MERN stack powered by MongoDB

Naan Mudhalvan - Project Documentation

Introduction

Project Title: Myflight (Flight Booking Application)

College: Tagore Engineering College – 4127

Department: B.E CSE IV Year 7th Sem

Team ID: NM2024TMID02021

Team Members:

S.No	Student Name	Register No.	Naan Mudhalvan ID	Role
1.	Tom Roger Tarun B	412721104054	F24F5FA1FA8008395CC4B11F28DEF2DE	Backend
2.	Abinesh S	412721104002	1353FB1BC99F6E489D548B2B947F1EE0	Frontend
3.	Raakesh K	412721104038	3D5E15839888B05B66B4B7D63FCDB637	Frontend
4.	Sudhakar S	412721104050	250D3FA1BD70AC2AAFB477830E8F3A9E	Frontend
5.	Vasantharaja V	412721104057	E35921B27FA96DCBA144E29F2D3AF766	Backend

Project Overview

Purpose:

This Flight Booking Application is designed to provide users with an intuitive platform to search, book, and manage flight reservations efficiently. It eliminates the use of JWT for authentication, relying on session-based authentication for secure user sessions.

Features:

The application includes the following features:

- **- User Authentication:** Session-based login and signup with secure password handling.
- **- Flight Search:** Search for flights by destination, date, and price range.
- **- Booking Management:** View, confirm, and manage flight bookings.
- **- Payment Integration:** Seamless and secure payment processing using a payment gateway.
- **- Admin Dashboard:** Manage flights, user accounts, and monitor bookings.
- **- Responsive Design:** Ensures compatibility across devices.

Architecture

The application leverages the MERN stack for efficient development and deployment:

- **- Frontend:** Developed using React.js, styled with Bootstrap for responsiveness.
- **- Backend:** Implemented with Node.js and Express.js for APIs and session handling.
- **- Database:** MongoDB for storing flight details, user data, and bookings.
- **- Middleware:** Uses bcrypt for password hashing and session-based authentication for security.

Setup Instructions

Prerequisites:

- Node.js
- MongoDB Atlas Account
- Cloudinary account for image and video storage.

Installation:

1. **Clone the repository:**

   git clone https://github.com/tomroger1823/Flight-Booking-App-MERN

   cd online-learning-platform

2. **Install dependencies:**
   - **Frontend:**

     cd client

     npm install
   - **Backend:**

     cd server

     npm install
3. **Set up environment variables:**
   - Create **.env** file in the **server** folder with the following:

MONGODB_URI='mongodb+srv://tomroger1823:flight%40123@cluster0.cwt4c.mongodb.net/?retryWrites=true&w=majority&appName=Cluster0'

JWT_SECRET='t9843yt8hg0h8y834th893hy89h'

EMAIL_USER='tomroger1823@gmail.com'

EMAIL_PASS='szlv xili ndrw qasg'

CLOUDINARY_CLOUD_NAME='dgosdgcem'

CLOUDINARY_API_KEY='695861773226449'

CLOUDINARY_API_SECRET='H7QShPl9ouKwmBTdrmnFihgaIIU'

- Create .**env.local** file in the **client** folder with the following:

VITE_REACT_APP_BACKEND_BASEURL=http://localhost:6001

Folder Structure

Client (Frontend):

- **src/:** Contains all React components, pages, and assets.
  - **Components/:** Reusable UI components such as Footer, Header, HeaderStudent and SideBar
  - **Pages/:** Folders such as Admin, User, Instructor and Navs containing Pages for different aspects of the application.

Server (Backend):

- **routes/:** Defines API routes for users, instructors, forums, and admin.
- **models/:** MongoDB schemas for Users, Courses, flights and booking.
- **middleware/:** Uses bcrypt for password hashing and session-based authentication for security.
- **utils/:** Contains the Cloudinary API.

Running the Application

- **Frontend:**

  cd client

  npm run dev

- **Backend:**

  cd server

  nodemon server.js (or) node server.js

API Documentation

The following API endpoints are available:

User Routes

· **- POST /signup:** Register a new user.

- **- POST /login:** Login user with session-based authentication.
- **- POST /logout:** End user session.
- **- GET /profile:** Fetch user profile details.

