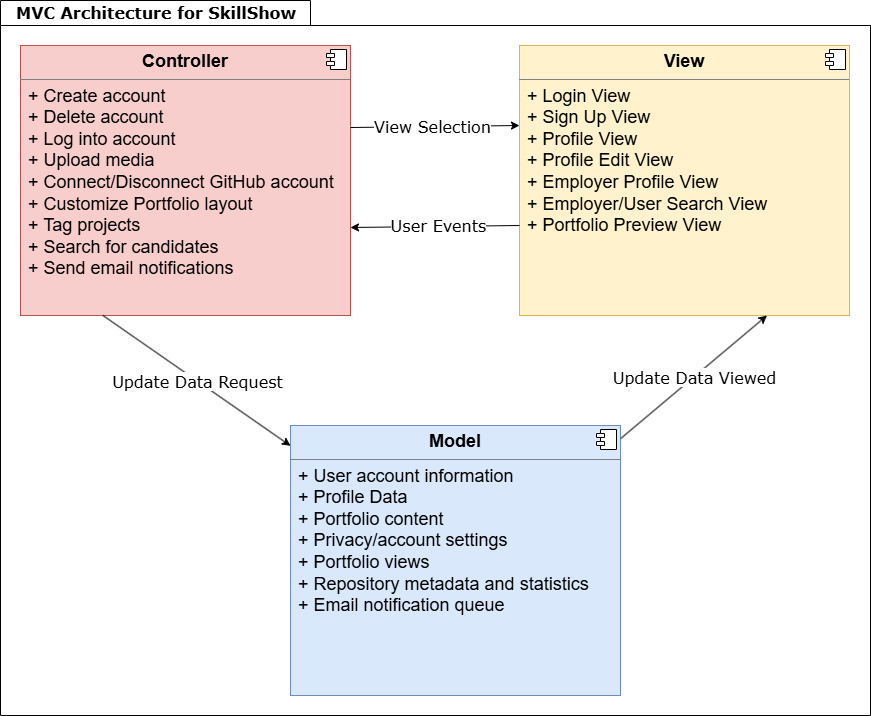
**Team: Dingo**

**Architecture Design:**

For designing the architecture of SkillShow we designed a Model-View-Controller (MVC) architecture. MVC was chosen for SkillShow because the separation of the components makes it easier to maintain the codebase as the project scales up. This particular architecture will allow the team to work on different components simultaneously without creating unnecessary and time intensive conflicts. In addition, this separation allows the team to modify and replace individual components without needing to redesign the entire ecosystem.

**Architecture Design Diagram**:

[](https://app.diagrams.net/#G15ZW-Nfna1xef9g0jNMbmV5vGjaFgxH_Z#%7B%22pageId%22%3A%22b5b7bab2-c9e2-2cf4-8b2a-24fd1a2a6d21%22%7D)

**Design Description:**

The architectural pattern consists of three components, the view, the controller and the model.   
  
The view for SkillShow 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. Any interaction with the UI can trigger user events sending information to the controller to handle the inputs. Libraries and packages that are planned on being included in the view layer.

* react: main front-end library for interactive components.
* react-dom: Renders react components to the browser.
* bootstrap: Provides Cascading Style Sheets (CSS) and prebuilt components.
* react-bootstrap: Lets you easily use bootstrap components in react.
* react-router-dom: Handles navigation between pages like login, profile, search, and etc.
* axios: Makes Hypertext Transfer Protocol (HTTP) requests to Firebase or backend Application Programming Interface (APIs).
* react-icons: For consistent icons across the UI. Like edit, delete, upload.
* react-hook-form: For building forms like login, signup, or profile edit.

The controller handles the various events from the view. These events can be a simple button press to further inspect an item for sale, login or out of an account, to start a conversation with a seller, or to create a new listing. 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. Libraries and packages that are planned on being included in the controller layer:

* axios: Mentioned previously, in controller can send/receive data between frontend and backend.
* react-router-dom: Mentioned previously, handles navigation logic made by user actions.
* firebase: Used to trigger firebase authentication actions like login, signup, logout.
* react-redux: For managing the app’s state. Like user session data, and portfolio state.
* emailjs-com: Send email notifications to frontend if not handled by backend.
* octokit: An Github API client for connecting Github accounts and fetching repository metadata.

The model for SkillShow 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. Libraries and packages that are planned on being included in the model layer:

* firebase: Mentioned previously, main backend as a service for authentication, storage, database, and hosting.
* @firebase/firestore: Database for storing user data, profiles, and portfolios.
* @firebase/auth: Handles user authentication and account information.
* @firebase/storage: Stores media.
* @firebase/functions: Server sided email notifications or Github sync.
* octokit: Mentioned previously, Used in model to fetch and sync Github repository data.