Booking System Overview and Architecture

System Overview and Architecture: Booking System

1. Document Purpose

The purpose of this document is to provide a high-level overview of the architecture, design, and functionality of the booking system. This document describes the system's primary components, their functions, and the overall flow of data and control within the application.

2. System Summary

The booking system is a web application developed using the MERN stack (MongoDB, Express.js, React, and Node.js). It enables users to view, book, and manage room bookings and provides an administrative interface for managing rooms, bookings, and users. The system includes various features for security and access control, aiming to safeguard user data and ensure that only authorized users can perform sensitive operations.

3. High-Level Architecture

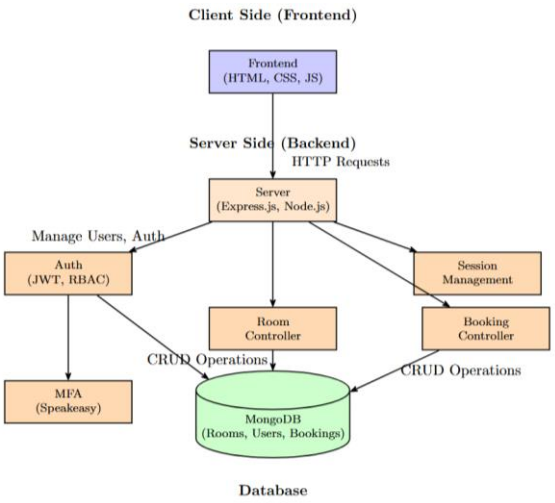

Figure 1: System Architecture Overview of the Booking System

Client Side (Frontend)

- **HTML, CSS, JavaScript**: Used for building the user interface, handling client-side routing, managing state, and interacting with the backend API.
- **Frontend Functionality**:
    - User and admin login, registration, and logout
    - Room viewing and filtering
    - Room booking and cancellation
    - Booking history display
    - Admin functionalities to manage rooms, view bookings, and manage promotional codes

Server Side (Backend)

- **Node.js and Express.js**: Used to develop the server and handle HTTP requests, define RESTful API endpoints, and enforce server-side business logic.
- **Mongoose**: ORM for MongoDB, used to define schemas and interact with the MongoDB database.

Database

- **MongoDB**: Stores data related to users, rooms, bookings, and sessions.
- **Collections**:
    - **Users**: Stores user information, hashed passwords, and roles (admin/user).
    - **Rooms**: Stores details for each room, including room number, location, capacity, price, amenities, and availability.
    - **Bookings**: Stores each booking record with references to the room and user, as well as the booking status and timeslot.
    - **Sessions**: (Optional) for managing sessions and tokens if sessions are stored in the database.

4. Core System Modules and Components

Each module is designed to handle specific responsibilities within the system, ensuring a separation of concerns and enhancing scalability and maintainability.

4.1 Authentication and Authorization

- **User Authentication**: Implemented using JWT (JSON Web Tokens) for session management, with tokens stored on the client side.
- **Roles and Permissions**:
    - **User**: Can view rooms, book available rooms, and manage their own bookings.
    - **Admin**: Has full control over rooms, bookings, and user management, including the ability to view all bookings and manage promotional codes.

4.2 Room Management

- **Room Creation and Editing**: Admin users can add new rooms with details like room number, location, capacity, price, and amenities. Images are uploaded and stored as binary data in MongoDB.
- **Room Availability Management**: Rooms have an availability flag that controls whether they can be booked by users.
- **Promotional Code Management**: Admins can assign promotional codes to rooms, which may grant discounts or other benefits.

4.3 Booking System

- **Booking Creation and Validation**: Users can book available rooms for specific dates and times. Before creating a booking, the system checks for room availability.
- **Booking Statuses**: Each booking has a status (e.g., booked, canceled, expired), which determines its visibility and access.
- **Booking Cancellation**: Users can cancel their bookings, changing the booking's status to "canceled." Admins can view and manage all bookings.

4.4 Security Components

- **Session Management**: JWT tokens are used to manage sessions and ensure secure access to endpoints.
- **Role-Based Access Control (RBAC)**: Middleware checks if users have admin privileges for certain actions.
- **Error and Input Validation**: Inputs are validated on the server side to prevent unauthorized access, SQL injection, and other common security threats.
- **Multi-Factor Authentication (MFA)**: Optional MFA setup for users for additional security.

4.5 Notification and Feedback

- **Success/Error Notifications**: Feedback provided on actions such as booking confirmation, room updates, and errors.
- **UI Elements**: Banners and pop-up messages are displayed on the frontend to indicate status updates for actions like booking and cancellation.

5. Data Flow

1. **User Authentication**:
    a. The client sends login credentials to the backend, where they are validated.
    b. If valid, a JWT token is generated and sent back to the client for future requests.
2. **Room Management**:
    a. Admins access the room management interface to add, edit, or delete rooms.
    b. Updates are sent to the backend via REST API calls, where room data is stored or updated in MongoDB.
3. **Booking Creation and Management**:
    a. Users select a room, date, and time slot for booking.
    b. The backend checks for room availability at the specified time, creates the booking if available, and updates the status.
    c. Bookings are displayed in the user's booking history and updated if canceled.
4. **Data Flow for Admins**:
    a. Admins have access to view and manage all rooms and bookings.
    b. Admin actions (e.g., adding rooms, updating booking statuses) propagate changes to the MongoDB collections and affect user-visible data.

6. Security Features and Access Controls

- **Authorization Middleware**: Verifies that users are authenticated and have the correct role (user or admin) for accessing each endpoint.
- **Validation Middleware**: Validates user input to prevent common vulnerabilities like SQL injection and XSS.
- **CORS Policy**: Configured to control the sources allowed to make requests to the server.
- **Error Handling Middleware**: Captures and logs errors while providing minimal information to end-users to avoid revealing sensitive information.

7. Technology Stack Summary

Component	Technology
Frontend Framework	HTML, CSS, JavaScript
Backend Framework	Node.js, Express
Database	MongoDB
Session Management	JWT
Authentication	Username/Password, MFA
Access Control	Role-Based Access Control (RBAC)
Storage	MongoDB (Rooms, Users, Bookings)

8. System Limitations and Assumptions

- **Local Hosting**: Currently designed as a locally hosted application. Scaling or transitioning to cloud hosting may require additional configuration.
- **Token Expiration**: JWT tokens have a 1-hour expiration for added security, requiring users to log in periodically.

9. Future Enhancements

- **Scalability Improvements**: Adapt the system for distributed databases and microservices architecture to improve performance.
- **Enhanced Reporting**: Add analytics features to generate reports on booking trends and room utilization.
- **Audit Logs**: Implement an audit trail for sensitive operations, such as room creation and user account modifications.

- End -