Project Assignment: Event Management System

Build a **full-stack Event Management System** where users can manage event information through a user-friendly web interface and a RESTful backend API.

This assignment is designed to assess your frontend and backend development capabilities using modern technologies like **React.js**, **Next.js**, **Node.js (NestJS or Express)**, **Java with Spring Boot**, and **SQLite** as the database.

Tech Stack

Frontend (Required)	**Backend (Choose One - Required)**
• **React.js** (with or without Next.js) • Styling using **TailwindCSS**, **Material-UI**, or custom CSS • **State management** using Context API or Redux • **Routing** with React Router or Next.js	You may implement either of the following: • **Node.js** with **Express.js** or **NestJS** • **Java** with **Spring Boot**

Database - Use **SQLite** for simplicity and ease of local setup

Implementing <u>both backends, or parts of both, will be considered a strong plus</u>.

Features & Requirements

Implement the following core features:

1. View a list of events
2. Add a new event
3. Edit or delete an event
4. Search for events by name or date
5. *(Bonus)* RSVP to events
6.

Frontend (React.js + optional Next.js)

1. **Event List Page**
   - 1.1. Display events in a table or card layout
   - 1.2. Include a search bar for name/date filtering
2. **Add/Edit Event Form**
   - 2.1. Fields: Name, Description, Date, Location
   - 2.2. Perform client-side validation (required fields, date format)
3. **Delete Event**
   - 3.1. Include a delete button with confirmation prompt
4. **(Bonus) RSVP Feature**
   - 4.1. RSVP to an event and display RSVP status
5. **Technical Requirements**
   - 5.1. Use **React Hooks** (useState, useEffect, useReducer)

5.2. Apply **React Router** or **Next.js routing**
5.3. Use **Context API** or **Redux** for state management
5.4. Ensure responsive design using a UI library or CSS

Backend (Node.js or Spring Boot)
Create a RESTful API with the following endpoints:

Method	Endpoint	Description
GET	/events	Retrieves all events with pagination support.
POST	/events	Creates a new event.
PUT	/events/:id	Updates an existing event.
DELETE	/events/:id	Removes an event.
POST	/events/:id/rsvp (Bonus)	Registers a user's RSVP to an event.

Common Requirements (for both backends)

- Validate inputs using middleware/annotations
- Implement pagination (e.g., GET /events?page=1&limit=10)
- Handle errors gracefully with meaningful HTTP status codes
- Structure code using clean and modular architecture
- Integrate with SQLite using ORM (e.g., TypeORM, Sequelize, JPA, Hibernate)

(Bonus) Authentication
Implement JWT-based authentication:

- POST /auth/register – Register new users
- POST /auth/login – Login and return JWT
- Protect event modification routes (POST, PUT, DELETE) with JWT verification

Deliverables

1. GitHub repository with:
   1.1. Source code for both frontend and backend(s)
   1.2. SQLite schema or migration files
2. README with:
   2.1. Setup instructions
   2.2. API documentation (in Markdown or Swagger/OpenAPI format)
3. Easy-to-run project setup:
   3.1. Frontend: npm start or npm run dev
   3.2. Backend: npm start (Node.js) or gradle spring-boot:run (Spring Boot)

Skills Assessed

Frontend (React.js)

- Component design and state management
- Routing and form handling
- Responsive UI implementation
- API integration and error handling

Backend (One or Both)

- RESTful API development
- Input validation and error handling
- Authentication and authorization (JWT)
- Clean code structure and modularity

Database (SQLite)

- Schema design and migrations
- Efficient CRUD operations

General

- Problem-solving and attention to detail
- Code readability and maintainability
- Local environment setup and documentation

Bonus Points

- Implement **both** backend versions
- Add **sorting** (by date/name) to event listings
- Implement **server-side filtering/search**
- Use **React Query** or equivalent for efficient data fetching

Submission Guidelines

- Upload your project to **GitHub**
- Include the following:
    - Complete frontend and backend source code
    - Database schema or migration scripts
    - Detailed **README** with setup instructions
- Ensure the project can run locally with minimal effort