MISRIMAL NAVAJEE MUNOTH JAIN ENGINEERINGCOLLEGE

(Managed By Tamil Nadu Educational and MedicalTrust) Thoraipakkam, Chennai – 600097.

NM1042

MERN Stack Powered by MongoDB

(FLIGHT BOOKING APP)

REGULATION – 2021

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

NAME	:
REGISTER NUMBER	:
YEAR	IV
SEMESTER	VII

MISRIMAL NAVAJEE MUNOTH JAIN ENGINEERING COLLEGE

(Managed By Tamil Nadu Educational and Medical Trust)Thoraipakkam, Chennai – 600097.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Register Number

BONAFIDE CERTIFICATE

This is to certify that this is a bonafide record of work done by___________________of IV-Year **B.E-Computer Science and Engineering** in the **NM1042-MERN Stack powered by MongoDB Laboratory** during the Academic year **2024-2025**

Staff In-Charge **Head of the Department**

Submitted for the University Practical Examination held on

Internal Examiner **External Examiner**

MISRIMAL NAVAJEE MUNOTH JAIN ENGINEERING COLLEGE, CHENNAI – 97

DEPARTMENT OFCOMPUTER SCIENCE AND ENGINEERING

VISION
Producing competent Computer Engineers with a strong background in the latest trendsand technology to achieve academic excellence and to become pioneer in software and hardware products with an ethical approach to serve the society
MISSION
To provide quality education in Computer Science and Engineering with the state of theart facilities. To provide the learning audience that helps the students to enhance problem solving skills and to inculcate in them the habit of continuous learning in their domain of interest. To serve the society by providing insight solutions to the real world problems by employing the latest trends of computing technology with strict adherence to professional and ethical responsibilities.

Flight Booking App

1. Introduction

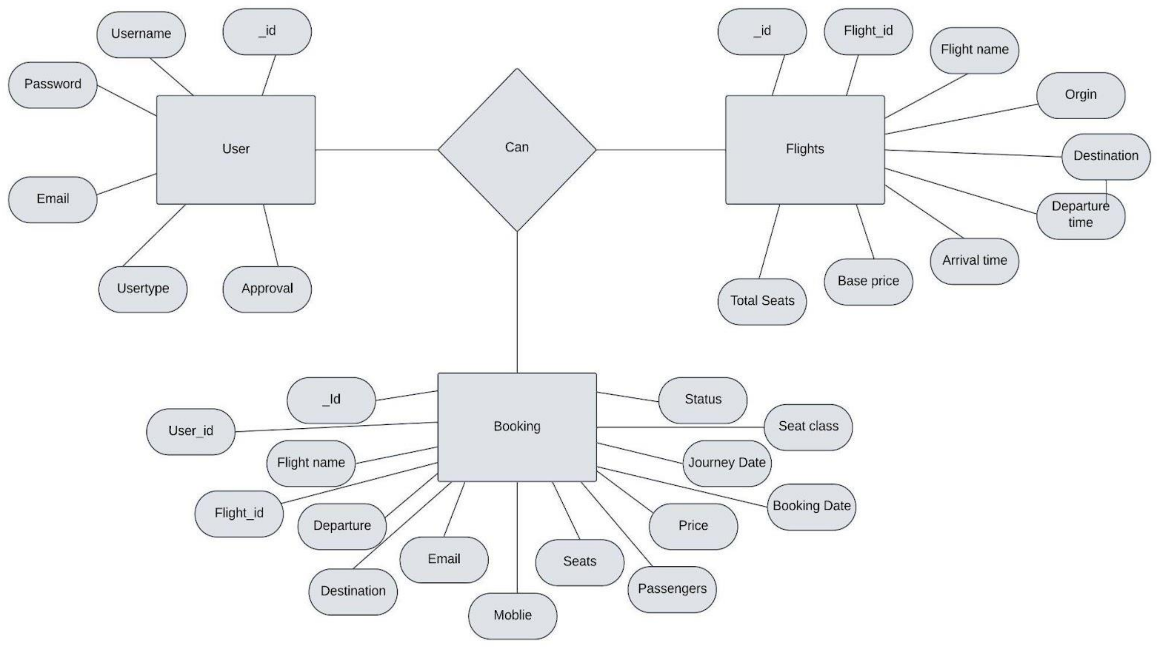
- **Project Title:**Flight Booking App
- **Team Members:**
  - Jaganathan S
  - Thameem Sayyed
  - Viknesh H
  - Pradeep M

