

# FULL STACK DEVELOPMENT WITH MERN PROJECT DOCUMENTATION

## 1. Introduction

**Project Title** : Flight finder -Navigating your Air Travel Option

**Team id** : LTVIP2025TMID56096

**Team Members:**

1. **Team Leader:**  
Nukala Lahari – Full Stack Developer & Project Coordinator  
Responsible for overall planning, coordination, GitHub management, and integration of frontend and backend.
2. **Team Member:**  
Nalliboina Manikanta – Frontend Developer  
Works on the React-based UI, handles component design, page routing, and user interactions.
3. **Team Member:**  
Nallamolu Naga Divya – Backend Developer  
Builds RESTful APIs using Node.js and Express.js, manages authentication and server logic.
4. **Team Member:**  
Naragam Karthik – Database Administrator  
Designs and manages MongoDB schemas, handles CRUD operations and ensures data consistency.

## 2. Project Overview

- **Purpose:**  
To simplify flight booking and management by providing users with real-time availability, secure bookings, and user-friendly search features.
- **Features:**
  - Search and filter flights
  - Book tickets with user login
  - Admin panel for managing flights
  - Email confirmation and booking history

### **3. Architecture**

- **Frontend:**  
Built with React.js using components, hooks, and React Router for navigation. UI powered by Material-UI.
- **Backend:**  
Developed using Node.js and Express.js. RESTful API handles all CRUD operations related to users, flights, and bookings.
- **Database:**  
MongoDB stores user data, flight details, and booking records. Mongoose is used for schema design and data validation.

#### ➤ **Frontend (React.js)**

##### **Components:**

- SearchForm.js: Handles user inputs (source, destination, dates).
- FlightList.js: Displays ranked results from ML model.

**State Management:** Redux for shared state (user sessions, bookings).

#### ➤ **Backend (Node.js/Express.js)**

##### **API Routes:**

javascript

```
app.get('/api/flights', flightController .search);
```

```
app.post('/api/bookings', authMiddleware , bookingController.create);
```

#### ➤ **Middleware:** JWT validation, rate limiting.

##### **Database (MongoDB)**

##### **Schemas:**

javascript

```
const User = new Schema({  
  email: { type: String, unique: true },  
  bookings: [{ type: Schema.Types.ObjectId, ref: 'Booking' }]  
});
```

### **4. Setup Instructions**

#### **Prerequisites:**

- Node.js >= 18
- MongoDB installed and running
- npm or yarn

### Installation:

- git clone https://github.com/your-username/FlightFinder.git
- cd FlightFinder
- cd server
- npm install
- cd ../client
- npm install

### Environment Variables:

Create a .env file in the /server folder with:

PORT=3000

MONGODB URI-

mongodb+srv://nehapriya:<db\_password>@cluster0.zghkpqk.mongodb.net/?retryWrites=true&w=majority&appName=Cluster0

## 5. Folder Structure

- **Client:**

/client

```

├── /src
|
|   ├── /components
|   ├── /pages
|   ├── /api
|   ├── App.js
|   └── index.js

