

1. Introduction

- **Project Title:** Flight Finder : Navigating Your Travek Booking options

- **Team Members:**

P.Manisha 218X1A05F7

DT.Suvarna Lakshmi 208X1A0515

T.RudraReddy 228X5A0517

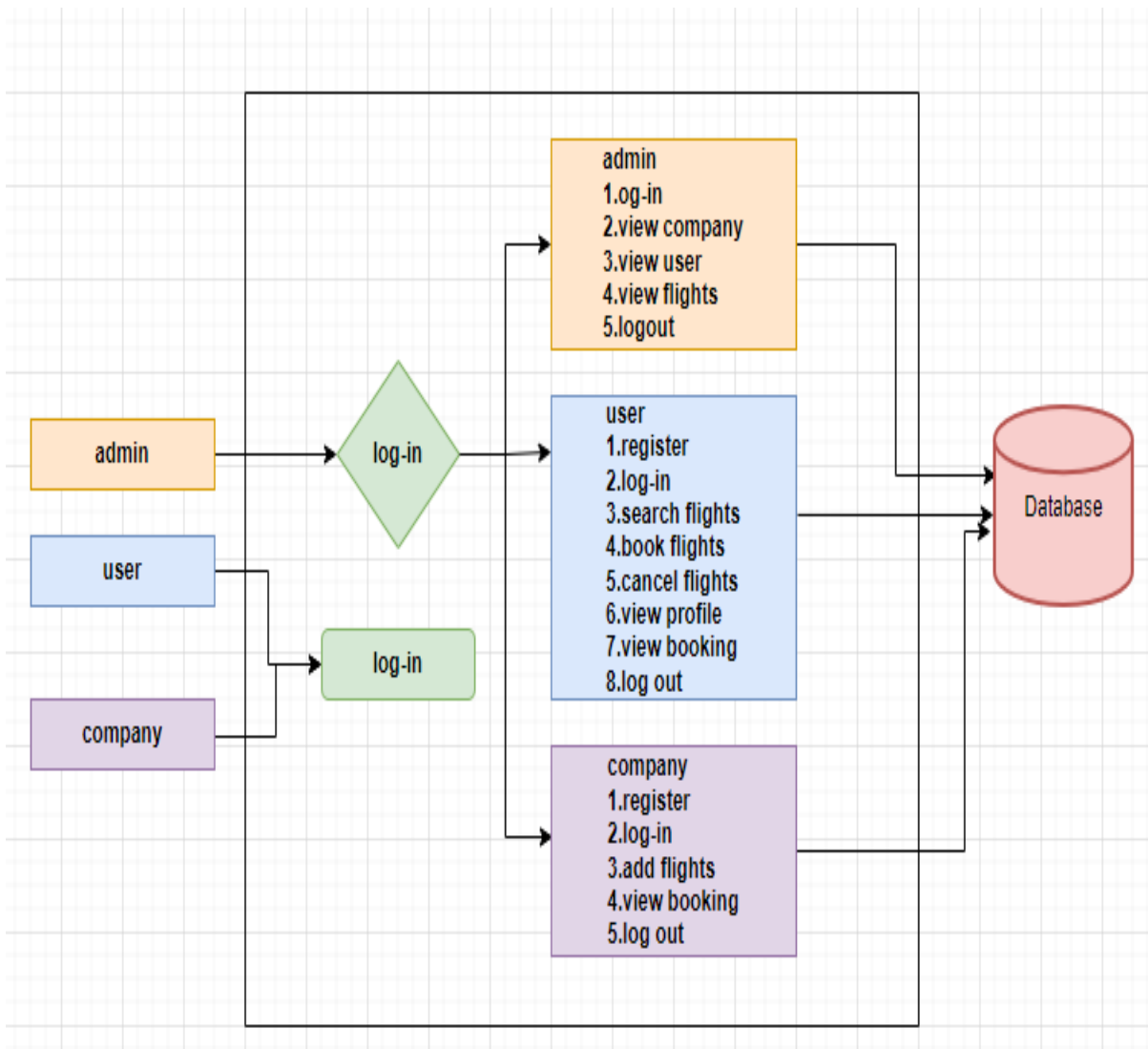
B. Ratan Singh 218X1A05D8

V. Naga Sai 218X1A05G3

2. Project Overview

- **Purpose:** The project presents a flight booking system that streamlines booking for users, flight management for companies, and system oversight for administrators.
- **Features:**
 - User registration, login, and flight search.
 - CRUD operations for companies on flight management.
 - Administrative approval for companies and management of flights.