---

2. Project Overview

- **Purpose:** The Flight Booking App is designed to revolutionize the flight ticket booking experience. It aims to offer users convenience, efficiency, and customization, simplifying the travel process for both frequent and occasional travelers.
- **Features:**
  - Search and filter flights based on preferences.
  - View flight details including price, duration, and airline.
  - Seat selection with an interactive map.
  - Secure online payment and instant e-ticket generation.

3.



1.Front-end:

- Built using React.js for a dynamic and responsive user interface.
- Components for user authentication, flight search, booking, and seat selection.
- State management with Context API.

2.Back-end:

- Node.js with Express.js powers the server-side logic.
- RESTful API endpoints handle user authentication, flight searches, bookings, and admin operations.

3.Database:

- MongoDB stores collections for users, flights, and bookings.
- Mongoose used for schema definitions and CRUD operations.

4.Third-Party Integrations

The app relies on external services to fetch flight data, process payments, and send notifications.

Key Features

Flight Data Providers: Amadeus, Sabre, Travelport, or other GDS (Global Distribution Systems)
Payment Gateways: PayPal, Stripe, Razorpay
Notification Services: Twilio (SMS), Firebase (Push), SendGrid (Email)
Geolocation Services: Google Maps API for airport locations

5.Middleware

Middleware acts as a bridge between the front-end, back-end, and third-party APIs.
Key Features
API Gateway: Manages incoming API requests and routes them appropriately.
Authentication Middleware: Validates user sessions (JWT, OAuth2). Rate Limiting: Protects APIs from abuse.

Technologies

API Gateway: Kong, Amazon API Gateway Authentication:
OAuth, Firebase Authentication, Keycloak

6.Security Layer

Flight booking apps handle sensitive data, so security is critical.
Key Features
Data encryption (SSL/TLS)
Secure user authentication
PCI-DSS compliance for payment
DDoS protection
OWASP practices for web/mobile apps

Technologies
Encryption: AES for sensitive data, SSL/TLS for communications
Firewalls: WAF (Web Application Firewall)

7.DevOps & Deployment
Continuous deployment and monitoring ensure the app is reliable and scalable.
Key Features
Automated CI/CD pipelines
Load balancing for traffic management
Server monitoring and logging

Technologies
CI/CD Jenkins, GitHub Actions
Cloud Hosting: AWS, Google Cloud, Azure
Containerization: Docker, Kubernetes
Monitoring: Prometheus, Grafana, New Relic

8.Scalability
The architecture should handle growth in user base and traffic.

Techniques
Load balancing using Nginx or AWS Elastic Load Balancer
Horizontal scaling with Kubernetes
Distributed databases for global availability

High-Level Architecture Diagram
1. Client Layer: Web App & Mobile App

2. API Gateway: Routes requests to appropriate microservices.

3. Microservices: Handles flight search, booking, payment, and notifications.

4. Third-Party Services: Connects to GDS, payment gateways, and notification APIs.

5. Data Layer: Manages persistent storage with relational/NoSQL databases.

6. DevOps: Ensures scalability, monitoring, and fault tolerance.

4. Setup Instructions

- **Prerequisites:**
  - Node.js
  - MongoDB
  - Git
- 1. Setting Up an Existing Flight Booking App
  If you’ve purchased or downloaded a pre-built flight booking app, follow these steps to configure it:

  a. Installation and Deployment
  1. Mobile Apps:
  For user-facing apps, download the app from Google Play Store or Apple App Store.
  If it's a white-label app, download the app's source code (if provided) and build it using Android Studio (for Android) or Xcode (for iOS□.

  2. Admin Panel:
  If the app includes an admin dashboard, deploy it on your web server. Use hosting providers like AWS, Google Cloud, or Heroku, depending on the app's requirements.

  3. Database Setup:
  Import the database schema provided with the app into a database management system (e.g., MySQL, PostgreSQL□.
  Set up database credentials in the app's configuration files.

