Full Stack Development with MERN

FlightFinder: Navigating Your Air Travel Options

1. Introduction

• Project Title: FlightFinder: Navigating Your Air Travel Options

• Team Members:

- Davuluri Dinesh (Team Leader, Full Stack Developer)

- Cherukuri Prakash (Frontend Developer)

- Challa Siva Ganesh (Backend Developer)

- Chilukoti Kevalya Deepthi (UI/UX Designer)

2. Project Overview

• Purpose: To create a streamlined, efficient, and user-friendly platform for searching, filtering, and booking flights with real-time seat selection and secure payment options.

• Features:

- User Authentication

- Flight Search and Filters

- Seat Selection Interface

- Payment Integration

- Booking Confirmation and E-Ticket Generation

3. Architecture

• Frontend: Built using React.js and Tailwind CSS. It provides a responsive UI for searching flights, viewing results, and booking tickets.

• Backend: Node.js with Express.js RESTful API endpoints handle user authentication, flight search, bookings, and admin management.

• Database: MongoDB stores users, flights, and booking information with structured schemas using Mongoose.

4. Setup Instructions

• Prerequisites: Node.js, MongoDB, npm, Git

• Installation:

1. Clone the repository: git clone https://github.com/DineshDavuluri/FlightFinder

2. Navigate to /client and run npm install

3. Navigate to /server and run npm install

4. Create a .env file in /server and add environment variables for MongoDB URI and JWT Secret

5. Start MongoDB service

5. Folder Structure

• Client:

- /components (UI Components)
- /pages (Route-based Pages)
- /services (API Calls)

• Server:

- /routes (API Routes)
- /controllers (Business Logic)
- /models (Mongoose Schemas)
- /middleware (Auth Middleware)

6. Running the Application

• Frontend:

```
cd client
npm start
```

• Backend:

```
cd server
npm start
```

7. API Documentation

• POST /api/auth/register - User Registration

• POST /api/auth/login - User Login

• GET /api/flights - Fetch Available Flights

• POST /api/bookings - Book a Flight

• GET /api/users/:id/bookings - Get User Bookings

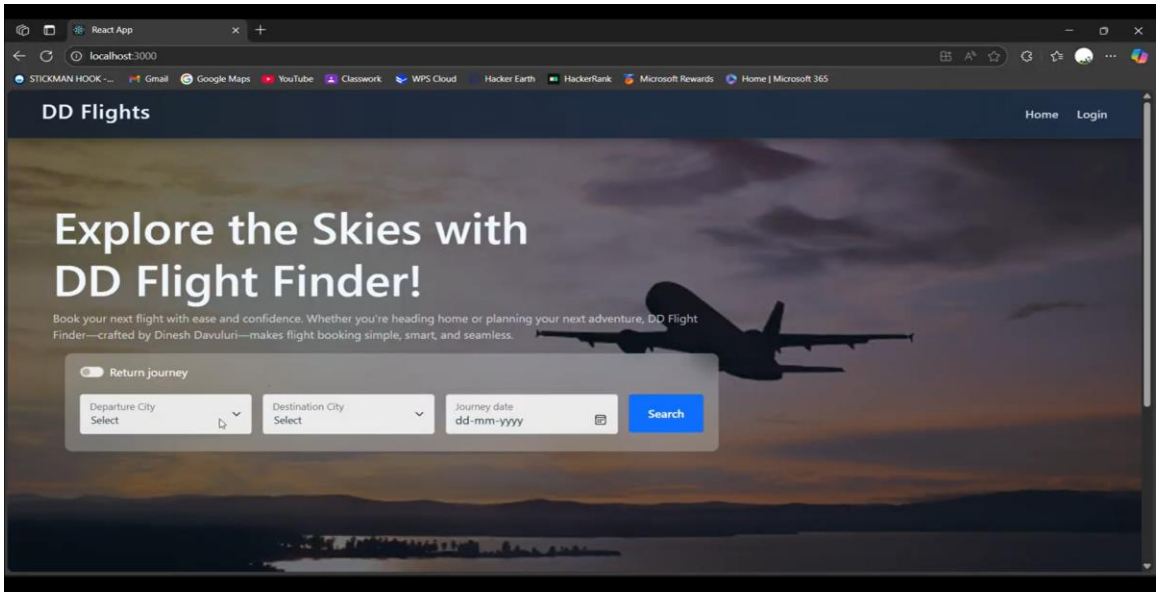
8. Authentication

• Authentication is handled via JSON Web Tokens (JWT). Upon login, users receive a token stored in local storage.

