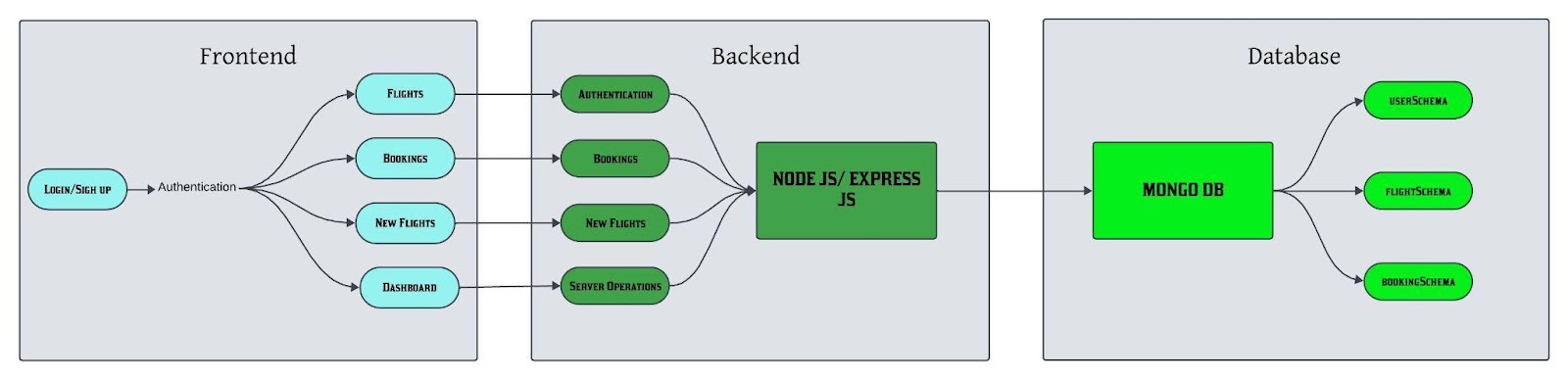
**Flight Booking App - Technical Architecture Report**

**1. Introduction**

The Flight Booking App is a robust and scalable platform designed to simplify flight ticket reservations using the MERN stack (MongoDB, Express.js, React.js, and Node.js). This report outlines the app's technical architecture, detailing its components, features, and design considerations to ensure efficiency, security, and seamless user experiences.

**2. Architectural Overview**

The architecture follows a modular and layered design pattern to ensure separation of concerns, scalability, and maintainability



**Frontend**

Technology: React.js

**Key Features :**

- Responsive and intuitive UI for both web and mobile views.

- Components for search, filtering, seat selection, and payment processing.

- Integration with APIs for real-time data display (e.g., flight availability and pricing).

- Client-side validation to enhance user experience.

**Backend**

- Technology: Node.js with Express.js

**Key Features:**

- RESTful API design for CRUD operations.

- Middleware for authentication, error handling, and logging.

- Business logic for search filters, booking management, and payment handling.

**Database**

Technology: MongoDB

**Key Features:**

- NoSQL database for efficient handling of dynamic schemas (e.g., flight schedules, user preferences).

- Collections:

- users (user profiles, preferences, and loyalty points)

- flights (flight details, pricing, availability)

- bookings (reservations, payment status, e-tickets)

**Payment Gateway Integration**

Third-party APIs: Stripe/PayPal for secure payment processing.

Features:

- Tokenization for secure transactions.

- Support for multiple currencies and payment methods.

**Authentication**

Technologies : JSON Web Tokens (JWT)

Features:

- Secure user authentication and authorization.

- Role-based access control for admins and users.

**Cloud Infrastructure**

Hosting Services:

- Frontend: AWS S3/Netlify for static content delivery.

- Backend: AWS EC2/Heroku for server deployment.

- Database: MongoDB Atlas for managed cloud database services.

CI/CD: GitHub Actions for automated testing and deployments.

Scalability Consideration

- Load balancers to handle high traffic.

- Horizontal scaling for backend servers and database shards.

- CDN (e.g., Cloudflare) for faster static content delivery.