---

b. Configuration

1. Customize Branding:

Change the app's logo, color scheme, and name in the source code or admin panel.

Update any splash screens or icons.

2. Set Up API Keys:

Integrate flight search and booking APIs, such as:

Amadeus: For global flight availability, prices, and bookings.

Sabre: Offers advanced booking systems with ancillary services.

Travelport: A comprehensive travel management API.

Obtain API keys from the provider and add them to the configuration files.

3. Payment Gateway Integration:

Configure payment systems like PayPal, Stripe, or Razorpay.

Enter API keys or credentials in the app's admin panel.

4. Email and SMS Configuration:

Integrate services like SendGrid (for emails) and Twilio (for SMS notifications).

Update the credentials in your app settings.

5. Localization:

Add support for multiple languages and currencies to cater to different regions.

Configure these settings via the admin dashboard or source code.

---

c. Testing

Test flight search, booking, and payment functionalities thoroughly.
Verify integration with third-party APIs and ensure they return accurate results.
Simulate various scenarios, such as cancellations or failed transactions, to ensure reliability.

---

d. Launch

Publish the mobile apps on the App Store and Google Play Store. Deploy the admin dashboard and ensure the website (if any) is accessible.

---

2. Building a Flight Booking App From Scratch

If you're creating the app from the ground up, follow these steps:

---

a. Plan the App Architecture

1. Define Core Features:

User registration and profiles.
Flight search and filtering by price, date, or airlines.
Real-time seat availability and ticket booking.
Payment and invoice generation.
Notifications for booking confirmation and updates.

2. Choose the Technology Stack:

Frontend: React Native, Flutter, or native languages like Swift (iOS) and Kotlin Android).
Backend: Node.js, Django, or Ruby on Rails.
Database: MySQL, PostgreSQL, or MongoDB.

---

b. Develop the App

1. Frontend:

Design an intuitive user interface using tools like Figma or Adobe XD.
Implement the design using the chosen frontend technology.

2. Backend:
Create APIs for user authentication, flight search, booking, and payments.
Set up server infrastructure using cloud providers like AWS or Azure.

3. Database:

Design tables for users, flights, bookings, and payments.
Use relational databases for structured data.

---

c. Integrate APIs
1. Flight Search and Booking:
Partner with API providers like Amadeus, Sabre, or Travelport. Use their documentation to integrate features like flight availability, ticket pricing, and booking confirmation.

2. Payment Gateways:
Integrate Stripe, PayPal, or other gateways for secure transactions.

3. Notifications:
Add real-time notifications using Firebase Cloud Messaging (FCM) or Twilio.

---

d. Testing and Deployment
1. Testing:

Conduct unit, integration, and system testing.
Simulate high-traffic scenarios to ensure scalability.

2. Deployment:
Publish mobile apps to the app stores.
Host the backend on a secure and scalable server.

---
e. Maintenance
1. Monitor app performance using tools like Google Analytics or New Relic.

2. Regularly update the app with new features and bug fixes.

Final Tip:
For faster setup, you can use app builders or templates available on platforms like CodeCanyon or BuildFire. However, ensure to customize these solutions to meet your brand’s specific requirements.

- **Installation:**

  Clone the r epository:

  ```
  bash
  Copy code
  git clone https://github.com/harsha-vardhan-reddy-07/Fl ight-Booking-App-MERN
  ```

  Navigate to the project directory:

  ```
  bash
  Copy code
  cd Flight-Booking-App-MERN
  ```

  Inst all dependencies:

```
bash
Copy code
npm install
```

Start the development server:

```
bash
Copy code
npm run dev
```

---

5. Folder Structure

- **Client:**
  - `src` Contains all React components, pages, and styles.
  - `components` Reusable UI components.
  - `pages` Individual pages like home, search results, and booking details.
- **Server:**
  - `routes` Defines API routes for users, flights, and bookings.
  - `controllers` Business logic for handling API requests.
  - `models` MongoDB schema definitions for users, flights, and bookings.
  -

General F older Structure

Root Directory

project_root/ ├── backend/ # Backend services ├── frontend/ # Mobile/Web front-end ├── shared/ # Shared resources (configs, assets) ├── docs/ # Documentation ├── tests/ # Test cases for different modules ├── scripts/ # Deployment or utility scripts ├── README.md # Project overview └── package.json # Dependencies (for