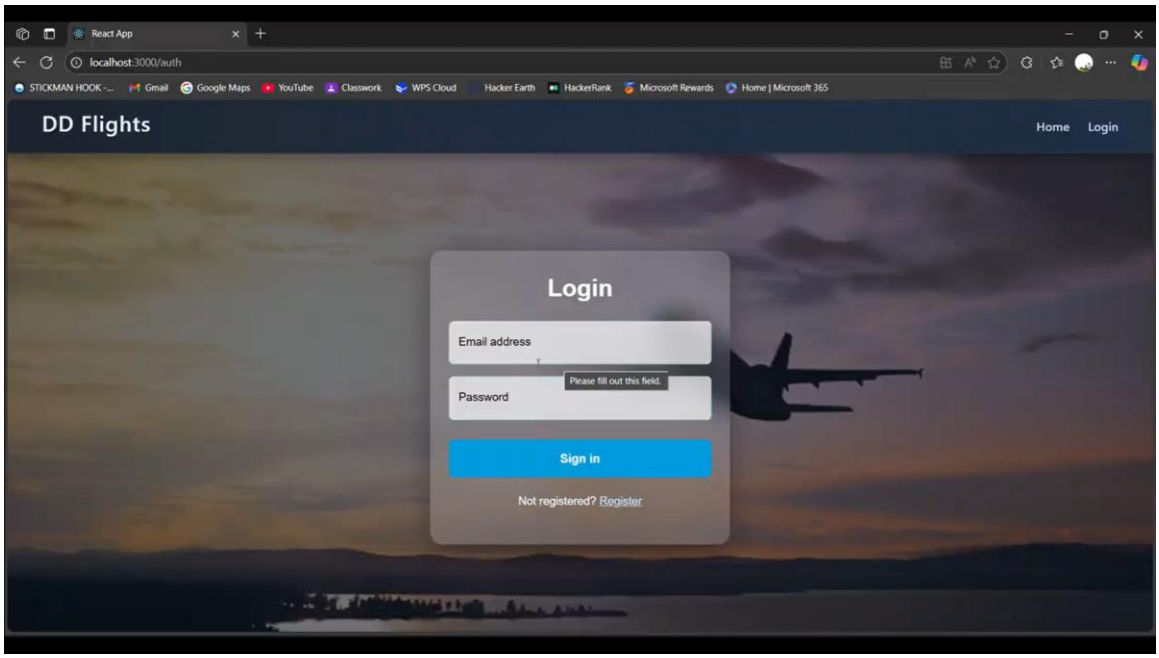
• Authorization middleware validates the token for protected routes.