```

- **Server:**

/server

```

├── /controllers
├── /routes
├── /models
├── server.js
└── .env

```

## **5. Running the Application**

### **Frontend:**

- cd client
- npm start

### **Backend:**

- cd server
- npm start

## **7. API Documentation**

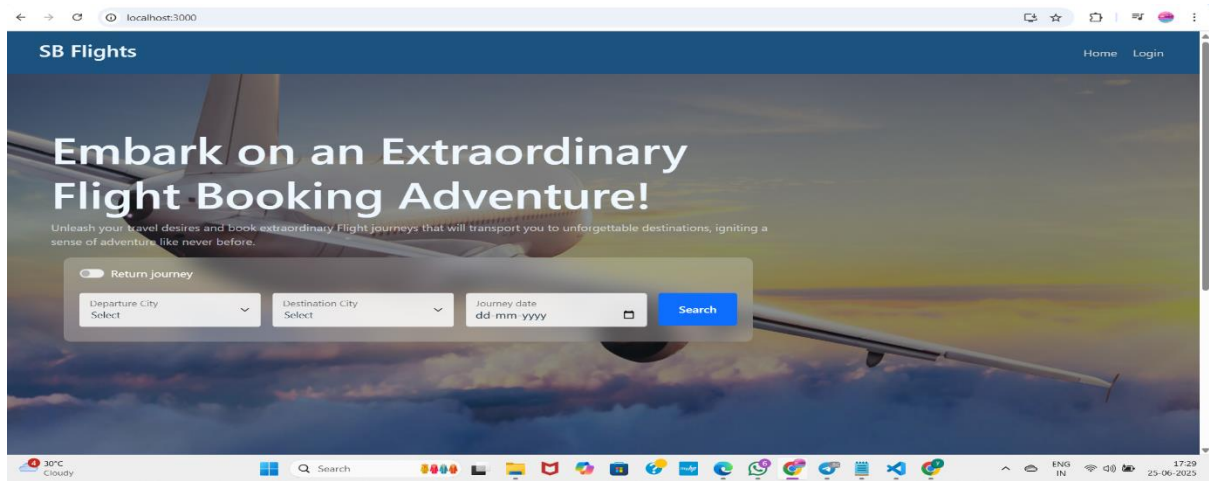
- **POST /api/auth/register**  
Registers a new user.  
Request Body: { name, email, password }  
Response: { success, token }
- **POST /api/auth/login**  
Logs in an existing user.
- **GET /api/flights**  
Retrieves list of available flights.
- **POST /api/bookings**  
Books a flight for the logged-in user.

## **8. Authentication**

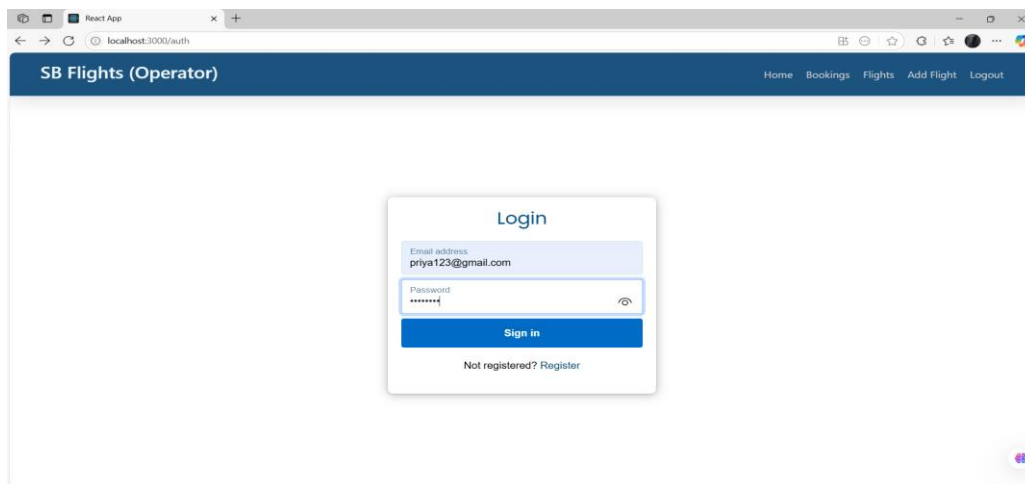
- Implemented using JWT (JSON Web Token).
- Tokens are stored in HTTP-only cookies.
- Protected routes validate tokens before granting access.