JS-based apps)

---
2. Detailed Structure
A. Backend Folder
For managing server-side logic, APIs, and databases.
backend/ ├── src/ │ ├── controllers/ # Business logic for APIs │ ├── models/ # Database schemas (e.g., flights, users) │ ├── routes/ # API endpoints │ ├── services/ # Third-party integrations (e.g., APIs, payment) │ ├── middlewares/ # Middleware for authentication, validation │ ├── config/ # Configuration files (e.g., environment variables) │ ├── database/ # Database connection and migrations │ └── utils/ # Utility functions/helpers ├── tests/ # Unit tests for backend ├── package.json # Node.js dependencies └── server.js # Entry point of the backend
B. Frontend Folder
For mobile or web front-end implementation.
React Native/Flutter Example
frontend/ ├── src/ │ ├── assets/ # Images, fonts, icons, etc. │ ├── components/ # Reusable components (e.g., buttons, modals) │ ├── screens/ # Screens for each feature │ │ ├── Authentication/ │ │ ├── FlightSearch/ │ │ ├── Booking/ │ │ ├── Profile/ │ │ └── Support/ │ ├── navigation/ # Navigation logic (e.g., stack, drawer) │ ├── services/ # API integration (e.g., flight search, payments) │ ├── context/ # State management (e.g., Redux/Context API/Provider) │ ├── hooks/ # Custom React hooks │ ├── styles/ # Global stylesheets │ ├── utils/ # Utility functions (e.g., date formatter) │ └── app.js # Main entry point ├── android/ # Android-specific files (React Native/Flutter) ├── ios/ # iOS-specific files (React Native/Flutter) └── package.json # Dependencies
Web Example
frontend/ ├── public/ # Static files (e.g., index.html, favicon) ├── src/ │ ├── components/ # Reusable UI components │ ├── pages/ # Pages for routing │ ├── services/ # API calls │ ├── context/ # Global state management │ ├── styles/ # CSS/SCSS files │ ├── utils/ # Helper functions │ ├── App.js # Root React component │ └──

index.js   # Entry point ├── tests/   # Frontend unit/integration tests └── package.json   # Dependencies

---
C. Shared Resources
For files used by both frontend and backend.
shared/ ├── config/ │   ├── apiConfig.js   # API keys, URLs │   ├── paymentConfig.js   # Payment gateway settings │   ├── env.js   # Environment variables ├── assets/ │   ├── icons/   # Icons used in both layers │   ├── translations/   # Localization files (e.g., JSON) │   └── themes/   # Themes (e.g., dark/light modes) ├── validators/   # Shared validation logic (e.g., input validation) └── utils/   # Common utilities (e.g., logging, error handling)

D. Documentation Folder
Contains project documentation, API references, and more.
docs/ ├── API.md   # API reference ├── README.md   # General instructions ├── SETUP.md   # Setup instructions └── architecture.png   # System architecture diagram

---
E. Tests Folder
Centralized testing for frontend and backend.
tests/ ├── backend/   # Backend-specific test cases ├── frontend/   # Frontend-specific test cases ├── integration/   # End-to-end test cases └── config/   # Testing utilities and configurations

---
F. Deployment and Scripts
Automates deployment, backups, or utility tasks.
scripts/ ├── deploy.sh   # Script for deployment ├── backup.sh   # Backup script for database └── lint-fix.sh   # Linting and formatting script

---

3. Key Considerations

Modularity: Group related functionality in separate modules for scalability.
Reusability: Use shared folders for common resources. Environment-Specific
Configuration: Use .env files to manage environment variables.
Testing: Maintain a clear folder structure for tests to ensure maintainability.

This folder structure ensures scalability, modularity, and maintainability for a flight booking app. Adjust as needed based on your specific requirements.

