To create the project outlined in \*\*SWE-group7 Sprint3.pdf\*\* (Panther Hub), a campus event management platform, you will need to structure the development process into several key sections based on the project breakdown provided. Here's a suggested structure for the development:

### 1. \*\*Project Setup\*\*

- \*\*Technologies\*\*:

- Frontend: React.js

- Backend: Node.js with Express

- Database: MySQL

- APIs: Integration for map (e.g., Google Maps API) and notifications

- Version Control: GitHub (Link: https://github.com/Hloading/Panther-Hub)

- \*\*Setup\*\*:

1. Create a GitHub repository and initialize it with a `README` file.

2. Setup a React.js frontend and Node.js/Express backend.

3. Setup MySQL database and integrate the backend with it using `sequelize` ORM.

### 2. \*\*Core Components and Features\*\*

#### Frontend (React.js)

- \*\*Login/Registration\*\*:

- User login for students, faculty, and admins.

- Admin login for event organizers with privileges.

- \*\*Event Listings\*\*:

- Display events based on filters (date, category, location).

- Search and sort events functionality.

- \*\*Event Details Page\*\*:

- Show details like description, date, location (integrated with campus map), RSVP options.

- \*\*User Profile\*\*:

- Manage user data and view registered events.

- \*\*Event Creation (Admin Panel)\*\*:

- Admins can create, edit, cancel, and promote events.

- Event check-in using a QR code.

- Event feedback form post-event.

#### Backend (Node.js with Express)

- \*\*API Endpoints\*\*:

- `POST /login`: For user/admin authentication.

- `GET /events`: Fetch event listings with search and filter options.

- `POST /events`: Admins create new events.

- `PUT /events/:id`: Admins update events.

- `DELETE /events/:id`: Admins delete/cancel events.

- `POST /rsvp`: Users register for an event.

- `POST /feedback`: Users submit event feedback.

- \*\*Database Management\*\*:

- \*\*MySQL\*\*:

- User table (for login info)

- Event table (event details)

- RSVP table (stores user registration info)

- Feedback table (stores user event ratings and comments)

- Admin table (for event organizers)

### 3. \*\*System Architecture and Diagrams\*\*

- \*\*Class Diagram\*\*: Define classes like `User`, `Event`, `Admin`, `Feedback`, and their relationships.

- \*\*Context Diagram\*\*: High-level architecture showing users interacting with the system through a centralized platform.

- \*\*Activity Diagram\*\*: For user actions like event registration, feedback submission, and admin event creation.

### 4. \*\*Behavioral Modeling\*\*

- \*\*Use Cases\*\*:

- View/Search Events

- Register for Event

- Create/Manage/Cancel Events (Admin)

- Provide Feedback

- Event Check-in (with QR code)

- \*\*Sequence Diagrams\*\*: Show interactions between users, frontend, backend, and the database for each major feature.

### 5. \*\*Testing\*\*

- \*\*Test Cases\*\*:

- User login/logout.

- Event search and filtering.

- Event creation, editing, and cancellation.

- Event registration and feedback submission.

- \*\*Automation Testing\*\*: Use tools like Jest for unit tests in React and backend API tests.

### 6. \*\*Deployment\*\*

- \*\*Cloud Hosting\*\*: Use services like AWS, Azure, or Heroku for deployment.

- \*\*Database\*\*: Use cloud-based MySQL (AWS RDS, Google Cloud SQL).

- \*\*Frontend\*\*: Deploy React.js frontend on platforms like Netlify or Vercel.

### 7. \*\*Documentation and Maintenance\*\*

- Maintain detailed documentation in the GitHub repository for both developers and users.

Following this structure, you can start working on the different sections to develop the Panther Hub platform step by step. Let me know if you need help with a specific part.