3. Architecture



Frontend (React.js):

The frontend of this project is built using **React.js**, a powerful library for building user interfaces. It enables users to interact with the application for flight searches, bookings, cancellations, and profile management. The frontend uses **React Router** for navigation and **Axios** for API calls to the backend.

Backend (Node.js + Express.js):

The backend is built using **Node.js** and **Express.js**, serving as the API that handles user, company, and admin interactions. It communicates with the frontend using RESTful APIs and connects to the **MongoDB** database for storing and retrieving data.

- **User Module:**

- Allows users to register, log in, search for flights, book, and cancel tickets.
- **Company Module:**
 - Enables companies to register, log in, and manage flights (CRUD operations).
- **Admin Module:**
 - The admin oversees company registration, user management, and flight monitoring.

Database (MongoDB):

The system uses **MongoDB** as a NoSQL database to store data related to users, flights, bookings, and companies. The database is structured to allow efficient querying and scalability for managing large volumes of data.

- **Users Collection:** Stores user profiles and login details.
- **Flights Collection:** Contains all flight details (e.g., flight number, origin, destination, time, capacity).
- **Bookings Collection:** Manages user bookings and cancellation records.
- **Companies Collection:** Stores company details and their registered flights.

4. Setup Instructions

Prerequisites:

Before setting up the environment, ensure the following software is installed:

- **Node.js:** Download and install from [Node.js Official Website](https://nodejs.org/en/). Make sure to install the LTS version. This will also install **npm** (Node Package Manager) by default.
 - **Check Installation:**
 - Open terminal and run


```
node -v
```

```
npm -v
```

MongoDB: Install MongoDB from [MongoDB Official Website](https://www.mongodb.com/).

Install MongoDB:

- Follow the setup instructions for your operating system.
- After installation, verify the MongoDB service is running using:

`Mongod`

- Install MongoDB Compass (optional), the GUI for managing MongoDB databases.

Step-by-Step Installation Guide:

1. Clone the Project Repository:

- Navigate to your desired folder in the terminal and run:

`git clone <repository_url>`

`cd <project_directory>`

1. Backend Setup (Node.js and Express.js):

- Navigate to the server folder: `cd server`
- Install required Node.js packages (e.g., Express.js) using npm:

`npm install`

- **Packages include:**

- ▢ Express.js: A minimal web framework for Node.js.
- ▢ Mongoose: A MongoDB object data modeling (ODM) library.
- ▢ JWT: For handling authentication and authorization.
- ▢ Cors: Middleware for handling cross-origin requests.

- **Set Environment Variables:**

- ▢ Create a `.env` file in the server folder:

`touch .env`

- ▢ Add environment variables like MongoDB URI, API keys, and JWT secret in the `.env` file.

`MONGODB_URI=mongodb://localhost:27017/flight-booking`

JWT_SECRET=mysecretkey

- **Run the Server:**

npm start

- The backend server should now be running on the specified port (e.g., <http://localhost:5000>).

2. Frontend Setup (React.js):

- Navigate to the client folder:

cd client

- Install React.js and required dependencies using npm:

npm install

- **Packages include:**

- ▢ React Router: For managing navigation within the app.
- ▢ Axios: For making HTTP requests.
- ▢ Redux (optional): For state management if needed.

- **Run the React App:**

npm start

- The React application should now be running on <http://localhost:3000>.

3. Deployment Environment:

- If deploying to platforms like **Heroku**, **Netlify**, or **AWS**, follow their respective deployment instructions:

- ▢ **Heroku Deployment:**

- ▢ Install the Heroku CLI from Heroku Official Website.

- ▢ After creating a Heroku app:

heroku create

git push heroku main

Netlify Deployment:

- ▢ For the frontend, use [Netlify](#) to host the React app.

- Simply connect your repository and deploy via the Netlify dashboard.
- **AWS Deployment:**
- Use **Elastic Beanstalk** or **EC2** instances for deploying your backend and **S3** buckets for frontend hosting.
- **Environment Variables:** Make sure to configure environment variables properly on the deployment platform to handle MongoDB URI, API keys, etc.

