

1. Introduction

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

- **Team Members:**

B.Bhavani 218X1A0449

K.Sushma 218X1A0441

G.Ayyappa 218X1A0453

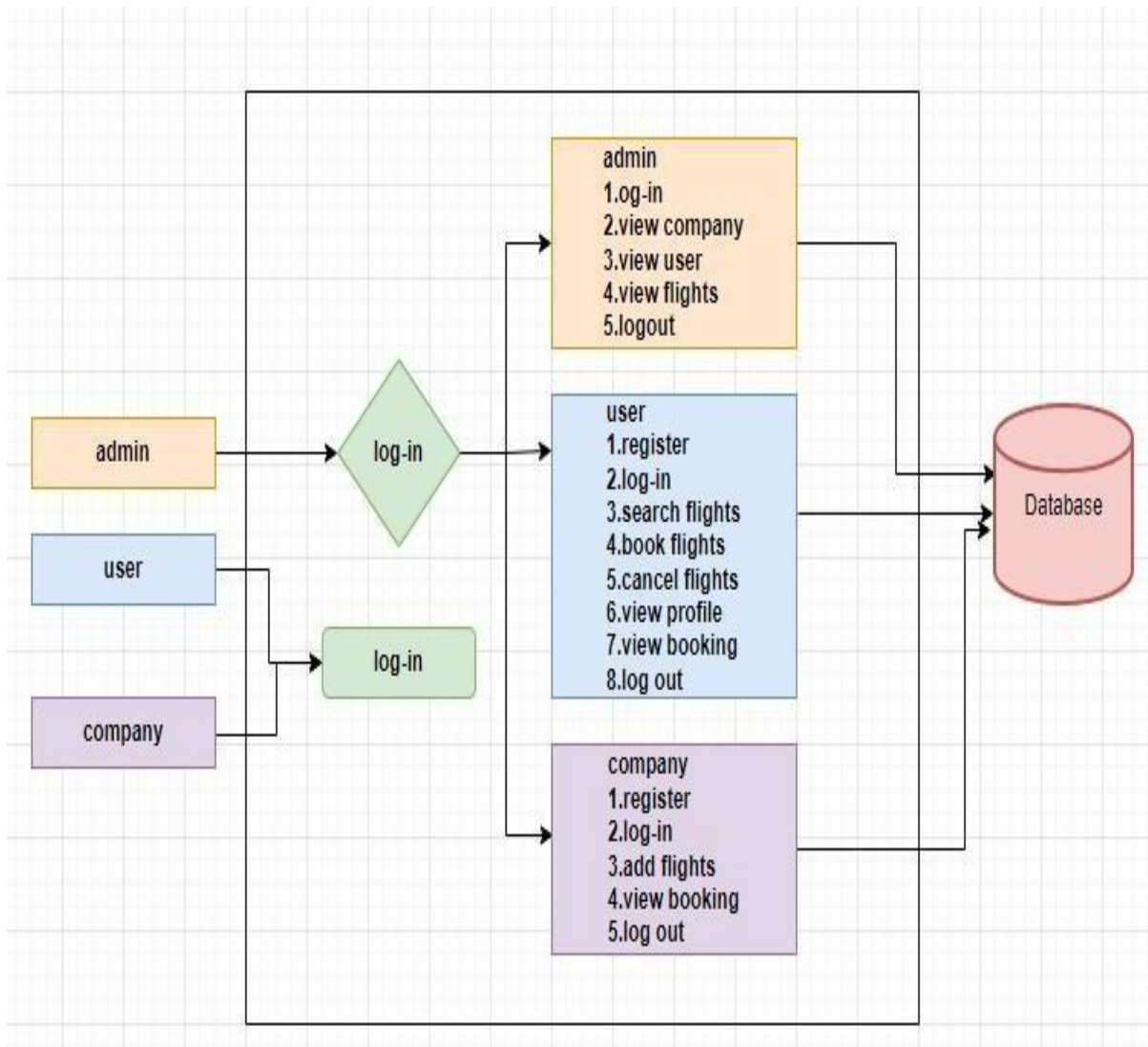
K.Hemanthkumar 218X1A0458

B.Lavakumar 218X1A0405

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: `npm start` . 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" }

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:

Unit Testing: Tests individual components like the flight search or booking functionality.

