Full Stack Development with MERN

Project Documentation

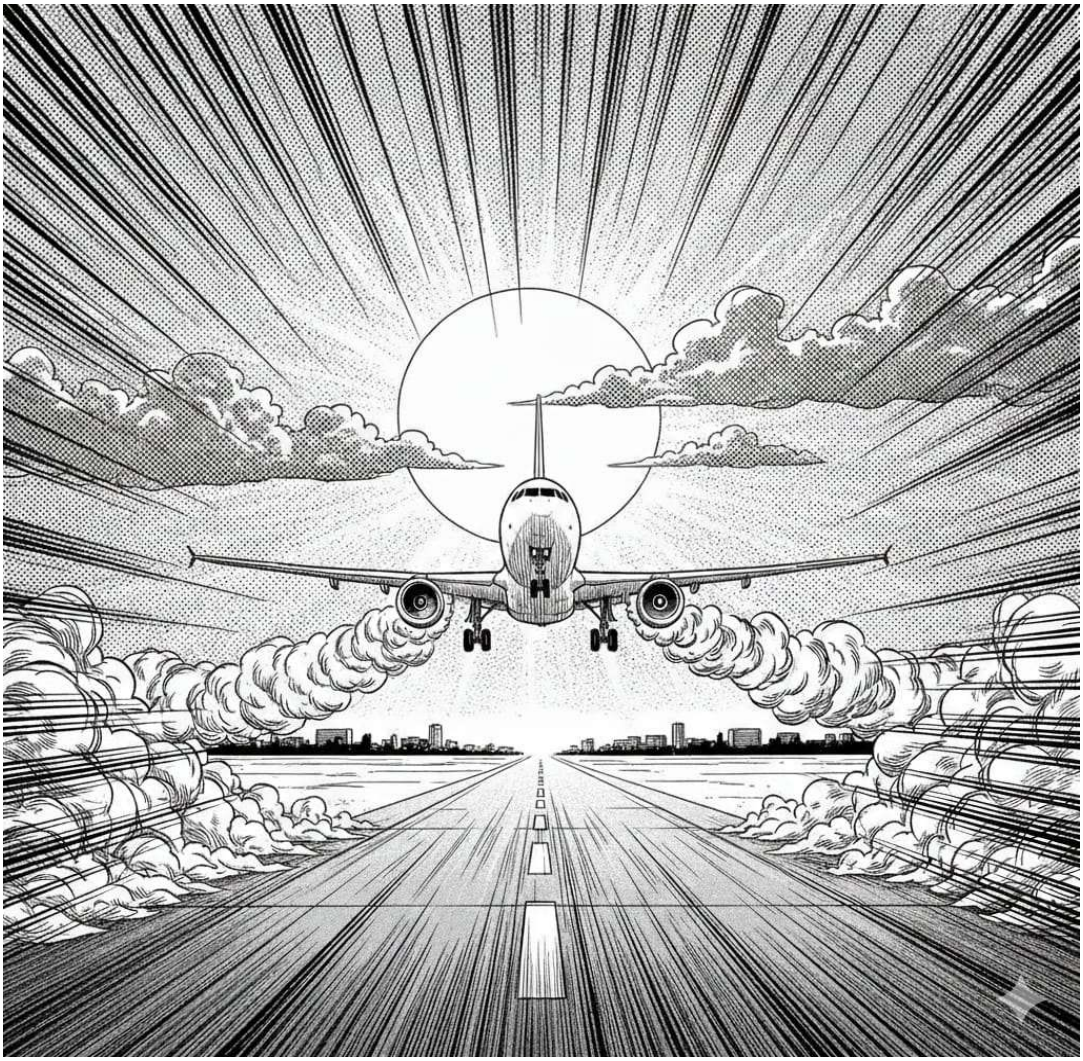
Project Title: FlightFinder - Online Flight Booking System

Team Leader: Sodimella Lavanya

Team Member: Gowtham Maharshi Penumarthi

Team Member: Gopi Pradeep Kumar Y

Team Member: Jalla Padmasri



1. Introduction

FlightFinder is a web-based Flight Booking System developed using the MERN stack (MongoDB, Express.js, React.js, Node.js). The application provides a seamless and secure platform for users to search, compare, book, and manage flights efficiently. It ensures data security through JWT authentication and role-based access control.