---

6. Running the Application

- **Frontend:**

```
bash Copy
code cd
client npm
start
```

- **Backend:**

```
bash Copy
code cd
server npm
start
```

---

7. API Documentation

- **GET** `/api/flights` : Retrieve available flights.
  - Parameters: Departure, Destination, Date.
  - Example Response:

json

Copy code

```json
{ "flights": [ { "id": 1, "price": 500, "airline": "Air ways" } ] }
```

- **POST** `/api/bookings` Book a flight.
  - Body Parameters: User ID, Flight ID, Seat Details.
  - Example Response:

    json

    Copy code

    ```json
    { "message": "Booking successful", "bookingId": 12345 }
    ```

---

8. Authentication

- **Method:**
  - Token-based authentication using JSON Web Tokens JWT.
  - Secure user sessions for login and booking functionalities.

---

9. User Interface

- **Highlights:**
  - Modern design with intuitive navigation.
  - Interactive flight and seat selection.
  - Screenshots:
    - Home Page
    - Flight Search Results
    - Seat Selection

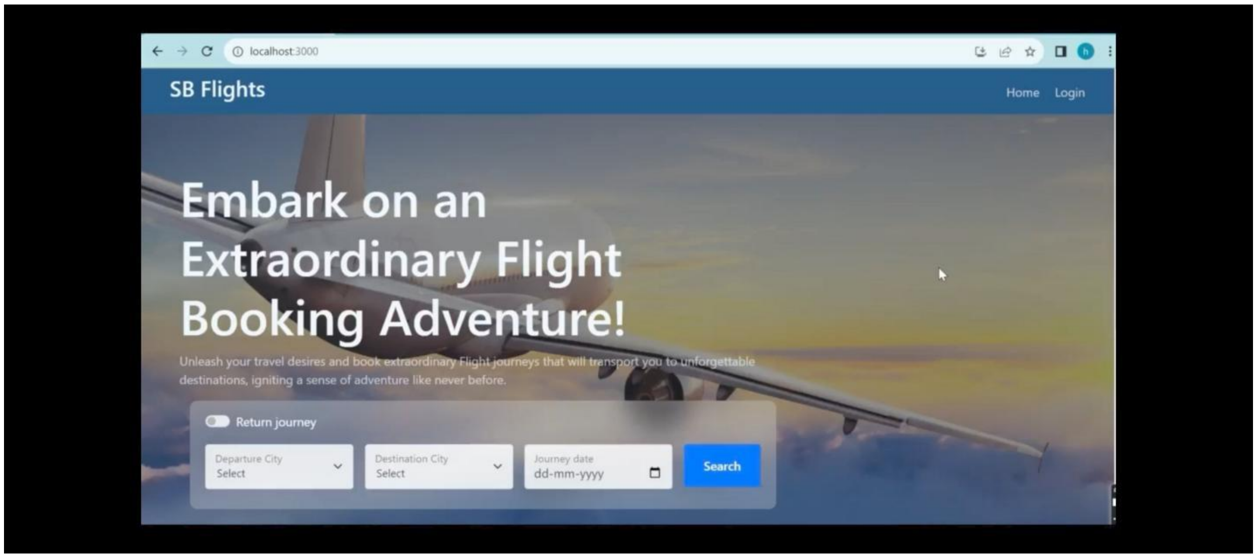
---

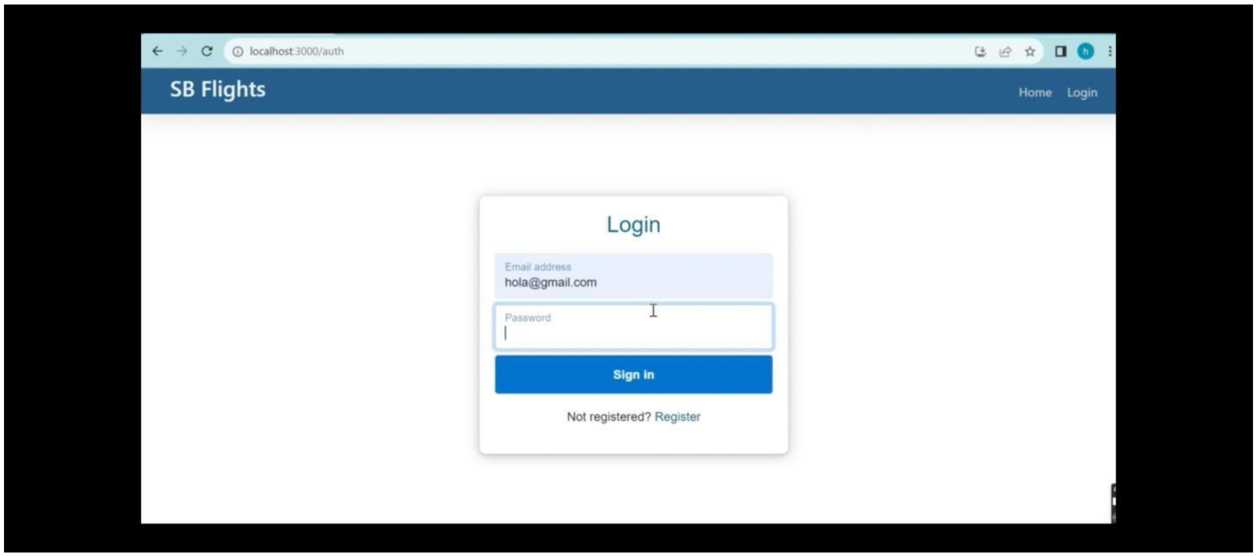
10. Testing

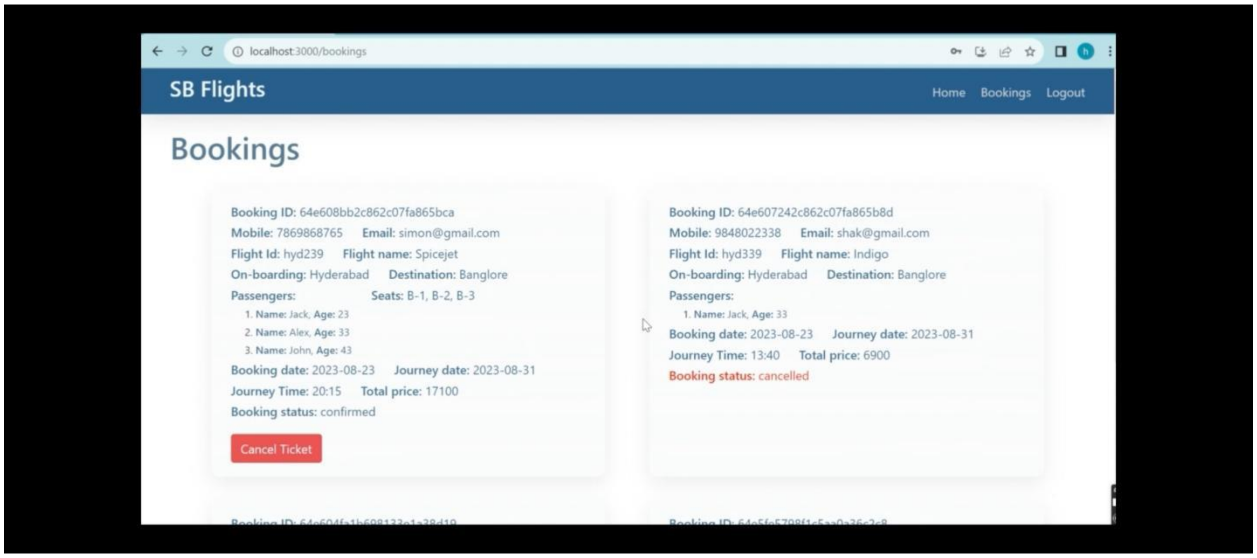
- **Strategy:**
    - Unit tests using Jest for individual components.
    - End-to-end tests using Cypress to verify complete user journeys.

---

11. Screenshots


localhost:3000
SB Flights
Home
Login
Embark on an Extraordinary Flight Booking Adventure!
Unleash your travel desires and book extraordinary Flight journeys that will transport you to unforgettable destinations, igniting a sense of adventure like never before.
Return journey
Departure City
Select
Destination City
Select
Journey date
dd-mm-yyyy
Search