9. User Interface

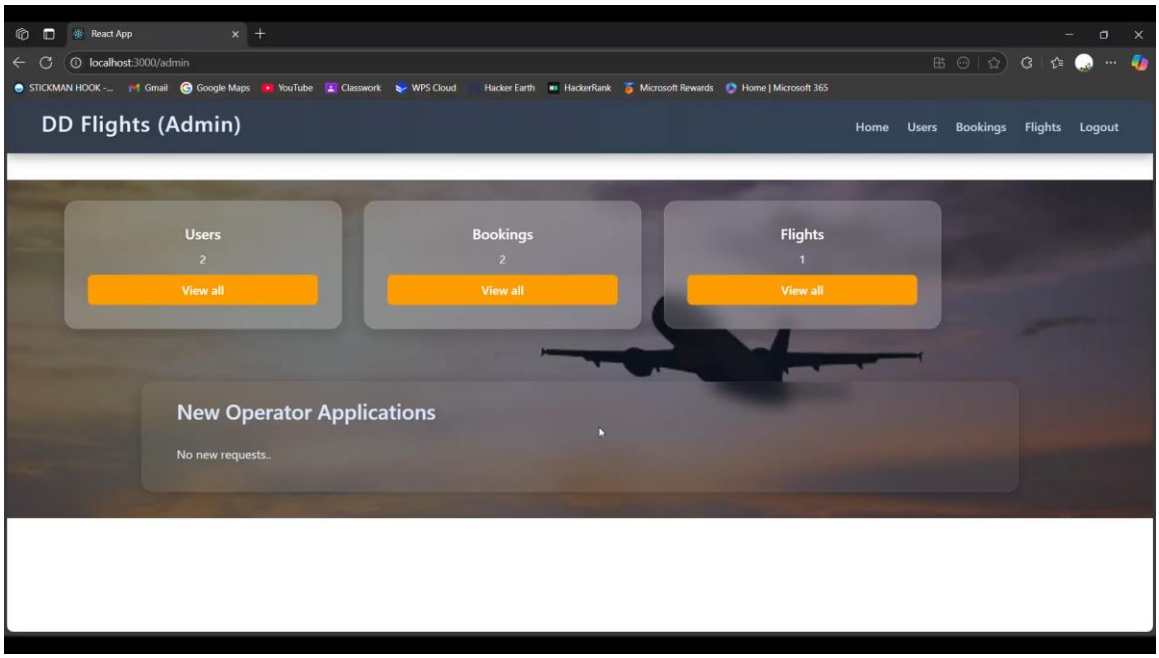
• Homepage with search form



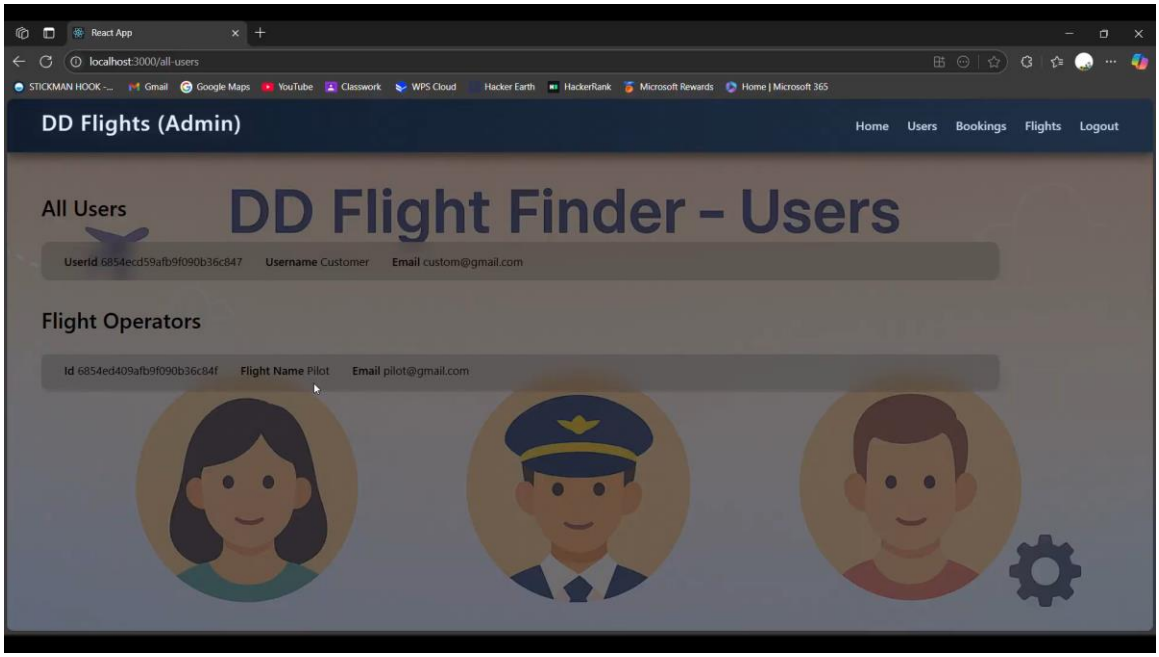
• Login Page results



• Admin interface



• Admin Users view



10. Testing

• Manual Testing: Performed across all modules.

• Tools: Lighthouse for performance, Postman for API testing.

• Performance Score: 91/100 (Google Lighthouse)

11. Screenshots or Demo

• Demo Video: https://drive.google.com/file/d/1RzGIXwahdWaGrKTjTyRuRIvMlB8w8mSH/view?usp=drivesdk

• GitHub: https://github.com/DineshDavuluri/FlightFinder

12. Known Issues

• No real-time flight data integration (mock data used).

• No email notifications post booking.

13. Future Enhancements

• Integration with real-world flight APIs

• Email and SMS confirmations

• Mobile app version

• Loyalty program integration