Integration Testing: Ensures that the frontend and backend interact properly, particularly when making API calls.

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

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.

Results:

Customer Registration Page: New customers must create an account on this portal before booking. To do so, they need to fill out the required information for account creation.

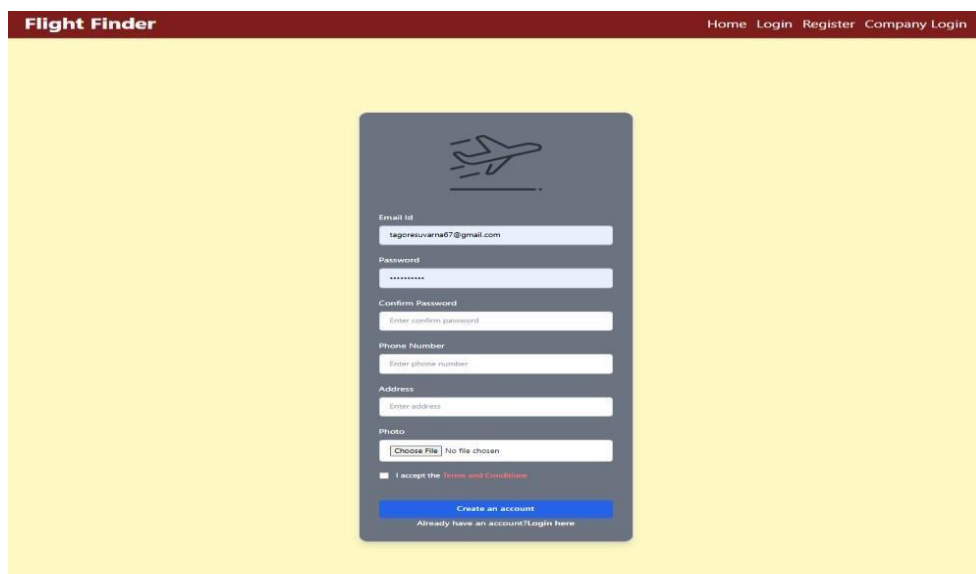
The screenshot shows the 'Flight Finder' website's customer registration page. The page has a dark red header with the site name and navigation links. The main content area is yellow and features a dark grey registration form. The form includes fields for Email Id, Password, Confirm Password, Phone Number, Address, and Photo. It also has a checkbox for accepting terms and conditions, and buttons for 'Create an account' and 'Already have an account? Login here'.

Fig : Customer registration Page

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

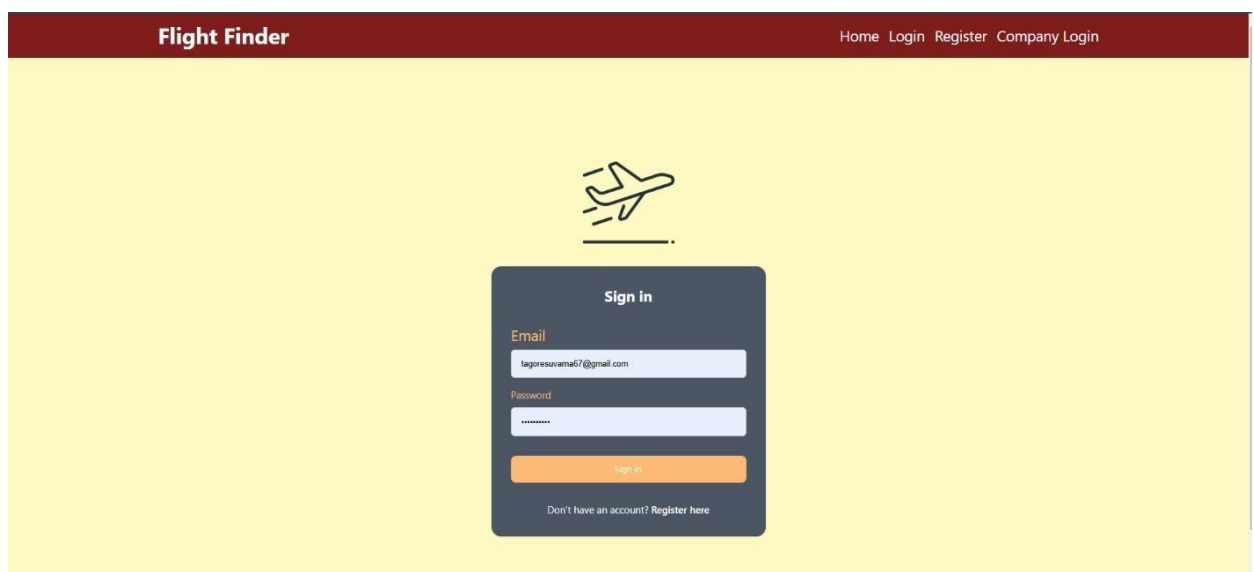
The screenshot shows the 'Flight Finder' website's login page. The page has a dark red header with the site name and navigation links. The main content area is yellow and features a dark grey login form. The form includes fields for Email and Password, a 'Sign in' button, and a link for 'Don't have an account? Register here'.