2. Project Overview

FlightFinder allows users to register, log in, search flights based on departure, destination, travel date, and class type. Users can book flights, view booking history, and cancel reservations if needed. The system also includes an admin dashboard to manage flight listings, monitor bookings, and track revenue.

3. Setup Instructions

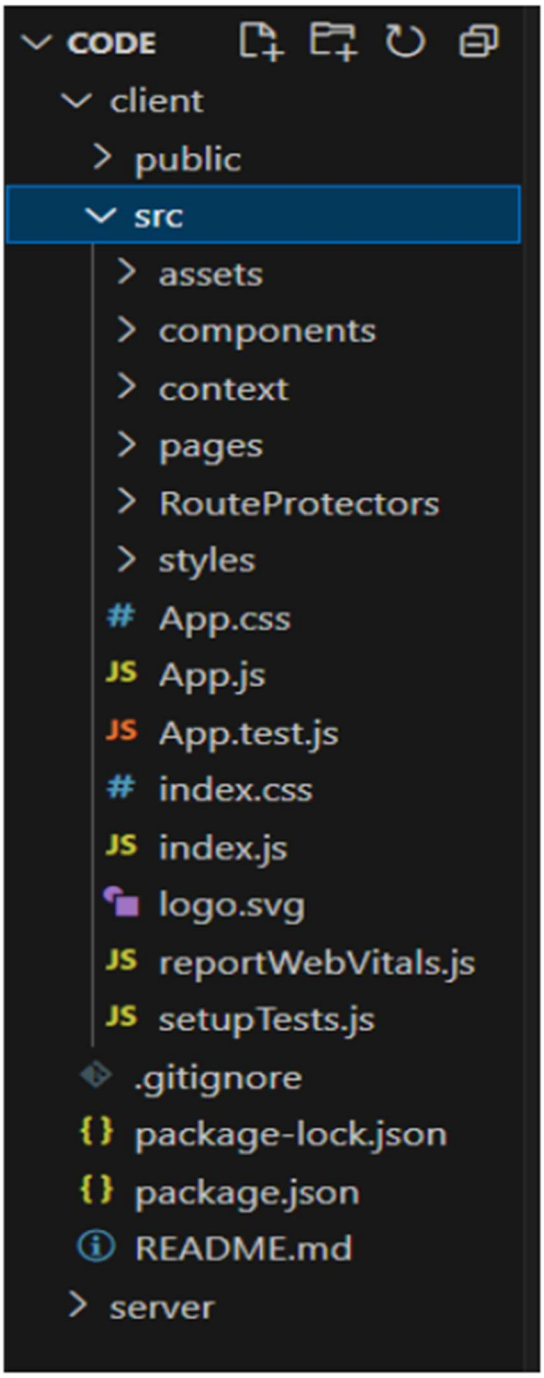
1. Clone the repository from GitHub.
2. Navigate to the project directory.
3. Install dependencies using npm install.
4. Configure environment variables (MONGO_URI, JWT_SECRET).
5. Start backend using npm run dev.
6. Start frontend using npm start.
7. Open http://localhost:3000 in browser.

4. Folder Structure

Client Folder: Contains src, components, pages, services, App.js, and routing files.

Server Folder: Contains models, routes, controllers, middleware, config, and server.js.

The below is the folder structure



5. API Documentation

- POST /api/register - Register user
- POST /api/login - Login user
- GET /api/flights - Get all flights

- POST /api/flights - Add flight (Admin)
- PUT /api/flights/:id - Update flight (Admin)
- DELETE /api/flights/:id - Delete flight (Admin)
- POST /api/bookings - Create booking
- GET /api/bookings - Get user bookings
- PUT /api/bookings/:id - Cancel booking