Flight Routes

- **- GET /flights:** Retrieve all available flights.
- **- POST /book:** Book a flight.
- **- GET /bookings:** View user booking history.
- **- DELETE /bookings/:id:** Cancel a booking.

Admin Routes

- **- POST /add-flight:** Add a new flight.
- **- PUT /edit-flight/:id:** Edit flight details.
- **- DELETE /remove-flight/:id:** Remove a flight.
- **- GET /all-bookings:** View all bookings.

Passenger Routes

- `POST /signup`: Register a new passenger.
- `POST /login`: Login user.
- `GET /verify-email`: To verify the newly registered passenger's email.
- `POST /book-flight`: Book a flight.
- `GET /bookings/:email`: Fetch all bookings for a specific passenger.
- `DELETE /bookings/:id`: Cancel a specific booking.
- `GET /booking-details/:id`: Get booking details by ID.

User Interface

1. **Customer Dashboard:**
   - Displays available flights, booking history, and profile management.
   - Search and filter options for flights based on destinations, dates, and price range.

2. **Admin Dashboard:**
   - **Manage flights:** Add, update, or remove flight details (e.g., schedules, fares, seats).
   - **User management:** View customer profiles, bookings, and manage refunds.
   - **Analytics:** View insights like most popular routes, revenue reports, etc.

3. **Airline-Operator Dashboard:**
    - Monitor booked flights and passengers.
    - Seat assignment and updates on flight status (e.g., delays or cancellations).

Testing

1. **Manual Testing:**
    - Verified functionalities such as:
        - User signup/login and profile updates.
        - Flight search and filtering options.
        - Booking confirmation and payment flow.
        - Viewing booking history and handling cancellations.
2. **Postman:** Used for API endpoint testing.

API Endpoint Testing:

User Authentication: Tested endpoints for login, signup, and validation.

Flight Operations:

`GET /flights`: Fetch available flights.

`POST /bookings`: Confirm bookings.

`PUT /flights/:id`: Update flight details.

Admin Operations:

`POST /flights`: Add new flights.

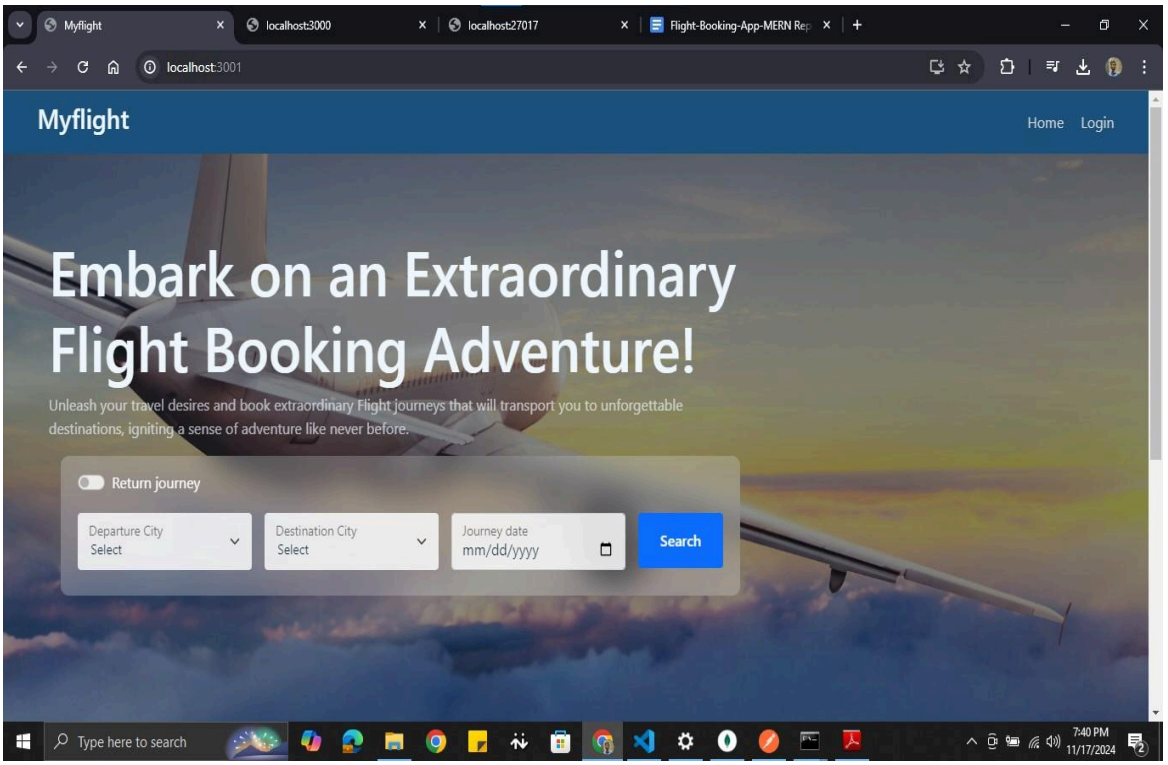
`DELETE /flights/:id`: Remove flights.