Fig: Login Page

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 in.



The screenshot shows the Admin Panel interface. On the left is a sidebar with links: Company, Users, Flights, and Logout. The main content area is titled 'Company Table' and features a search bar labeled 'Search by name...'. Below the search bar is a table with the following data:


Photo	Name	Email	Mobile	Address	Website	Action
	Indigo	indigo@gmail.com	1234567890	test	indigo.com	Approved
	Goibibo	goibibo@gmail.com	1234567890	Hyd	goibibo.com	Approved
	KHIT	Airtravel@gmail.com	1234567890	Guntur	khit.com	Approved

Fig : Admin Page

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



The screenshot shows the Admin Panel interface for viewing registered customer details. The sidebar on the left has links: Company, Users, Flights, and Logout. The 'Users' link is highlighted. The main content area has a search bar labeled 'Search by name...'. Below the search bar is a table with the following data:

Photo	Name	Email	Mobile	Address
	undefined	admin@gmail.com	1234567890	test
	undefined	tagoresuvarna67@gmail.com	7799353610	guntur
	undefined	goibibo@gmail.com	1234567890	ASD

At the bottom of the table, there are navigation buttons: 'Previous', 'Page 1 of 1', and 'Next'.

Fig: View Registered Customer details

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
Airbus 320	24543	Mumbai	Delhi	23:27	01:27	5300	140
SS	67	hyderabad	Delhi	22:51	01:51	5300	619
SK	123	Hyderabad	Gujarat	20:10	20:10	3100	679
KHIT	1234	Nepal	Hyderabad	17:16	14:19	2500	789
Airtravels	134	Bhoopal	Bangalore	14:22	18:19	5000	784
ASD	1263	Nandhayala	Nepal	14:25	15:20	1263	1263
MSD	777	Kerala	Delhi	14:26	14:24	10000	50
Leo	6751	Paris	LA	14:27	16:23	15000	100
AirAsia	45678	Italy	Europe	14:28	19:26	25000	120

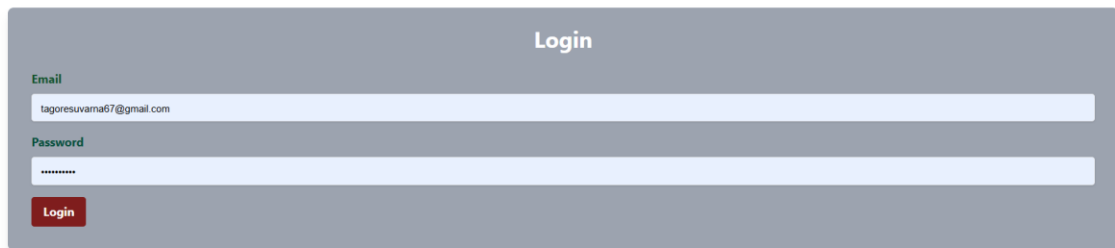
Fig : View Flight Details

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

Flight Finder	Home Register Login
<p align="center">Register</p> <p>Name <input type="text"/></p> <p>Email <input type="text" value="tagoresuvarna67@gmail.com"/></p> <p>Password <input type="password" value="*****"/></p> <p>Mobile Number <input type="text"/></p> <p>Address <input type="text"/></p> <p>Website <input type="text"/></p> <p>Photo <input type="button" value="Choose File"/> No file chosen</p> <p align="center"><input type="button" value="Register"/></p>	