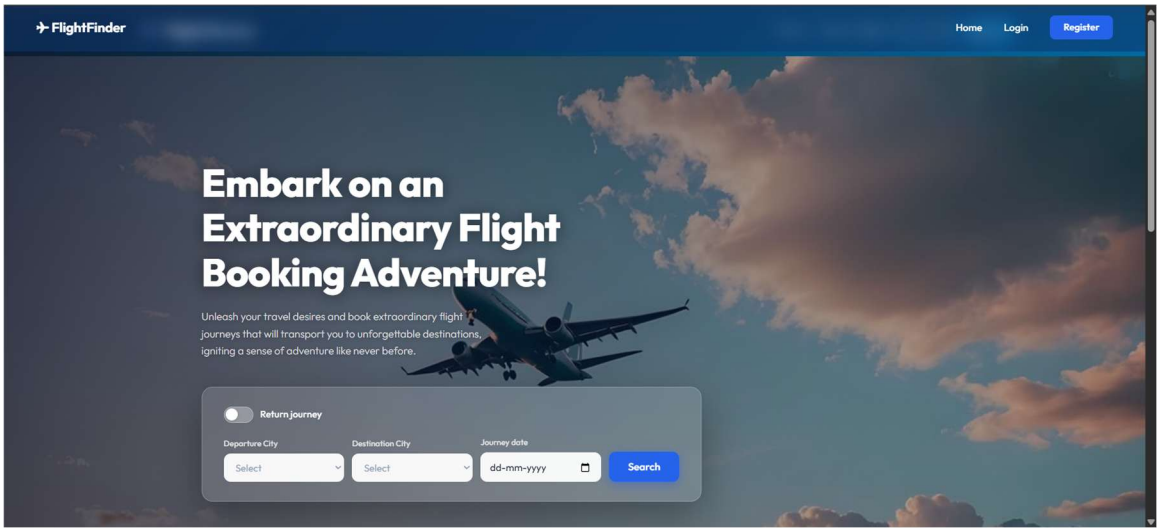
6. Authentication

Authentication is implemented using JWT tokens. User passwords are encrypted using bcrypt before storing in MongoDB. Role-based middleware ensures that only authorized users can access protected routes.

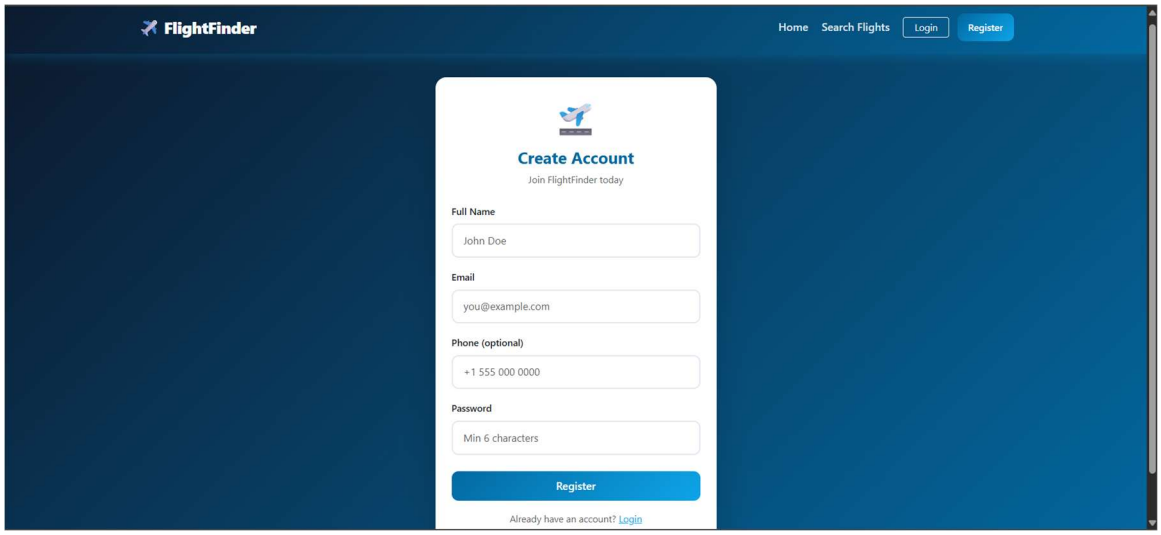
7. Testing

API testing was performed using Postman. Functional testing included registration, login, flight search, booking creation, cancellation, and admin flight management. Performance testing was conducted to verify response time and database efficiency.

