# I-FAM Project – Sprint 2 Code Review

## A. Team

- Alice Ge
- Habone Said
- David Huang
- Yashvitha Thatigotla

## B. System/Project

- I-FAM Web App
    - Allows for event postings, RSVP, attendance tracking, and member management

## C. Subsystems

- Admin Dashboard
- Event Editing (Under Event Management)

## D. Context of Subsystems

### *Admin Dashboard*

- **General Context**: The Admin Dashboard is the first page that the admin sees after logging into the Admin Portal. It contains an “Upcoming Events” section where all the event cards are listed, and the admin is able to edit the event or record attendance for the event by clicking on different symbols attached to the event cards. The dashboard also contains a sidebar where the admin can access different features. Further details are included below.
- **Side Bar Navigation**: The side bar provides shortcuts to several key functionalities to the Admin Portal, including “Create Event”, “Manage Main Public Page”, etc. each tab is linked to the corresponding page, allowing the admin to quickly navigate the portal.
- **Event Management**: Upon clicking on the “edit” or “attendance” symbols on any event card, the admin will be redirected to the corresponding page for that

specific event. Further descriptions of the “edit” tab is included in the next section.

***Event Editing***

- **General Context:** An “editing” symbol is attached to each event card under the Admin Dashboard, and by clicking the symbol, the admin will be redirected to the Event Editing page where they can edit details of a published event (including date, time, description, and more). The page allows the admin to save their new edits and return to the Dashbaord. Further functionalities are included below.
- **Event Data Management**: Upon initialization, the component fetches existing event details from the Firestore database, populating form fields with data such as event name, location, details, dates, and flyers.
- **Dynamic Form Handling**: Utilizes Ant Design UI components for input fields, date and time pickers, and file uploads, offering a user-friendly interface for administrators to modify event details.
- **File Storage and Retrieval**: Integrates with Firebase Storage for handling event flyer uploads, enabling administrators to add new flyers and retrieve existing ones, facilitating promotional efforts for events.
- **State Management**: Employs React's useState and useEffect hooks for managing component state and side effects, ensuring the UI is responsive and data is up to date.
- **Submission and Update Process**: On form submission, the component updates the event details in the Firestore database, reflecting changes across the platform and ensuring users have access to the latest event information.

**E. Informal Specification**

***Admin Dashboard*** (including components)

- **Main Objective**: The primary goal is to create a centralized platform containing all needed links to direct the client to their desired features and pages.

- **User Stories and Features**:
  1. As an administrator, I want to be able to see and manage all upcoming events on a centralized dashboard to make easy changes to my events and keep track of the upcoming ones.
  2. As an administrator, I want to navigate to different parts of the portal in at most two clicks so that I can reach my desired page in a simple and efficient manner.
- **Key Components and Their Functions:**
  1. **SideBar.js:** This js file contains the source code to the side bar, which contains the navigation menu to different functionalities.
  2. **LayoutSider.js:** This is the layout file that wraps the side bar in a standardized layout, which allows different pages in the admin portal to adopt the same side bar without having to copy and paste the same code multiple times. Pages can easily adopt the side bar by wrapping the code within the imported layout function.
  3. **UpcomingEvents.js:** This file is a React component responsible for displaying upcoming events within an application. It fetches event data from a server, handles potential errors, and visually presents this information using a list of EventCard components. The component supports conditional rendering, offering additional administrative actions when it is used admin page
  4. **EventCard.js:** This file is designed to visually represent information about an individual event within a card. The component accepts event and actions as props. The event prop includes details such as the event's name, time, location, description, and associated flyers, while the actions prop allows for the inclusion of interactive elements such as buttons or links specific to each event.
  5. **Backend - api/event/route.js**: this file defines the GET API endpoint for event
  6. **service/back-end/event.js**: this file is dedicated to handling operations related to event data, specifically fetching all event records from a Firestore database. It utilizes Firebase Firestore and Storage services to retrieve event details and associated flyer images.

***Event Editing***

- **Main Objective**: The primary goal of this code is to establish a connection to Firebase services, which are essential for the backend infrastructure of the I-FAM project. This includes initializing the Firebase application with a specific configuration and providing access to Firestore and Firebase Storage services.

- **User Stories and Features**:
  1. As an administrator, I need to edit and update event details in real-time, ensuring that any changes to the event's time, date, location, or other pertinent details are immediately reflected and accessible to potential attendees. This capability is crucial for maintaining up-to-date information and ensuring the success of community events.
  2. As an administrator, I want to securely manage event flyers and promotional materials by uploading them to a reliable storage solution. This allows me to efficiently communicate event specifics, updates, and engaging content to students, thereby fostering greater participation and interest in upcoming events.
- **Key Components and Their Functions:**
  1. **Firebase Initialization (initializeApp):** Initializes the Firebase application using the provided configuration object. This configuration includes keys and identifiers that allow the application to connect to the specified Firebase project, enabling access to Firebase's cloud services.
  2. **Firestore Database Access (getFirestore)**: Provides access to Firestore, a NoSQL database for storing and syncing data. This service will be used for CRUD operations related to student information, event details, and other necessary data.
  3. **Firebase Storage Access (getStorage):** Grants access to Firebase Storage, a powerful and secure object storage solution for storing user-generated content such as images, documents, and event flyers. This service supports operations like uploading, listing, and retrieving files.
- **Data Structures:**

1. Firestore utilizes a document-oriented database model, where data is stored in documents (which can be thought of as records) and collections (which are akin to tables).
2. Firebase Storage organizes files in a hierarchical structure like a filesystem, with support for blob storage, making it suitable for handling binary data like images and documents.

**F. Changes**

***Admin Dashboard***

- **Adding in the remaining functionalities(Manage Main Page, Manage Past Events, Event Activities, etc.)**
- **Connecting to the backend**

***Edit Event***

- **Finish with the backend**