Fig : Company Registration Page

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

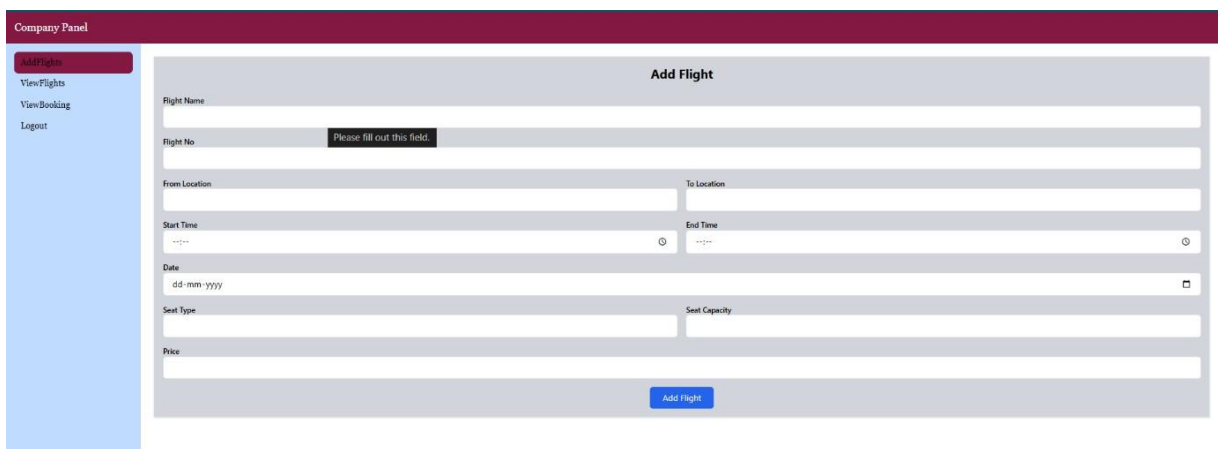


The login form is titled "Login" and is set against a light gray background. It contains two input fields: "Email" with the value "tagoresuvarna67@gmail.com" and "Password" with masked characters "*****". A red "Login" button is positioned below the password field.

© 2024 Company Name. All rights reserved.

Fig: Company Login Page

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.



The "Flight Adding Page" features a dark red header with "Company Panel" on the left. A light blue sidebar contains navigation links: "Add Flights" (highlighted), "View Flights", "View Booking", and "Logout". The main content area, titled "Add Flight", contains several input fields: "Flight Name", "Flight No" (with a "Please fill out this field." error message), "From Location", "To Location", "Start Time" and "End Time" (with time pickers), "Date" (with a calendar icon), "Seat Type", "Seat Capacity", and "Price". A blue "Add Flight" button is at the bottom right.