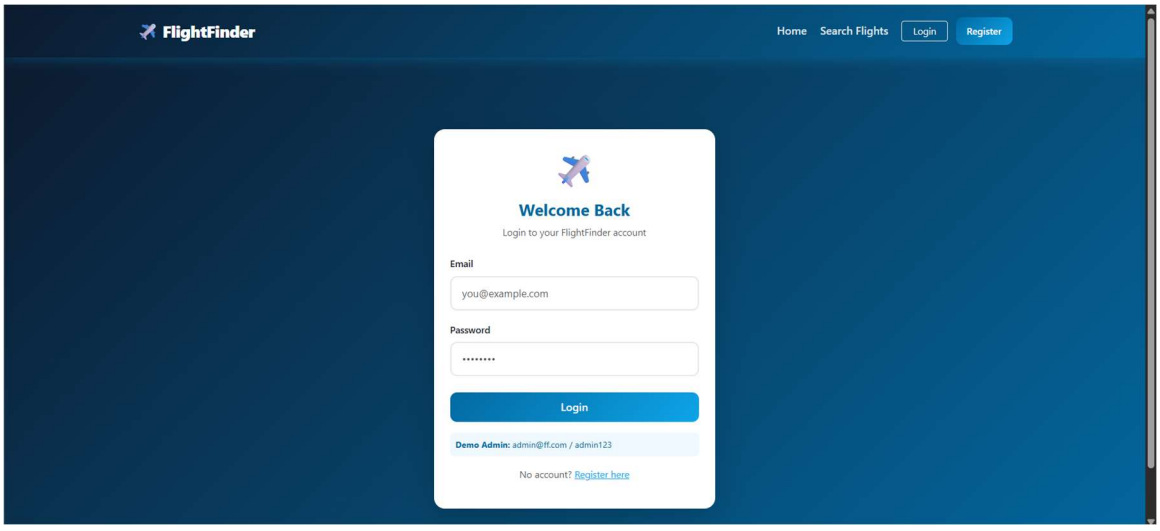
8. Screenshots / Demo



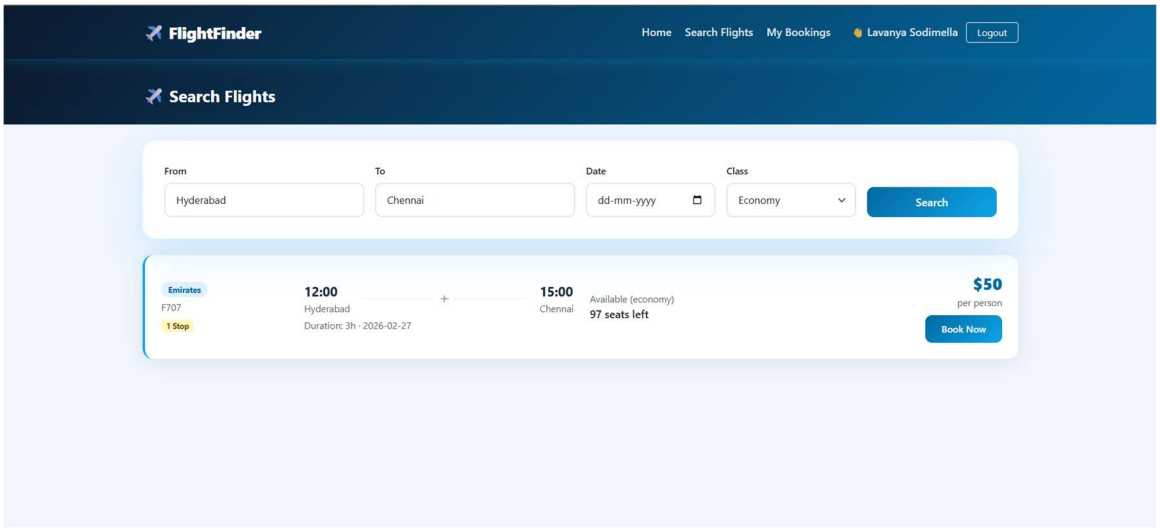
1.Home Page



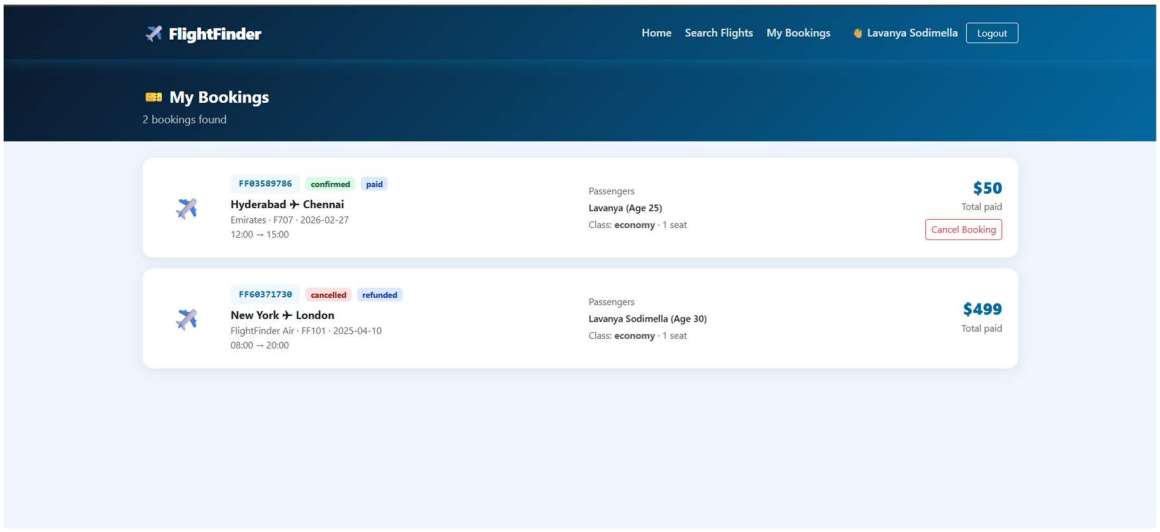
2.Registeration Page



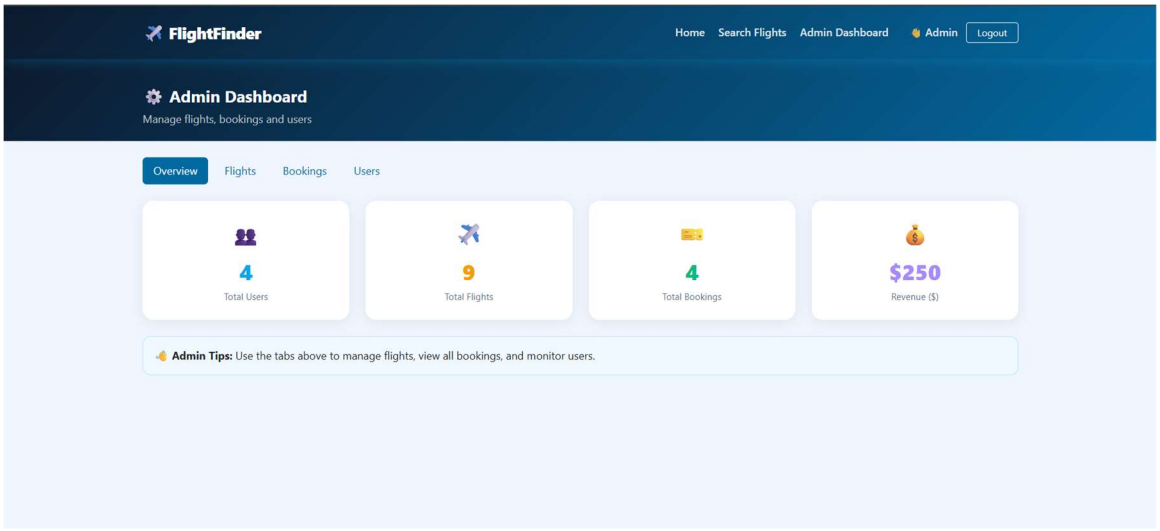
3.Login Page