**3. Data Flow**

1. User Interaction: Users interact with the React-based frontend to search and book flights.

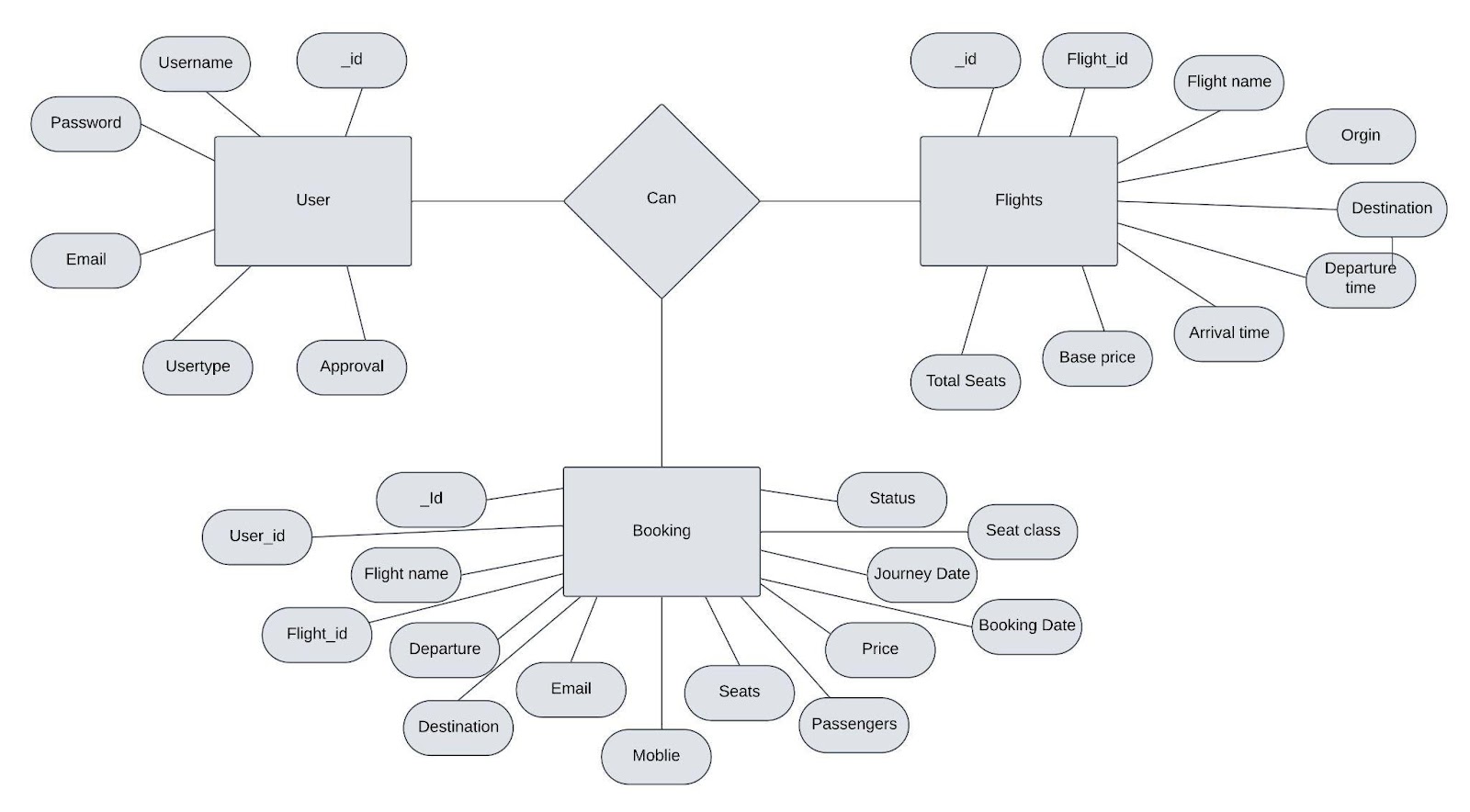
2. Request Handling: The frontend sends requests to the Node.js backend through RESTful APIs.

3. Data Processing: Backend processes the requests, applying business logic and interacting with MongoDB to fetch or update data.

4. Payment Gateway: Payment information is securely sent to the integrated payment API.

5. Response Delivery: The backend sends the response (e.g., booking confirmation, e-tickets) to the frontend for user display.

### ER DIAGRAM



The flight booking ER-diagram represents the entities and relationships involved in a flight booking system. It illustrates how users, bookings, flights, passengers, and payments are interconnected. Here is a breakdown of the entities and their relationships:

**User:** Represents the individuals or entities who book flights. A customer can place multiple bookings and make multiple payments.

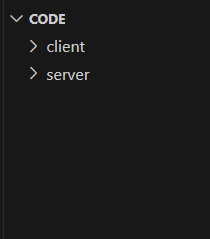
**Booking:** Represents a specific flight booking made by a customer. A booking includes a particular flight details and passenger information. A customer can have multiple bookings.

**Flight**: Represents a flight that is available for booking. Here, the details of flight will be provided and the users can book them as much as the available seats.