Fig : Flight Adding Page

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

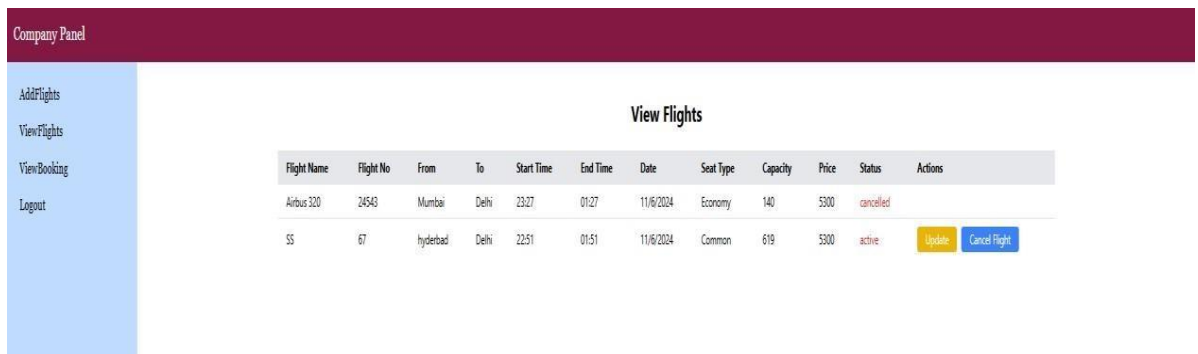


Fig : View Added Flight Details

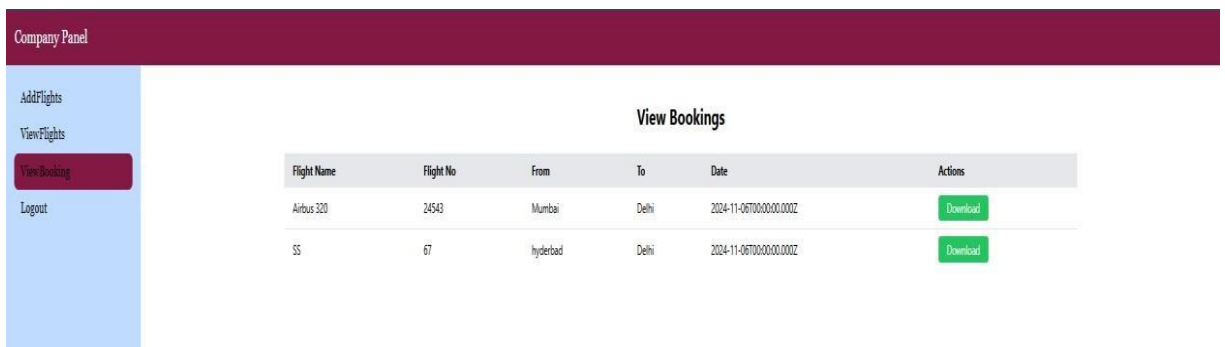


Fig : View Booked Flight Details

View Booked Flight Details: Users can view their booked flight details here. Booked Customer Details in PDF: Company can download the booked details in PDF format, which includes the customer information.

