Project Assignment: Hotel Booking System

Build a full-stack Hotel Booking System where users can browse available hotel rooms and make bookings 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** with **Node.js (NestJS or Express)** OR **Java with Spring Boot**, and SQLite OR any database if you are comfortable with the database but **using RDBMS will get you bonus points**.

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**

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 available hotel rooms
2. Add a new hotel room (admin only)
3. Edit or delete a hotel room (admin only)
4. Search for hotel rooms by location or date
5. (Bonus) Book a room and view booking status

Frontend (React.js + optional Next.js)

1. **Hotel Room List Page**
   - 1.1. Display rooms in a grid or list format
   - 1.2. Include search/filter by location or availability dates
2. **Add/Edit Room Form**
   - 2.1. Fields: Hotel Name, Location, Room Type, Price/Night, Availability
   - 2.2. Perform client-side validation (required fields, number format, etc.)
3. **Delete Room**
4. Include a delete button with confirmation prompt (admin only)
5. **(Bonus) Booking Feature**
   - 5.1. Users can book a room and select check-in/check-out dates
   - 5.2. Display booking confirmation/status

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

Method	Endpoint	Description
GET	/rooms	List all rooms (with pagination & filters)
POST	/rooms	Add a new hotel room (admin only)
PUT	/rooms/:id	Update room details (admin only)
DELETE	/rooms/:id	Remove a room listing (admin only)
POST	/rooms/:id/book	Book a room (Bonus)
POST	/rooms/:id/cancel	Cancel a booking (Bonus)

Common Requirements (for both backends)

- Validate inputs using middleware/annotations
- Implement pagination (e.g., GET /rooms?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 room 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 rooms 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