5. Folder Structure

Client Structure (React.js):

The **client** folder contains the React frontend code. It is structured as follows:

- **public/**: Static assets and the HTML template used for the single-page application.
- **src/**: The source folder where all the React components and logic are located.
 - **components/**: Contains reusable UI components such as forms, buttons, and navigation.
 - **pages/**: Contains pages such as Home.js, Login.js, FlightSearch.js, and Profile.js.
 - **services/**: Contains functions that make API calls using axios to interact with the backend.
 - **redux/** (optional): If using Redux for state management, this folder contains the actions, reducers, and store setup.
 - **App.js**: The root component that brings together all the routes and logic for the app.
 - **index.js**: Entry point for the React application.

Server Structure (Node.js + Express.js):

The server folder contains the backend logic using Node.js and Express.js. The structure is as follows:

- **config/:** Contains configuration files for the database, environment variables, and middleware.
- **controllers/:** Business logic to handle incoming requests (e.g., flight booking, user management).
- **models/:** Defines Mongoose models such as User, Flight, Booking, and Company.
- **routes/:** Defines the RESTful API routes (e.g., /api/users, /api/flights).

- **Example route setup for flights:**

```
const express = require('express'); const router = express.Router(); const {
  getFlights, createFlight } = require('../controllers/flightController');
router.get('/', getFlights); router.post('/', createFlight); module.exports =
router;
```

middlewares/: Contains authentication middleware for protecting routes (e.g., JWT token verification).

app.js: The main application file that sets up middleware, routes, and server listening.

package.json: Manages backend dependencies and scripts.

6. Running the Application

To run the application locally, you need to start both the frontend (React.js) and backend (Node.js) servers.

Running the Backend (Node.js + Express.js):

1. **Navigate to the server directory:**

```
cd server
```

2. **Start the MongoDB service:**

- **Ensure MongoDB is running either locally or on a cloud service like MongoDB Atlas.**

mongod

3. Start the backend server:

npm start

- **The backend server will now run on `http://localhost:5000`. This server handles API requests for flight bookings, user authentication, company operations, and admin functions.**

Running the Frontend (React.js):

1. Navigate to the client directory:

cd client

- #### **2. Start the frontend server:**
- **The React.js app will run on `http://localhost:3000` and act as the user interface for interacting with the flight booking system.**

Both servers need to be running simultaneously for the full-stack application to function correctly.

7. API Documentation

The backend exposes several API endpoints to handle various functionalities like user registration, flight management, and bookings. Below is a documentation of the key API endpoints.

User APIs:

- **POST /api/users/register:** Register a new user.
 - Request Body: { "name": "John Doe", "email": "john@example.com", "password": "password123" }
 - Response: User object with a token for authentication.
- **POST /api/users/login:** User login to obtain authentication token.
 - Request Body: { "email": "john@example.com", "password": "password123" }
 - Response: { "token": "JWT_TOKEN" }

Flight APIs:

- **GET /api/flights:** Fetch all available flights.
 - Response: List of flight objects.
- **POST /api/flights:** Add a new flight (Company only).
 - Request Body: { "flightNumber": "123", "departure": "NYC", "arrival": "LAX", "price": 200 }
 - Response: Success message with flight details.
- **DELETE /api/flights/**

: Delete a flight by its ID (Company only).

Booking APIs:

- **POST /api/bookings:** Book a flight (User only).
 - Request Body: { "userId": "user_id", "flightId": "flight_id" }
 - Response: Success message and booking details.
-

DELETE /api/bookings/ :

Cancel a booking by its ID.

Admin APIs:

- **GET /api/admin/companies:** View and manage company registrations (approve or reject).
- **GET /api/admin/users:** View registered users and their bookings.

8. Authentication

The system uses JWT (JSON Web Tokens) for authentication to ensure secure access for users, companies, and admins.