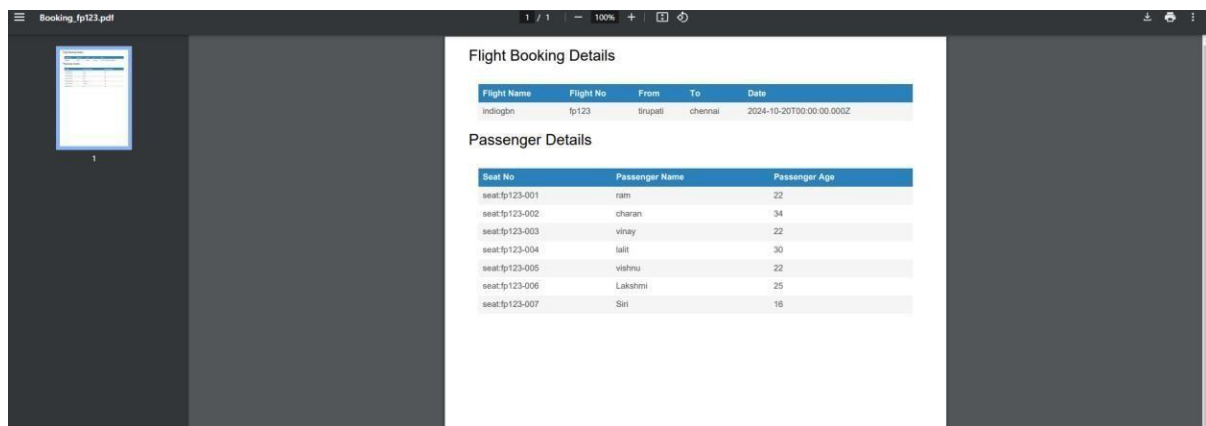
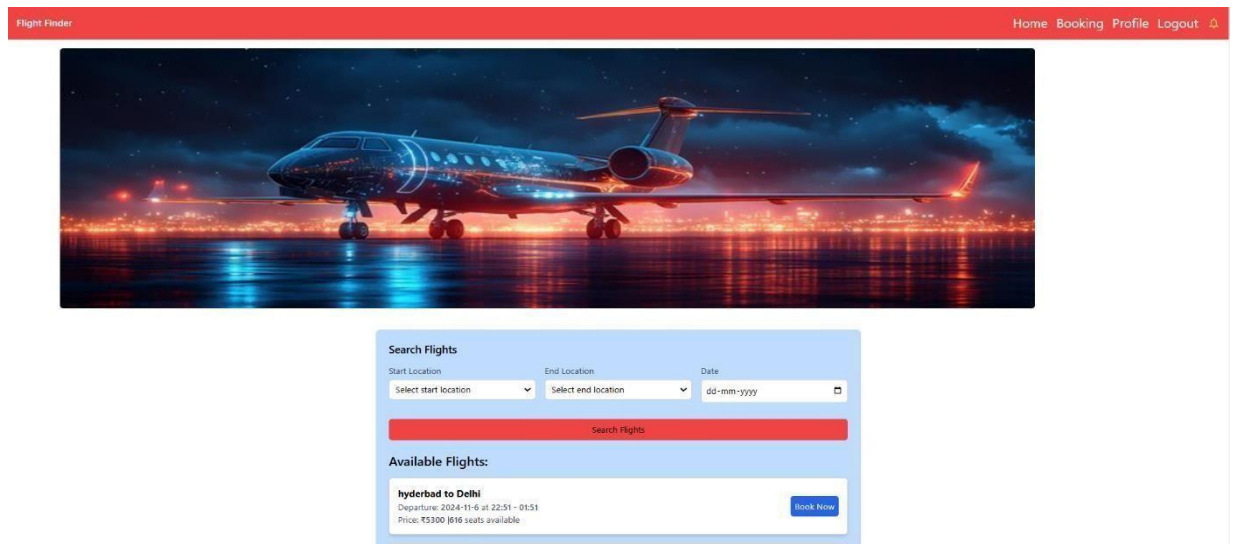


Fig : Booked Customer Details in PDF

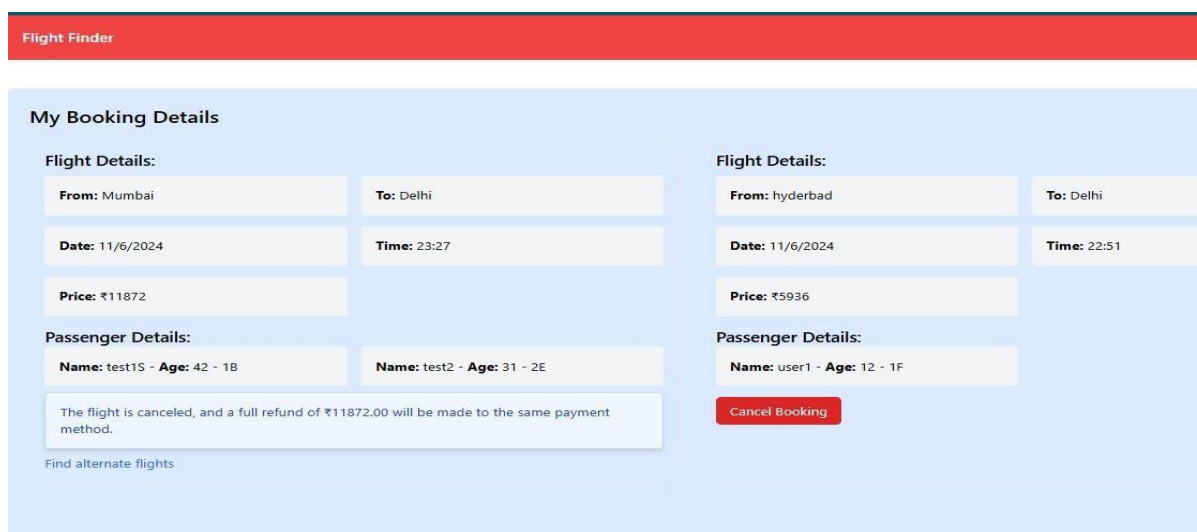
After Customer Login: Once logged in, customers can view the flight details as shown below. **Flight Booking Form:** This form contains the details required for flight booking, allowing customers to reserve their flight seats.