Screenshots or Demo

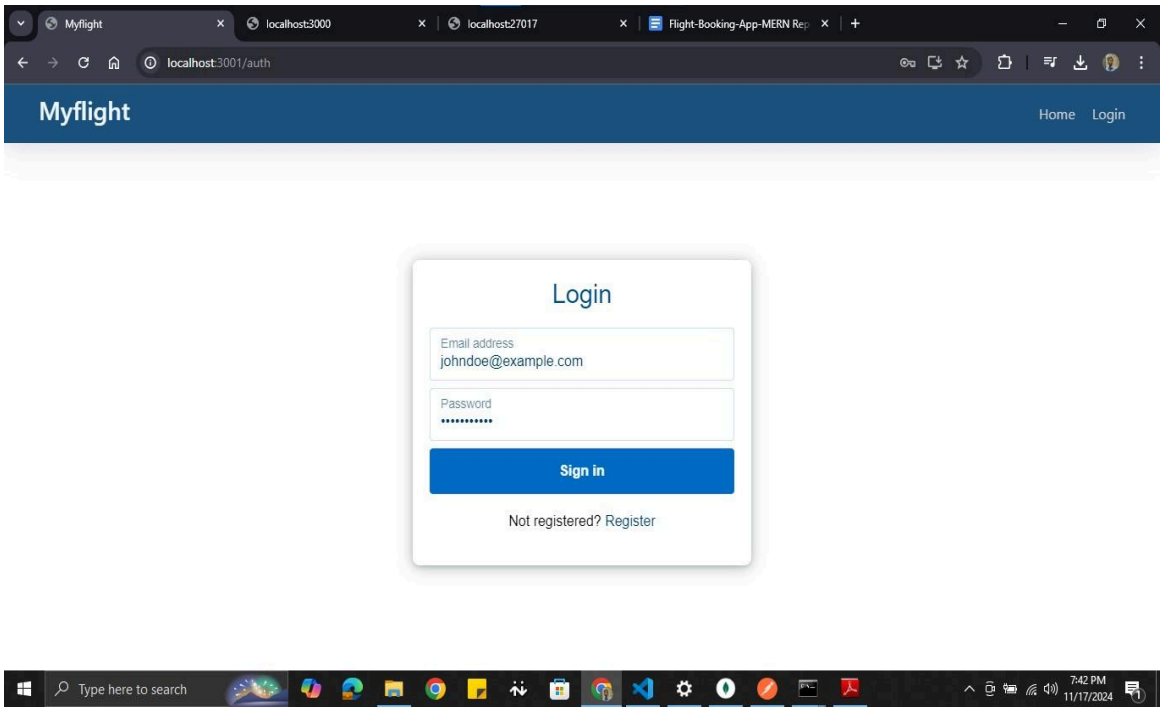
- **Demo Link:**

Screenshots:

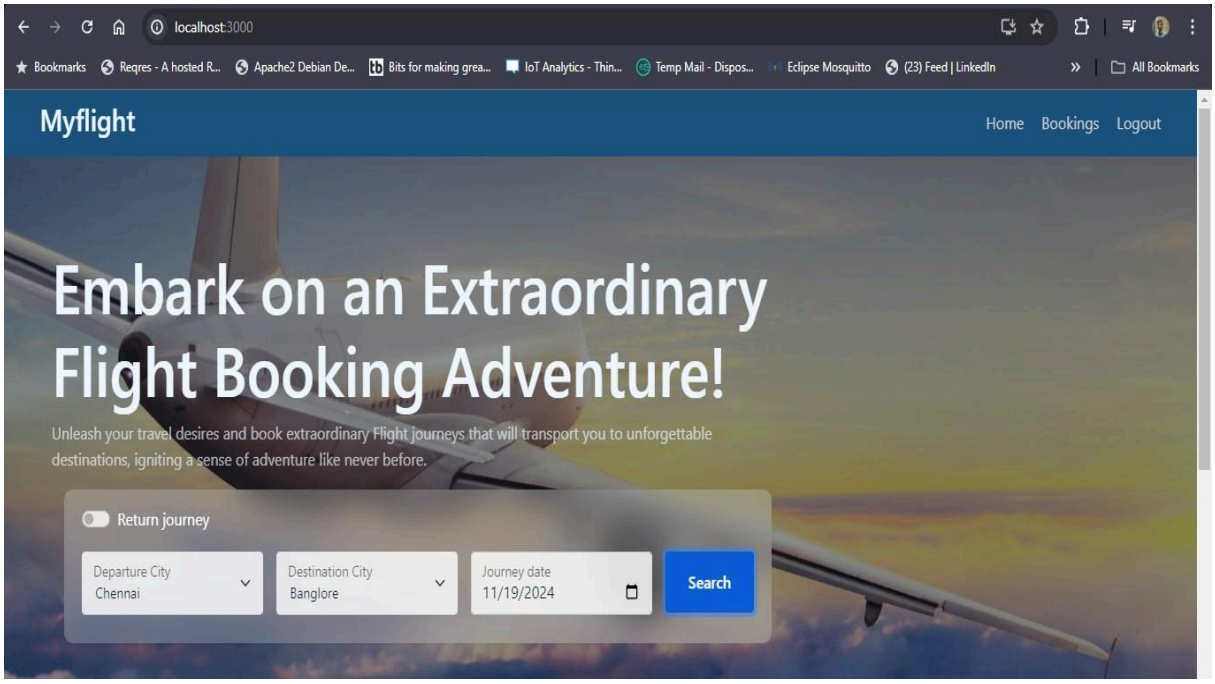
1. Landing Page



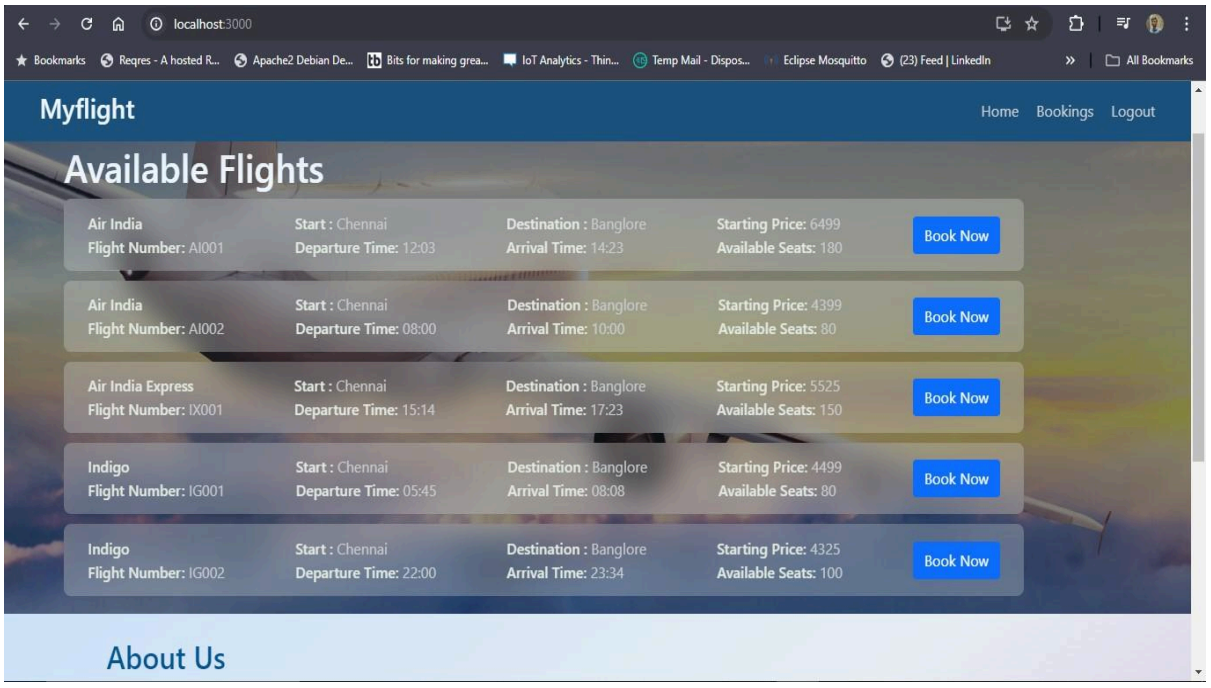
2. User Login



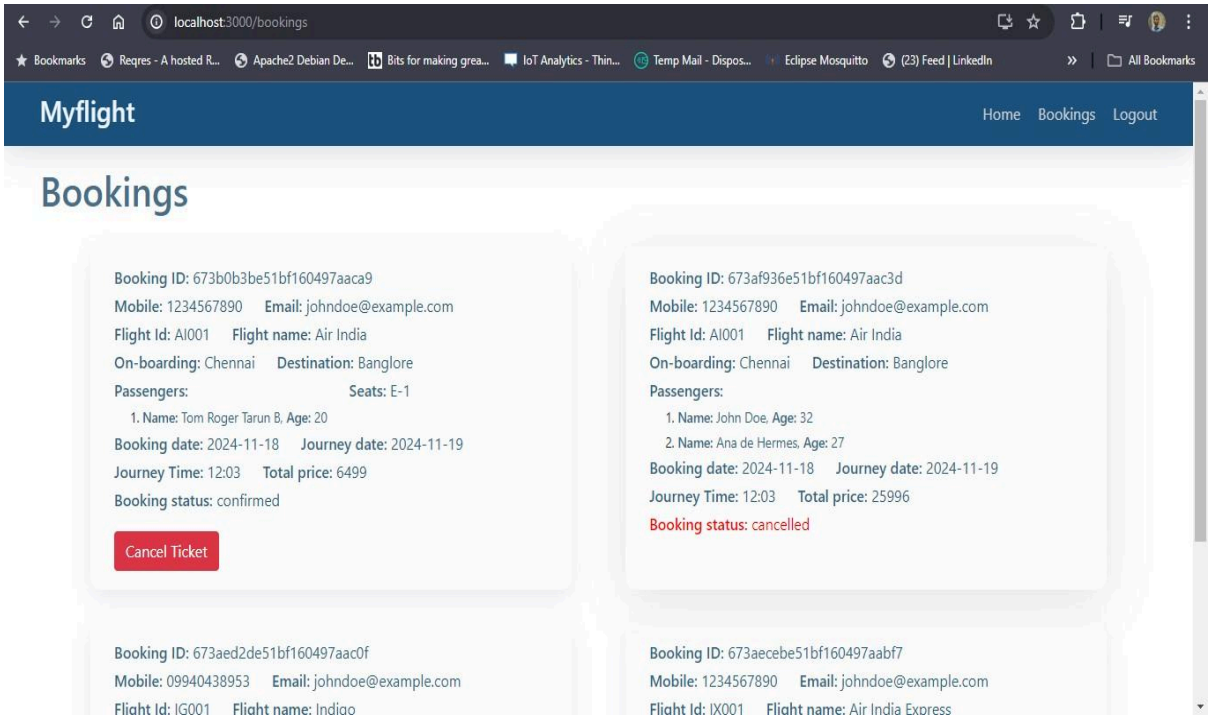
3. User Dashboard



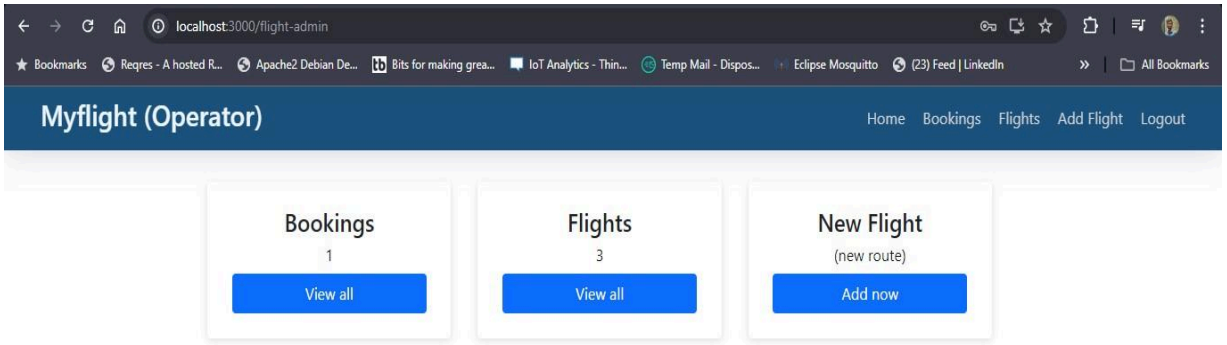