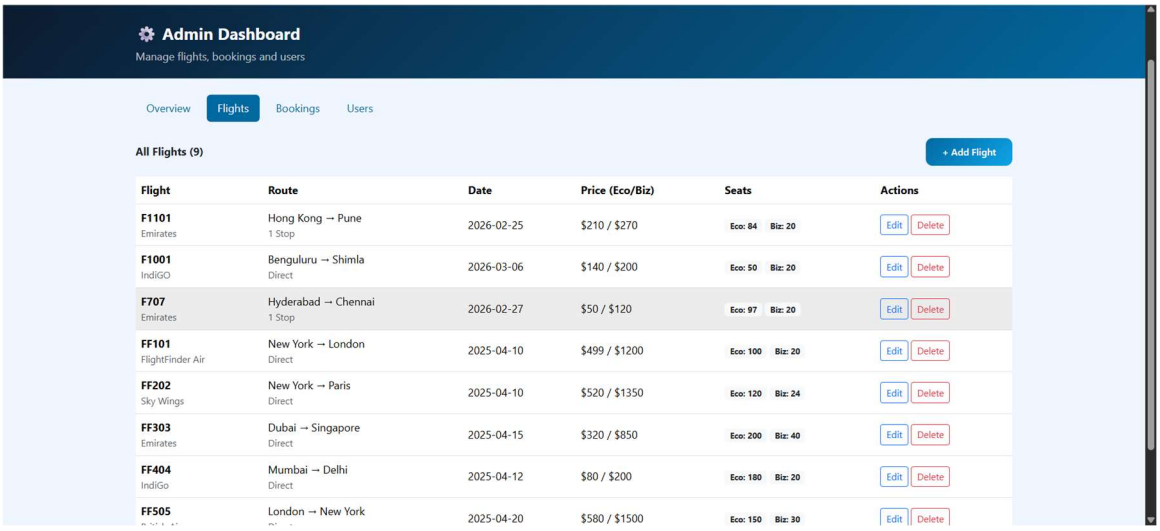
4.Search Flights



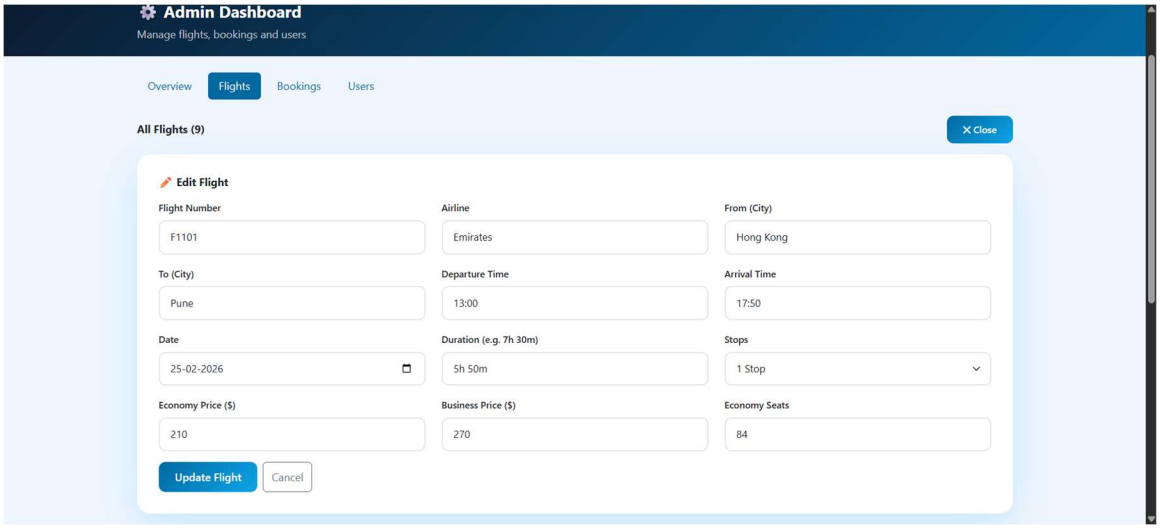
5.My Bookings



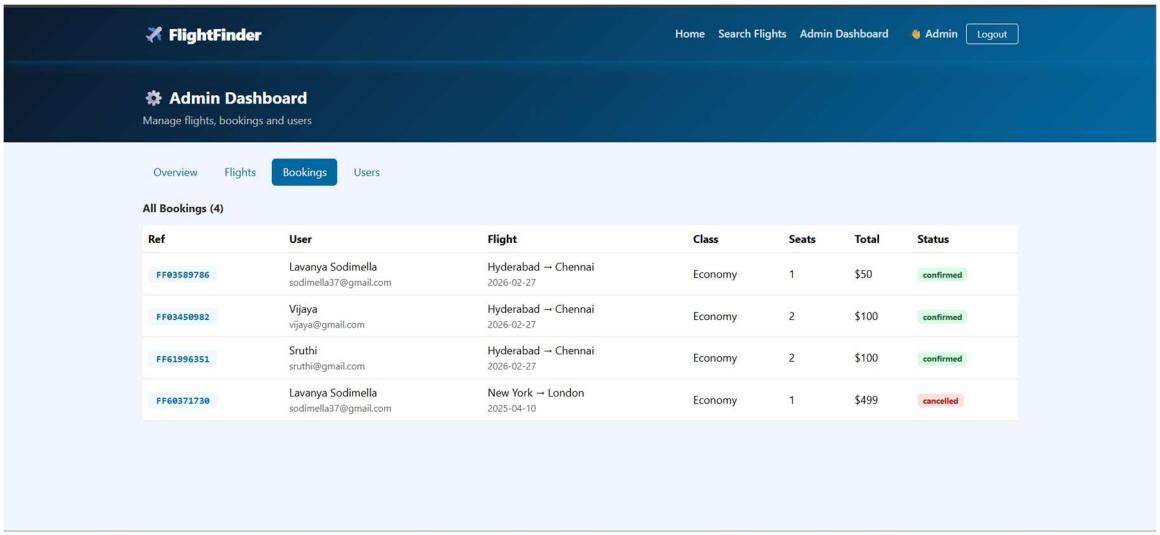
6.Admin Dashboard



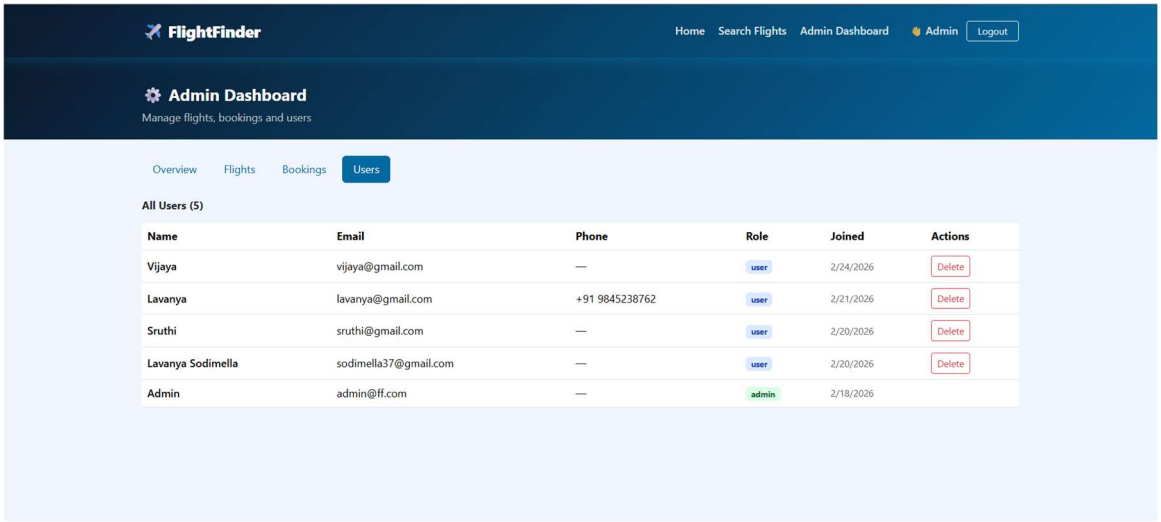
7.Admin Managing Flights



8.Admin Adding Flights



9.Admin Managing Bookings



10.Admin Managing Users

9. Known Issues

No real payment gateway integration.

Limited advanced filtering options.

UI responsiveness can be improved for smaller screens.

10. Future Enhancements

- Real payment gateway integration (Razorpay/Stripe).
- Live airline API integration.
- Seat selection interface.
- Email notifications for booking confirmation.
- Mobile application development.
- AI-based flight recommendations.