Process:

1. User Authentication:

- **During login, a JWT is generated for the user and returned in the response. This token must be included in the headers of subsequent API requests for authentication.**
- **Example: Authorization: Bearer <token>.**

2. Company Authentication:

- **Similar to users, companies log in and receive a token to manage their flights securely.**

3. Admin Authentication:

- **Admins log in and obtain a token that grants them privileges to view users, manage companies, and oversee flight operations.**

Token Storage:

- **The tokens are stored securely on the client side (usually in local storage or cookies) and must be included in every request that requires authentication.**

Role-Based Access Control:

- **Users:** Can only access booking-related features
- **Companies:** Can manage their flights and view bookings.
- **Admins:** Have full access to the platform, including user and flight management.

9. User Interface

The user interface is designed to be intuitive, clean, and responsive. Below are some of the key screens and features:

1. Home Page:

- The landing page displays a search form where users can input their desired departure and arrival locations, along with dates to search for available flights.

2. Registration and Login:

- Users and companies have separate registration and login pages, allowing them to create an account or log in securely.

3. Flight Search:

- Users can search for flights, view available options, and see details like price, flight number, and seat availability.

4. Profile and Booking Management: Users have access to a profile page where they can update their details and view their booking history.

- Companies can view and manage their flights, while admins can see all registered users and companies.

5. Admin Dashboard:

- Admins can view and manage user details, company registrations, and flight details.

10. Testing

Testing Strategy:

The application has been tested using unit, integration, and functional tests to ensure proper functionin

Types of Tests:

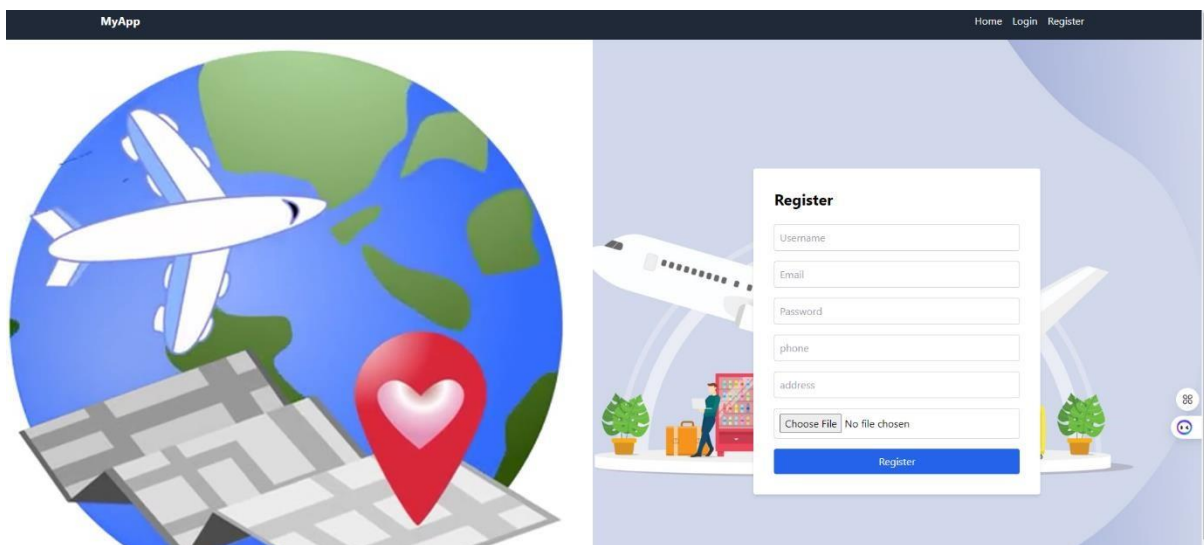
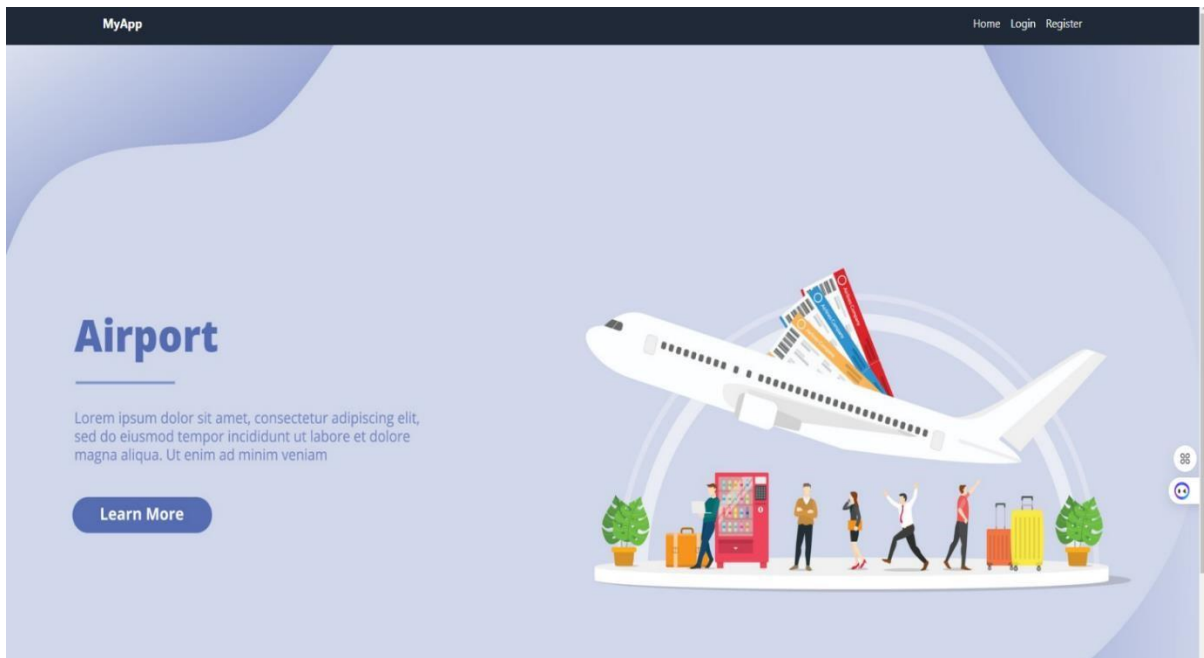
- 1. Unit Testing:** Tests individual components like the flight search or booking functionality.
- 2. Integration Testing:** Ensures that the frontend and backend interact properly, particularly when making API calls.
- 3. Functional Testing:** Tests the overall functionality such as booking a flight, canceling it, and ensuring user profiles work as expected.