4. Flight Details



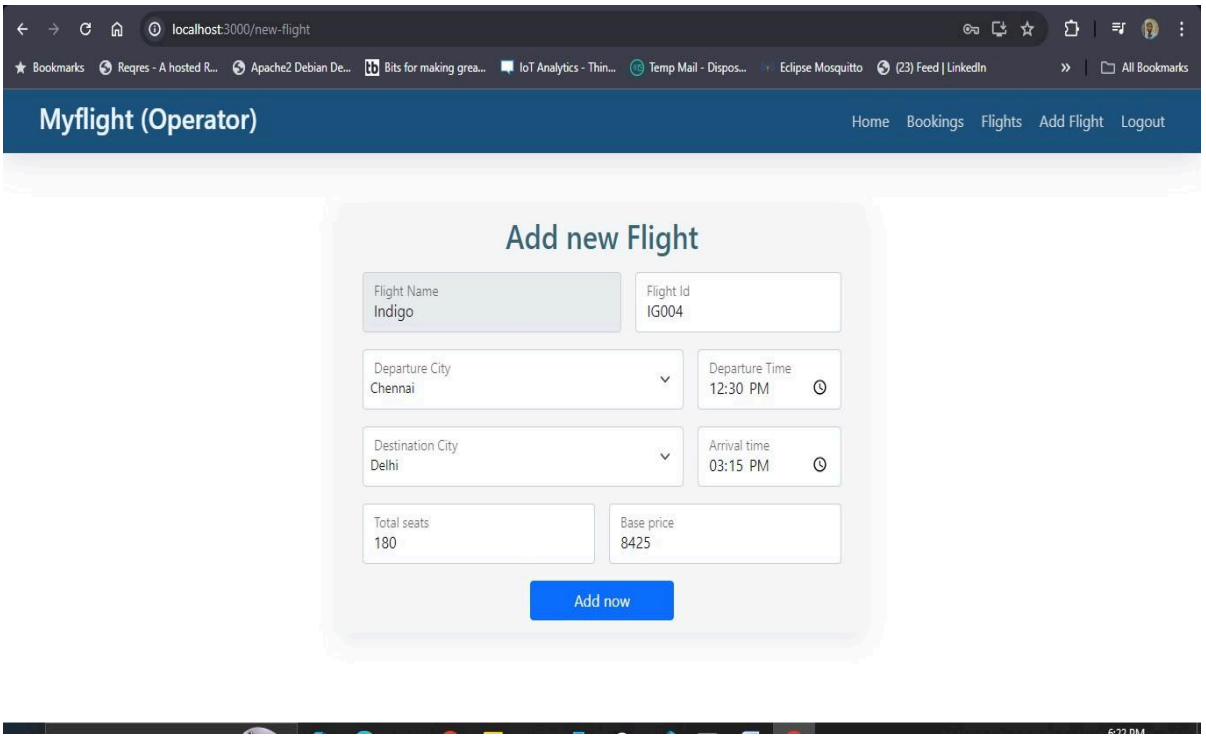
5. Bookings Page



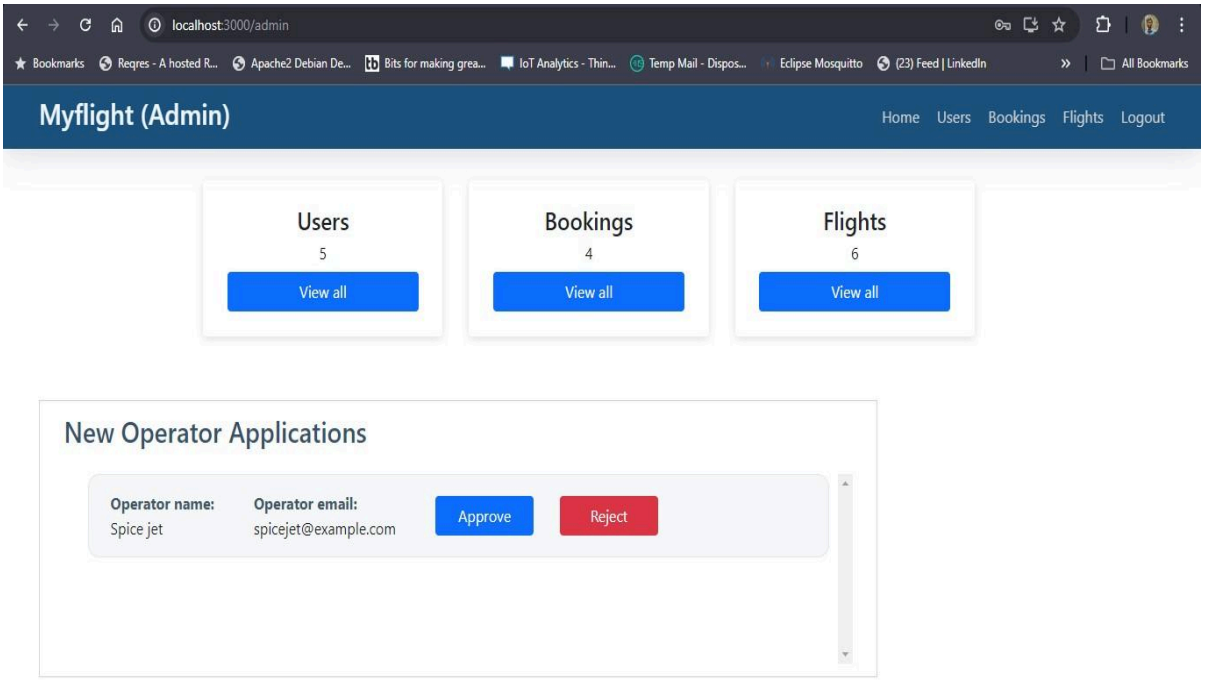
6. Flight-Operator Dashboard



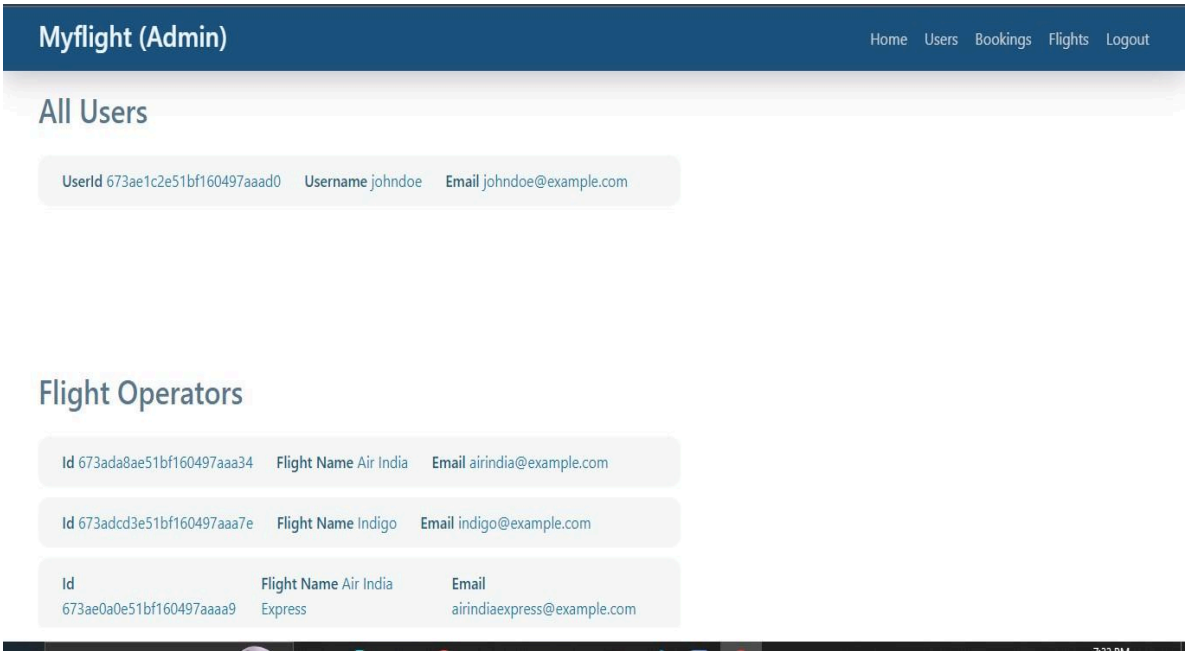