## **9. User Interface**

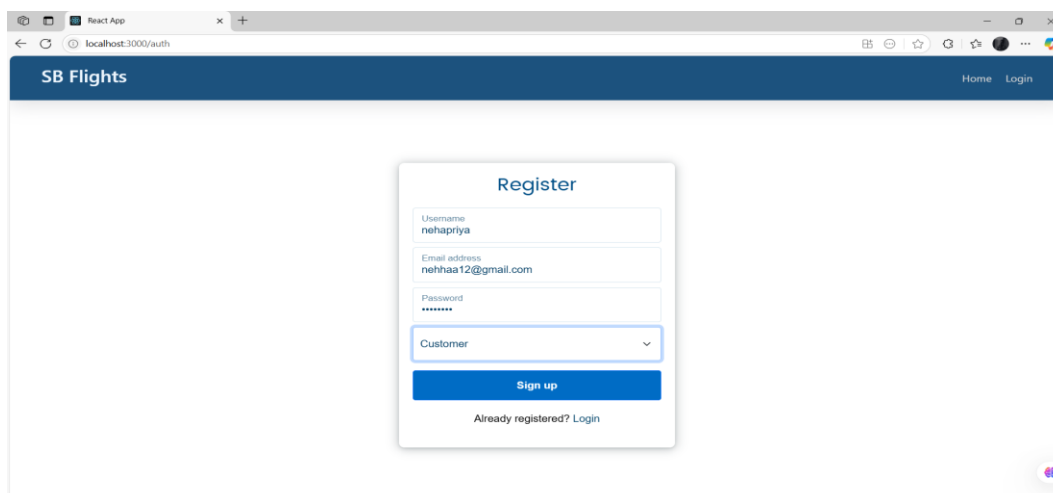
Home Page with flight search



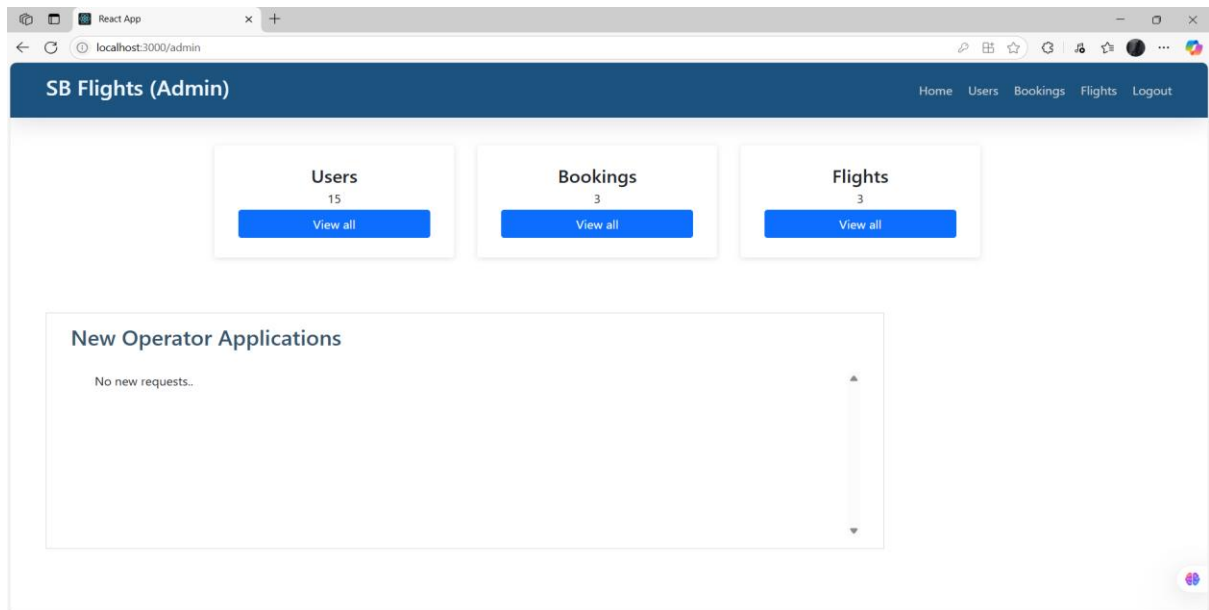
## Login form



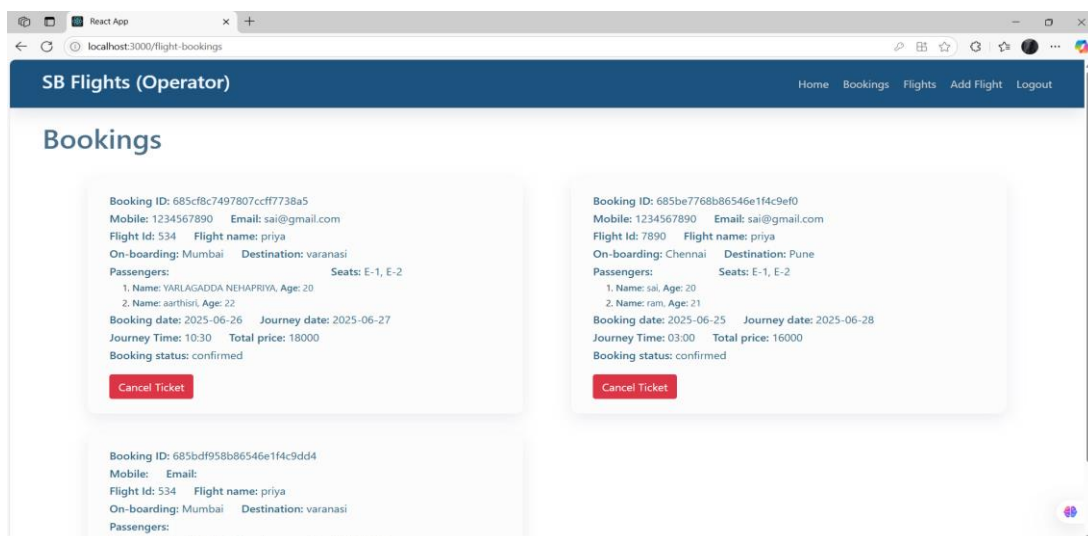
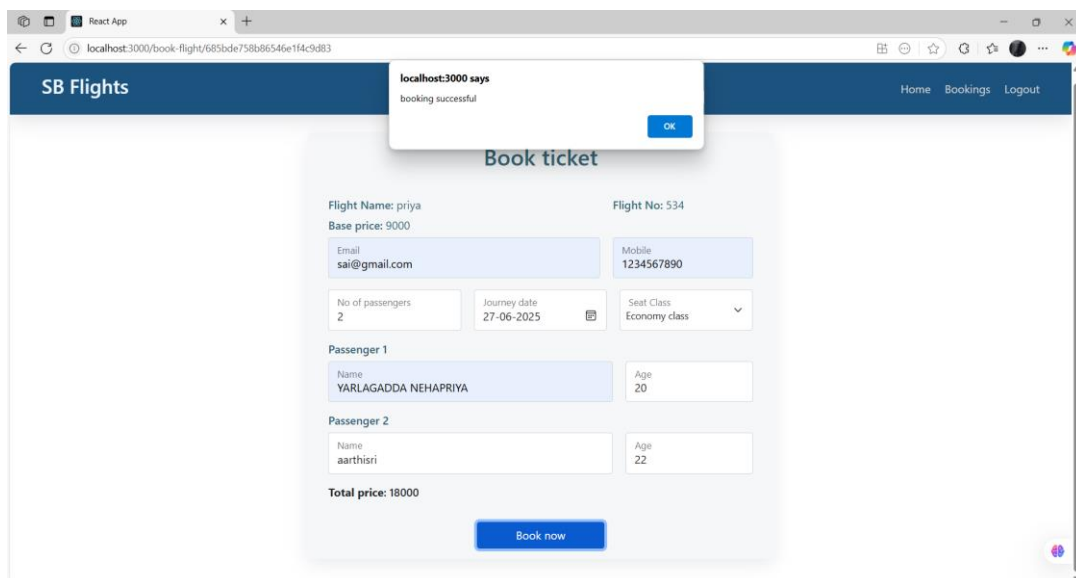
## Register form



## Admin dashboard



## Booking confirmation page



## **10. Testing**

- Manual testing on Postman for backend APIs
- Basic component tests in React using Jest and React Testing Library

## **11. Known Issues**

- No password recovery implemented yet
- UI responsiveness needs improvement on tablets

## **12. Future Enhancements**

- Payment integration with Razorpay/Stripe
- Role-based admin access
- Real-time flight tracking integration