Testing Tools:

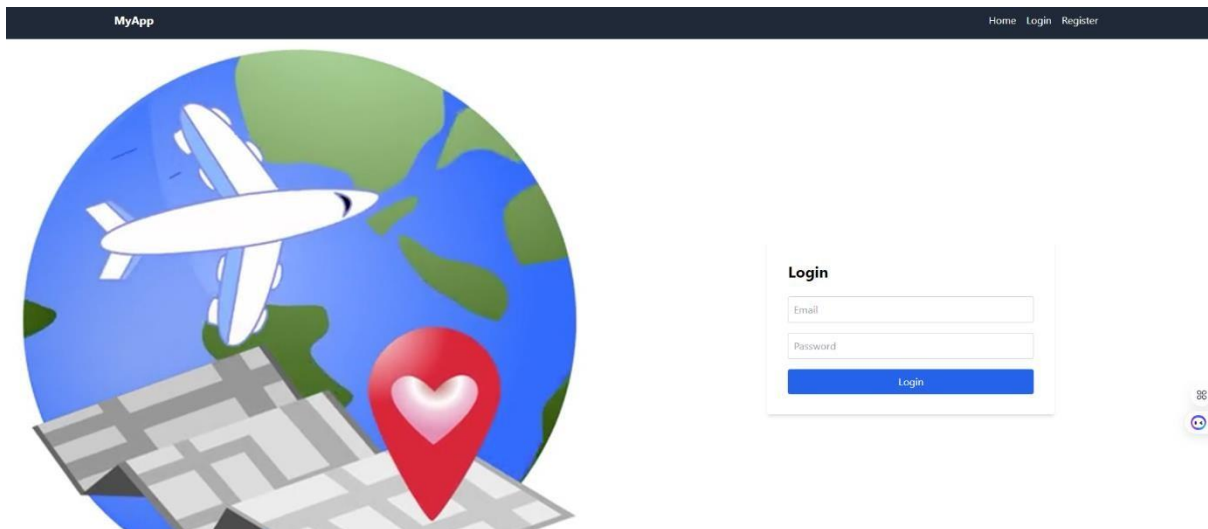
- **Jest:** Used for unit testing React components.
- **Postman:** For testing API endpoints and ensuring they return the correct responses.
- **Mocha/Chai:** For backend testing, particularly focusing on API logic and database interactions.