7. Add New Flight Page



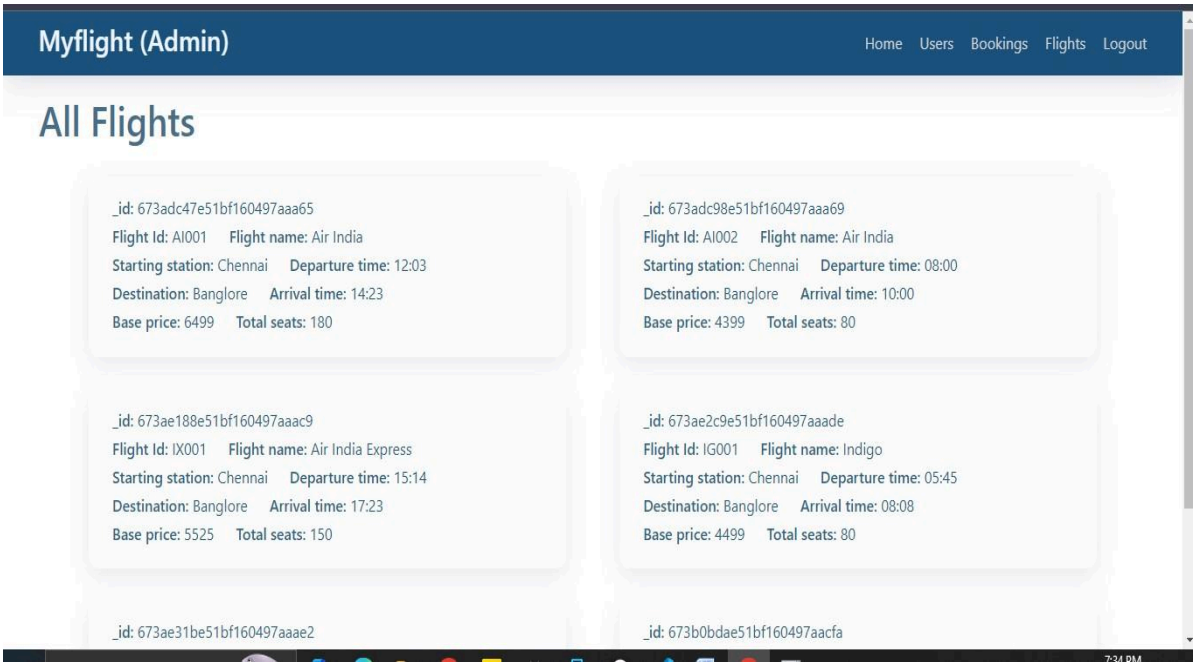
8. Admin Dashboard



9. All Users page



10. All Flights Page



Known Issues

- Admin lacks a dedicated login functionality but only a route.
- No robust error handling for invalid uploads.
- No payment since, the instructor of the course, told us not to include any payment gateway.
- Doesn’t fetch the image, video names while editing the profile.

Future Enhancements

- Add payment gateway for flight booking.
- Implement advanced analytics for admin insights.
- Include a recommendation engine for personalized suggestions.
- Integrate AI for automate special features.