**Admin**: Admin is responsible for all the backend activities. Admin manages all the bookings, adds new flights,etc.

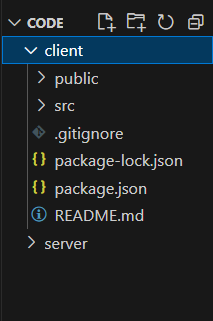
**4.Project Structure**

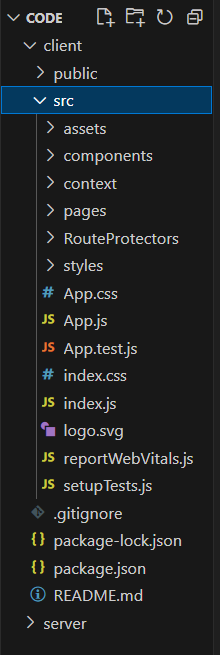
* Inside the Flight Booking app directory, we have the following folders



* **Client directory:**

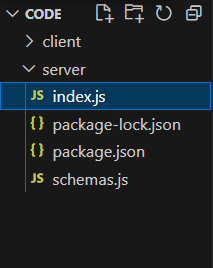
The below directory structure represents the directories and files in the client folder (front end) where, react js is used along with Api’s.





* **Server directory:**

The below directory structure represents the directories and files in the server folder (back end) where, node js, express js and mongodb are used along with Api.



**5. Tools and Libraries**

| Component | Tools/Libraries |

|-------------------------|---------------------------------------------------|

| Frontend | React.js, Redux, Bootstrap, Axios |

| Backend | Node.js, Express.js |

| Database | MongoDB, Mongoose |

| Authentication | JWT, bcrypt |

| Payment Gateway | Stripe, PayPal SDK |

| DevOps | GitHub Actions, Docker, Kubernetes |

**6. Challenges and Solutions**

| Challenges | Solutions |

|----------------------------------------------|------------------------------------------------------------------|

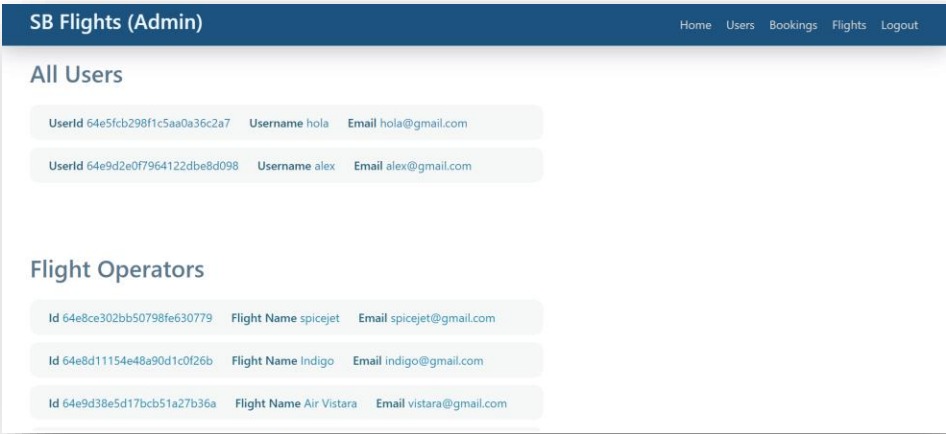
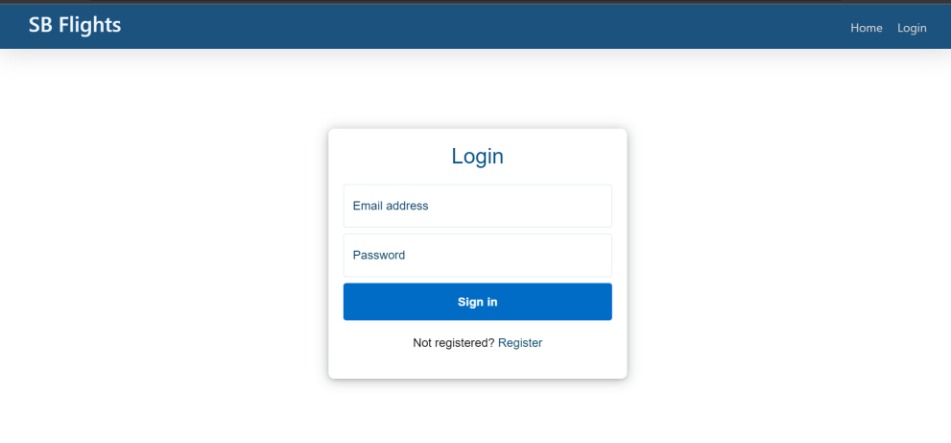
| Handling real-time flight data | Integration with third-party APIs (Amadeus, etc.) |