11. Screenshots or Demo

Home Page: This is the project's landing page.



Login Page: Both customers and admins can use this page to access their accounts and perform additional operations after logging in.





Admin Home Page: Company Details - The admin can view the details of registered companies. Company personnel can log in only after admin approval. If the admin rejects the request, the company will not be able to log

Admin Panel

- Company
- Users
- Flights
- Logout

Company Table

Search by name...

| Photo | Name | Email | Mobile | Address | Website | Action |
|---|---------|-------------------------|-------------|----------|---|-----------------|
|  | vinay | ramcharan33@gmail.com | 09012345678 | sgv | https://chatgpt.com/c/6708ac36-7470-800d-9f6d-7d35cbeb00bb | Approved |
|  | Takeoff | cse@takeoffprojects.com | 9874563210 | Tirupati | https://ymtsindia.com/ | Select Action ▼ |

View Registered Customer Details: The admin can view the details of registered customers or users.

Admin Panel



Company

Users

Flights

Logout

Search by name...

| Photo | Name | Email | Mobile | Address |
|---|--------|-----------------------|-------------|---------|
|  | vinay | ramcharan33@gmail.com | 09012345678 | sgv |
|  | lalith | ramcharan33@gmail.com | 09012345678 | sgv |

PreviousPage 1 of 1Next

88

View Flight Details: The admin can view flight details on this page.

Admin Panel

Company

Users

Flights

Logout

Flight Schedule

| Flight Name | Flight No | Start Location | End Location | Start Time | End Time | seat capacity | Available Seats |
|-------------|-----------|----------------|--------------|------------|----------|---------------|-----------------|
| indigo1bn | fp123 | tirupati | chennai | 11:11 | 12:12 | 4000 | 193 |

Flight list

View Flight Details: The admin can view flight details on this page.

Admin Panel

Company
Users
Flights
Logout

Flight list


Flight Schedule

| Flight Name | Flight No | Start Location | End Location | Start Time | End Time | seat capacity | Available Seats |
|-------------|-----------|----------------|--------------|------------|----------|---------------|-----------------|
| indigojn | fp123 | tirupati | chennai | 11:11 | 12:12 | 4000 | 193 |

Company Registration Page: Flight companies can register by filling in their name, email, password, mobile number, address, website, and uploading a photo.