localhost:3000/bookings

SB Flights — Home Bookings Logout

Bookings

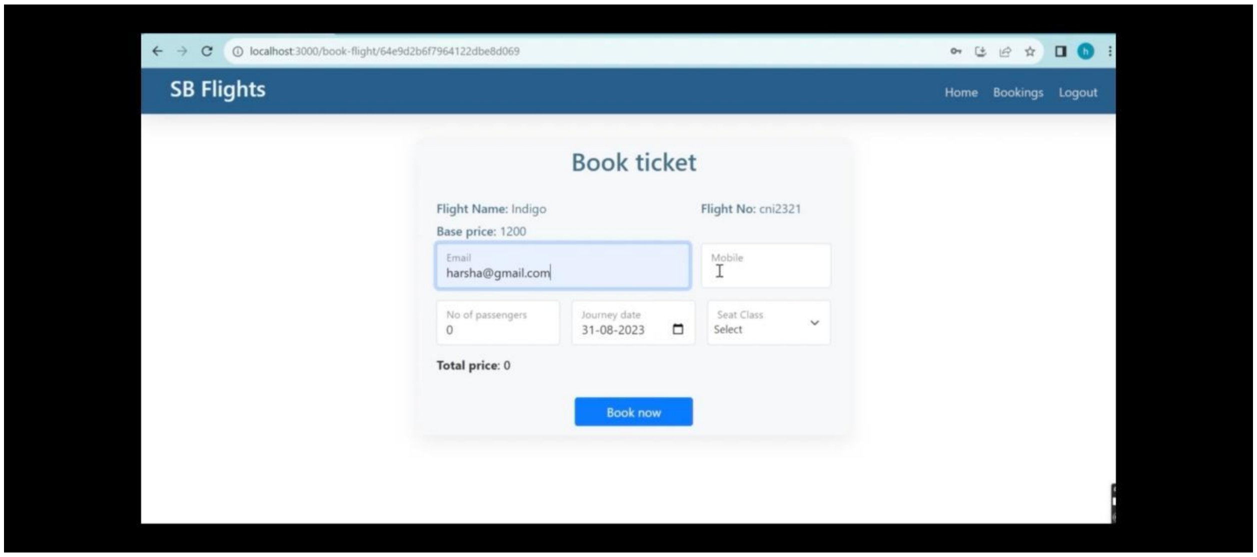
Booking ID: 64e608bb2c862c07fa865bca
Mobile: 7869868765 **Email:** simon@gmail.com
Flight Id: hyd239 **Flight name:** Spicejet
On-boarding: Hyderabad **Destination:** Banglore
Passengers: **Seats:** B-1, B-2, B-3
1. **Name:** Jack, **Age:** 23
2. **Name:** Alex, **Age:** 33
3. **Name:** John, **Age:** 43

Booking date: 2023-08-23 **Journey date:** 2023-08-31
Journey Time: 20:15 **Total price:** 17100
Booking status: confirmed

Cancel Ticket

Booking ID: 64e607242c862c07fa865b8d
Mobile: 9848022338 **Email:** shak@gmail.com
Flight Id: hyd339 **Flight name:** Indigo
On-boarding: Hyderabad **Destination:** Banglore
Passengers:
1. **Name:** Jack, **Age:** 33

Booking date: 2023-08-23 **Journey date:** 2023-08-31
Journey Time: 13:40 **Total price:** 6900
Booking status: cancelled