The screenshot displays the 'Flight Finder' web application. At the top, a red navigation bar contains the text 'Flight Finder' on the left and 'Home Booking Profile Logout' on the right. Below the navigation bar is a large, vibrant image of a commercial airplane at night, illuminated by city lights and its own lights, with its reflection visible on the water. Below the image is a 'Search Flights' form. The form has three input fields: 'Start Location' with a dropdown menu showing 'Select start location', 'End Location' with a dropdown menu showing 'Select end location', and 'Date' with a text input showing 'dd-mm-yyyy' and a calendar icon. A red 'Search Flights' button is positioned below these fields. Below the search form, there is a section titled 'Available Flights:'. It displays a flight from 'hyderabad to Delhi' with a departure time of '2024-11-6 at 22:51 - 01:51' and a price of '₹5300 (616 seats available)'. A blue 'Book Now' button is located to the right of the flight details.

Fig : After Customer Login

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



The screenshot shows the 'My Booking Details' page. At the top, a red navigation bar contains the text 'Flight Finder'. Below the navigation bar is a light blue section titled 'My Booking Details'. This section is divided into two columns. The left column contains 'Flight Details:' with fields for 'From: Mumbai', 'To: Delhi', 'Date: 11/6/2024', 'Time: 23:27', and 'Price: ₹11872'. Below these is 'Passenger Details:' with fields for 'Name: test1S - Age: 42 - 1B' and 'Name: test2 - Age: 31 - 2E'. A message box states 'The flight is canceled, and a full refund of ₹11872.00 will be made to the same payment method.' Below the message is a link 'Find alternate flights'. The right column contains 'Flight Details:' with fields for 'From: hyderabad', 'To: Delhi', 'Date: 11/6/2024', 'Time: 22:51', and 'Price: ₹5936'. Below these is 'Passenger Details:' with fields for 'Name: user1 - Age: 12 - 1F' and a red 'Cancel Booking' button.

Fig : After 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.

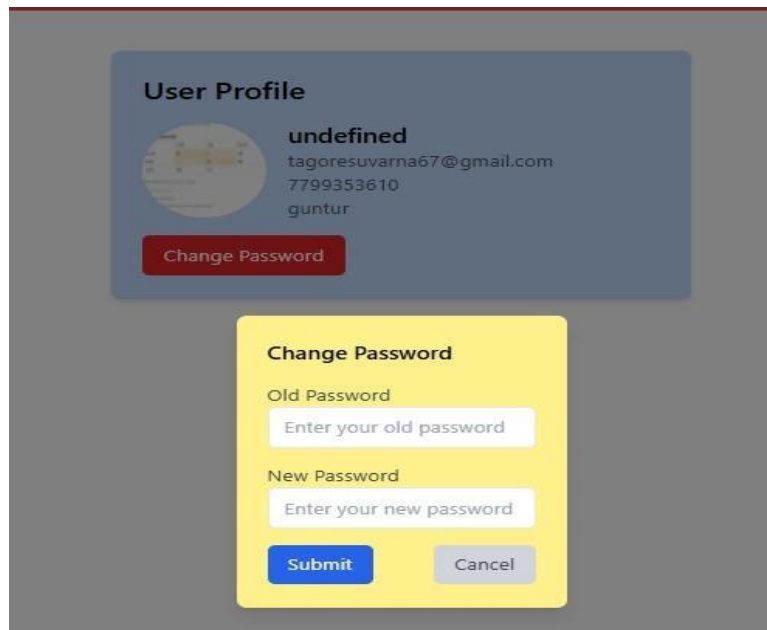


Fig: Profile Page and Change Password Page

Price Fluctuation Notification :When the companies tend to Increase or Decrease their Flight Booking Prices , a Notification will Displayed to users.