Company Logo

Flights Register Login



Name

Email

Password

Mobile Number

Address

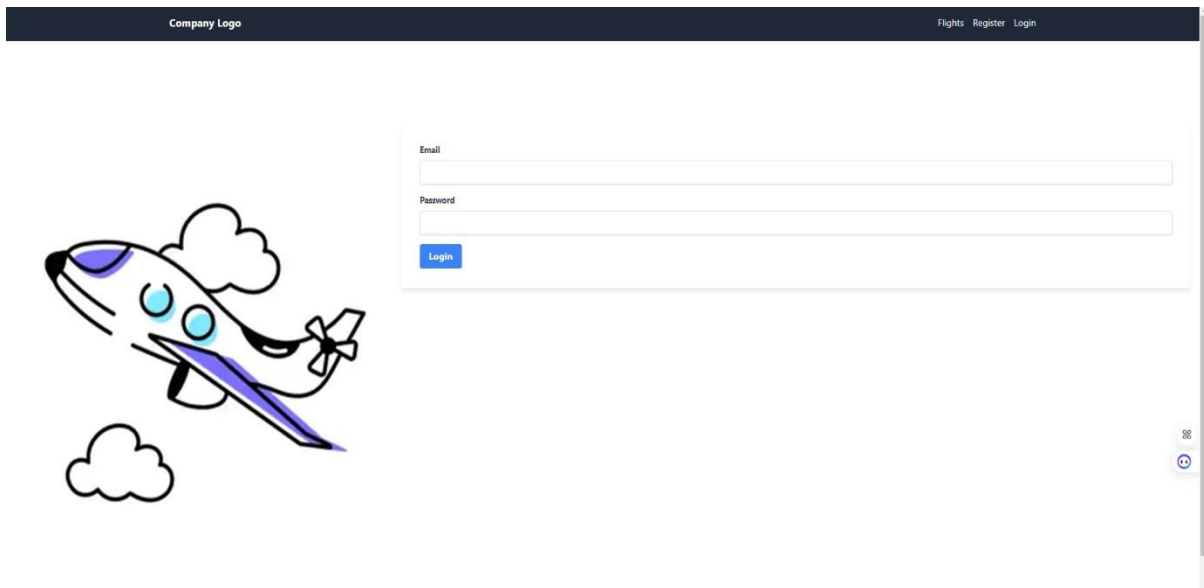
website

Photo

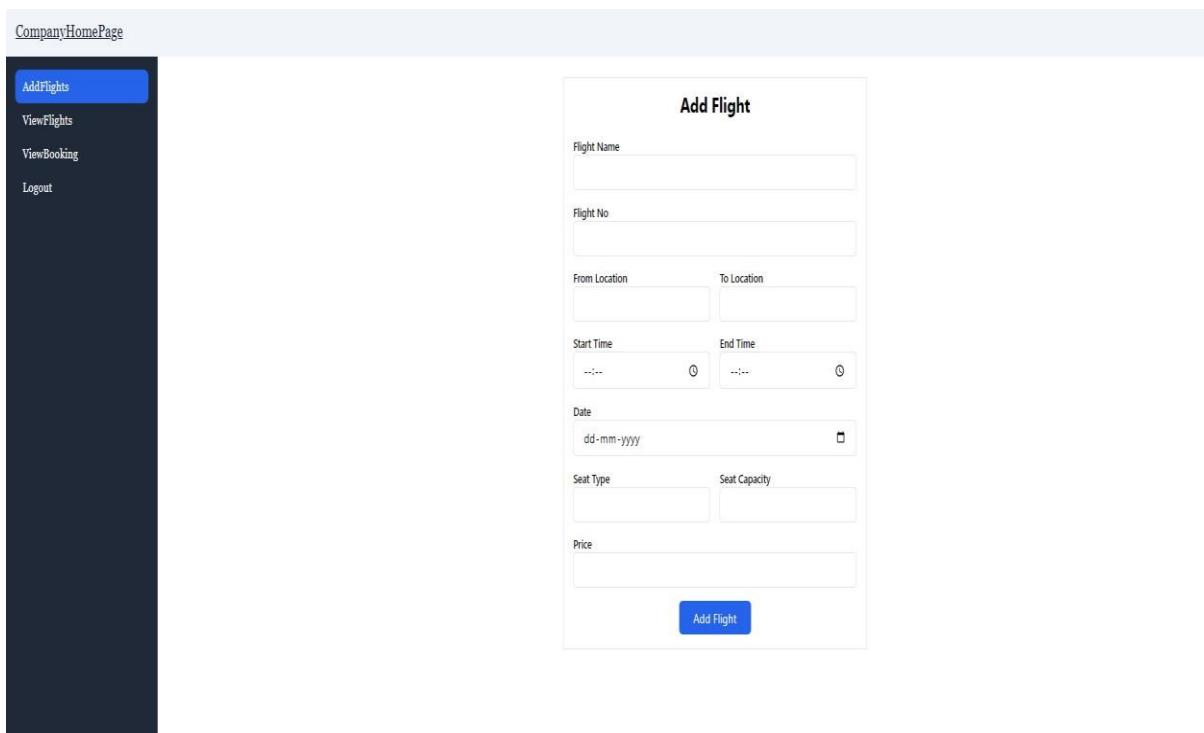
Choose File No file chosen

Register

Company Login Page: A company can log in using its email and password only after receiving approval from the admin.



Flight Adding Page: This page allows users to add flight details by entering the name, flight number, departure and arrival locations, start and end times, seat capacity, and price.



View Added Flight Details: Users can view flight details here, and if needed, they can update or delete the flight information.

CompanyHomePage

AddFlightsViewFlightsViewBookingLogout

Flights found

View Flights

| Flight Name | Flight No | From | To | Start Time | End Time | Date | Seat Type | Capacity | Price | Status | Actions |
|-------------|-----------|----------|---------|------------|----------|--------------------------|-----------|----------|-------|--------|-------------------------|
| indigo | fp123 | tirupati | chennai | 11:11 | 12:12 | 2024-10-20T00:00:00.000Z | common | 193 | 4000 | active | <div>UpdateDelete</div> |

View Booked Flight Details: Users can view their booked flight details here.