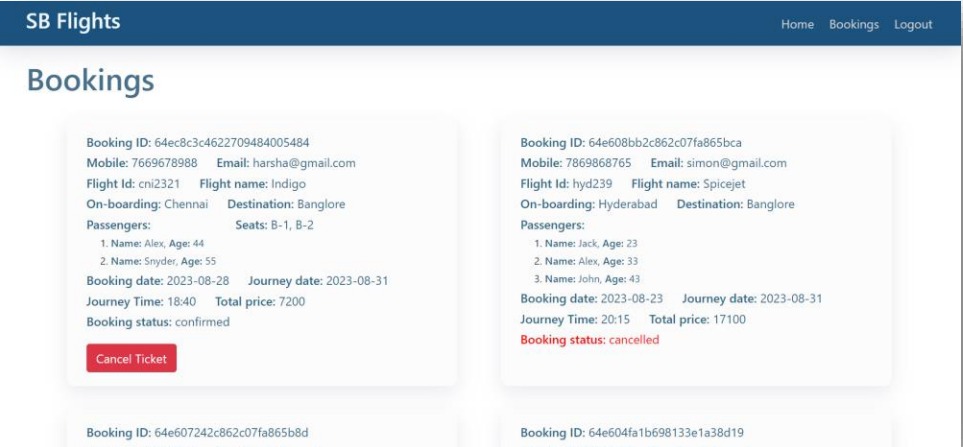
| Secure payment processing | Tokenized payment gateway and PCI compliance |

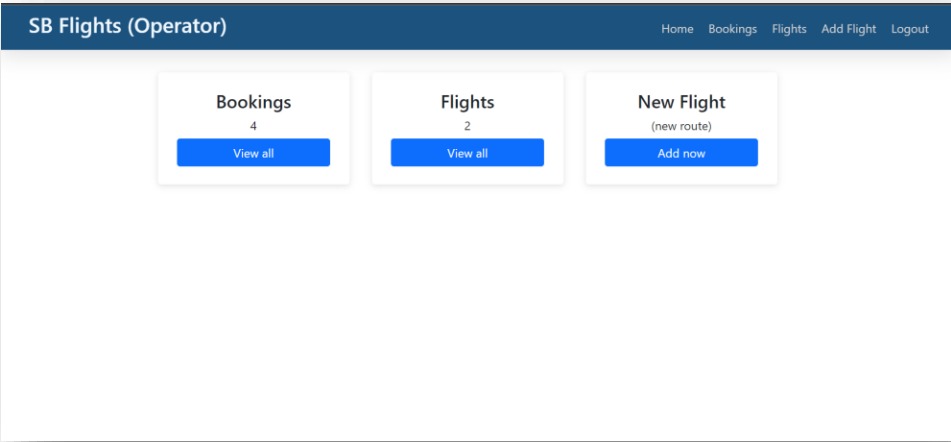
| Scalability to handle peak traffic | Cloud-based architecture with load balancers |

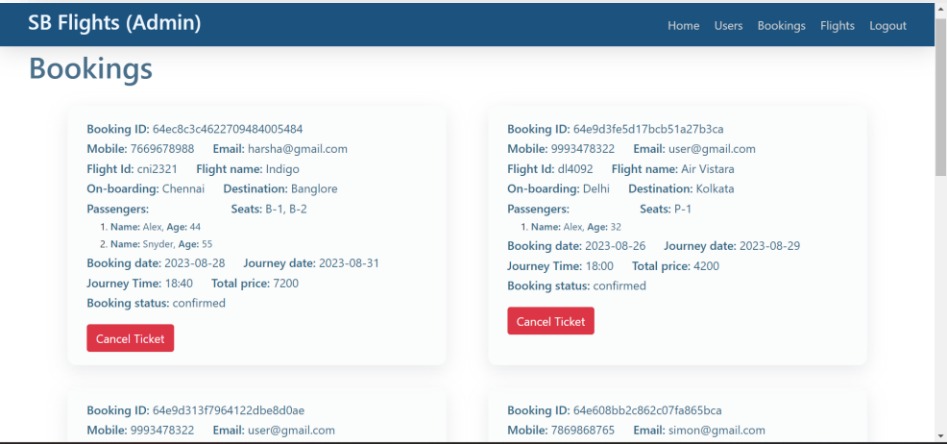
| Dynamic pricing updates | Scheduled jobs and API integrations |

**7.Output**



****

****



**8. Future Enhancements**

- AI-powered recommendations for flights based on user history.

- Multi-language support for a global audience.

- Integration with hotel and car rental services.

- Voice-activated search and booking.

**8. Conclusion**

The Flight Booking App leverages the MERN stack to provide a robust, user-friendly, and secure platform for flight reservations. Its modular architecture, combined with real-time features and scalability, ensures an optimal user experience while meeting industry standards for performance and security. This app is designed to continuously adapt to user needs, making travel planning effortless and enjoyable.