Fig : Price Fluctuation Notification

Alternate Flight Suggestions: Provide options for alternate flights when the original flight is canceled or delayed.

Flight Flinder

My Booking Details

Flight Details:

From: Mumbai

To: Delhi

Date: 11/6/2024

Time: 23:27

Price: ₹11872

Passenger Details:

Name: test1S - **Age:** 42 - 1B

Name: test2 - **Age:** 31 - 2E

The flight is canceled, and a full refund of ₹11872.00 will be made to the same payment method.

Find alternate flights

Fig : Alternate Flight Suggestions

Ticket Refund Charges: Calculate charges when a flight is canceled or delayed.

Flight Details:

Cancellation Details

Original Price:

₹5300.00

(-) Cancellation Deduction (30%):

₹1590.00

Ticket Refund Amount:

₹3710.00

(+) IGST Refund (12%):

₹636.00

Total Refund Amount:

₹4346.00

Confirm Cancel

Close

Cancel Booking

Fig : Ticket Refund Charges

Seat Allocation During Booking: Show available seats for passengers to choose from while booking tickets.

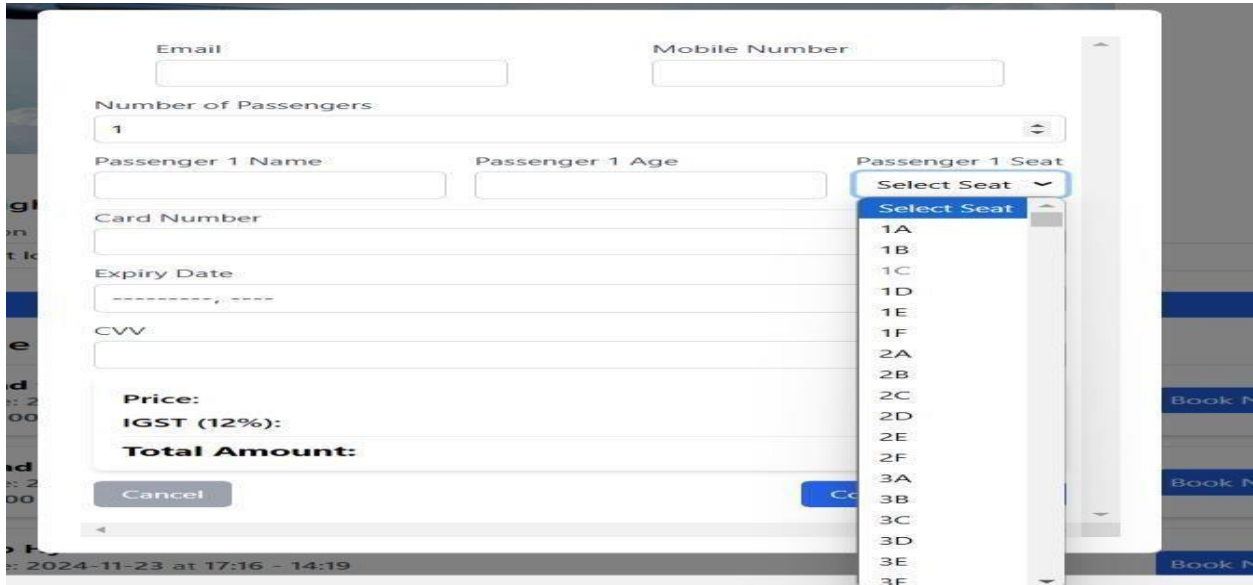
A screenshot of a flight booking form. The form includes fields for Email, Mobile Number, Number of Passengers (set to 1), Passenger 1 Name, Passenger 1 Age, Card Number, Expiry Date, and CVV. Below these fields, there are sections for Price, IGST (12%), and Total Amount. A dropdown menu for 'Passenger 1 Seat' is open, showing a list of available seats: 1A, 1B, 1C, 1D, 1E, 1F, 2A, 2B, 2C, 2D, 2E, 2F, 3A, 3B, 3C, 3D, 3E, and 3F. The form also has 'Cancel' and 'Book' buttons. The background shows a blurred view of the website's header and footer.

Fig : Seat Allocation during Booking

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.