CompanyHomePage


AddFlightsViewFlightsViewBookingLogout

View Bookings

| Flight Name | Flight No | From | To | Date | Actions |
|-------------|-----------|----------|---------|--------------------------|---------------------|
| indigo | fp123 | tirupati | chennai | 2024-10-20T00:00:00.000Z | <div>Download</div> |

After Customer Login: Once logged in, customers can view the flight details as shown below.

SB FlightLogoutBookingProfile



Search Flights

Start Location

End Location

Date

Select start location

Select end location

dd-mm-yyyy

Search Flights

Available Flights:

tirupati to chennai

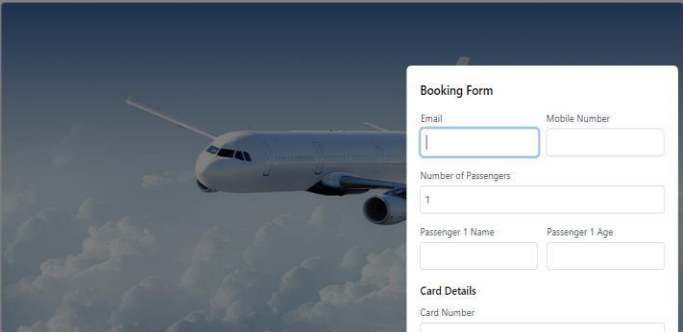
Departure: 2024-10-20 at 11:11 - 12:12

Price: ₹4000 | 195 seats available

Book Now

Flight Booking Form: This form contains the details required for flight booking, allowing customers to reserve their flight seats.

SB FlightLogoutBookingProfile



Search Flights

Start Location

End Location

Date

Select start location

Select end location

dd-mm-yyyy

Search Flights

Available Flights:

tirupati to chennai

Departure: 2024-10-20 at 11:11 - 12:12

Price: ₹4000 | 195 seats available

Book Now

Booking Form

Email

Mobile Number

Number of Passengers

1

Passenger 1 Name

Passenger 1 Age

Card Details

Card Number

Expiry Date

CVV

Cancel

Confirm Booking

After Booking: Once the ticket is successfully booked, customers can view their booked flight details as shown below.

My Booking Details

Flight Details:

From: tirupati

To: chennai

Date: 10/20/2024

Time: 11:11

Price: 8000

Passenger Details:


Name: Lakshmi - **Age:** 25 seat:fp123-006

Name: Siri - **Age:** 16 seat:fp123-007

Cancel Booking

Profile Page and Change Password Page: Customers can view their profile, and if they wish to change their password, they can do so here

User Profile



Lakshmi
lakshmi@gmail.com
9632587410
Tirupati

Change Password

User Profile



Lakshmi
lakshmi@gmail.com
9632587410
Tirupati

Change Password

Change Password

Old Password
Enter your old password

New Password
Enter your new password

Submit Cancel

12. Known Issues

Here are some known issues currently identified in the project:

1. **Concurrent Bookings:** Handling multiple concurrent bookings for the same flight can occasionally result in overbooking due to race conditions.
2. **Session Management:** There is no session expiration for JWT tokens, meaning users remain logged in until they manually log out.
3. **UI Bugs on Mobile:** Some parts of the user interface may not render properly on smaller mobile screens, particularly the booking form.
4. **No Email Confirmation:** Users don't receive an email confirmation after booking a flight, which could be improved by adding an email service like SendGrid.

13. Future Enhancements

Here are some potential future improvements to the system:

1. **Mobile App Development:** Create a mobile app for the platform to enhance user experience and provide booking services on the go.
2. **Enhanced Security:** Implement two-factor authentication (2FA) for users and companies to improve security during login.
3. **Online Payment Integration:** Allow users to make payments directly on the platform using services like Stripe or PayPal.
4. **AI-Powered Recommendations:** Add machine learning algorithms to suggest flights to users based on their past search and booking behavior.
5. **Analytics Dashboard for Admins:** Provide detailed insights and analytics for admins, showing user engagement, flight booking trends, and company performance.
6. **Improved Notifications:** Add email and SMS notifications for booking confirmations, flight reminders, and cancellations.
7. **Feedback System:** Implement a feedback mechanism where users can rate their booking experience and provide suggestions for improvements.