localhost:3000/book-flight/64e9d2b6f7964122dbe8d069

SB Flights — Home Bookings Logout

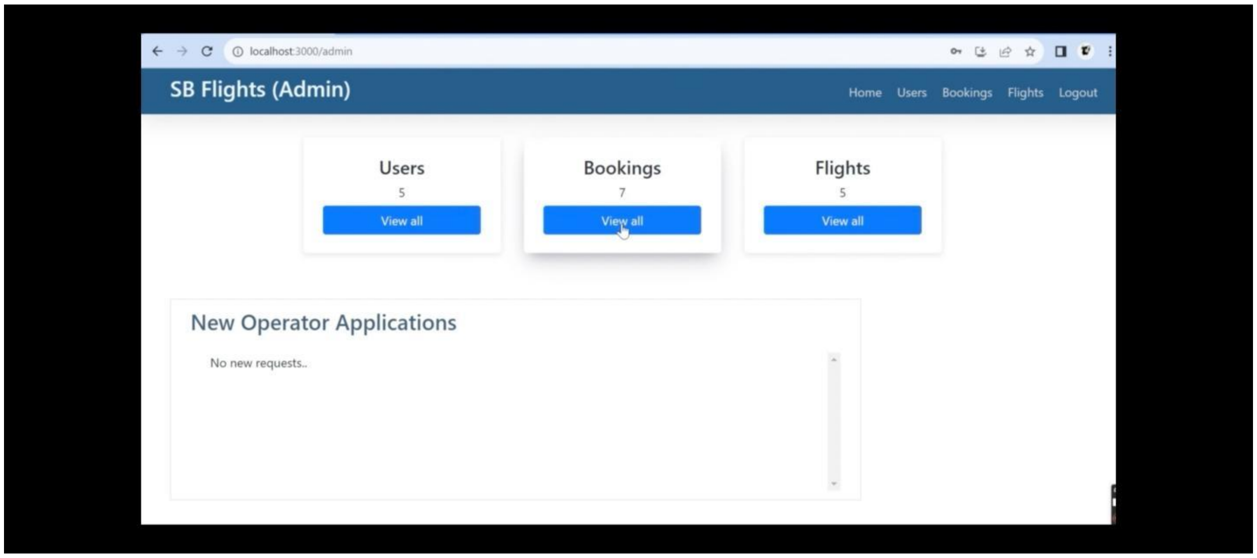
Book ticket

Flight Name: Indigo **Flight No:** cni2321
Base price: 1200

Email: harsha@gmail.com
Mobile:
No of passengers: 0
Journey date: 31-08-2023
Seat Class: Select

Total price: 0

Book now



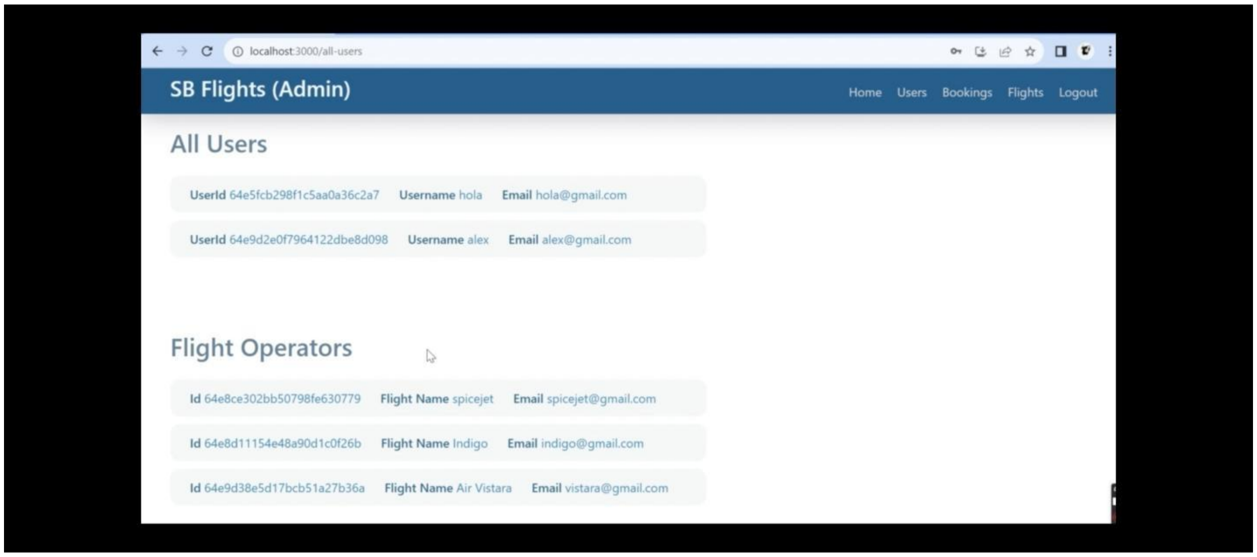
localhost:3000/admin

SB Flights (Admin) — Home Users Bookings Flights Logout

Users	Bookings	Flights
5	7	5
View all	View all	View all

New Operator Applications

No new requests..



SB Flights (Admin)

All Users

UserId 64e5fcb298f1c5aa0a36c2a7 **Username** hola **Email** hola@gmail.com

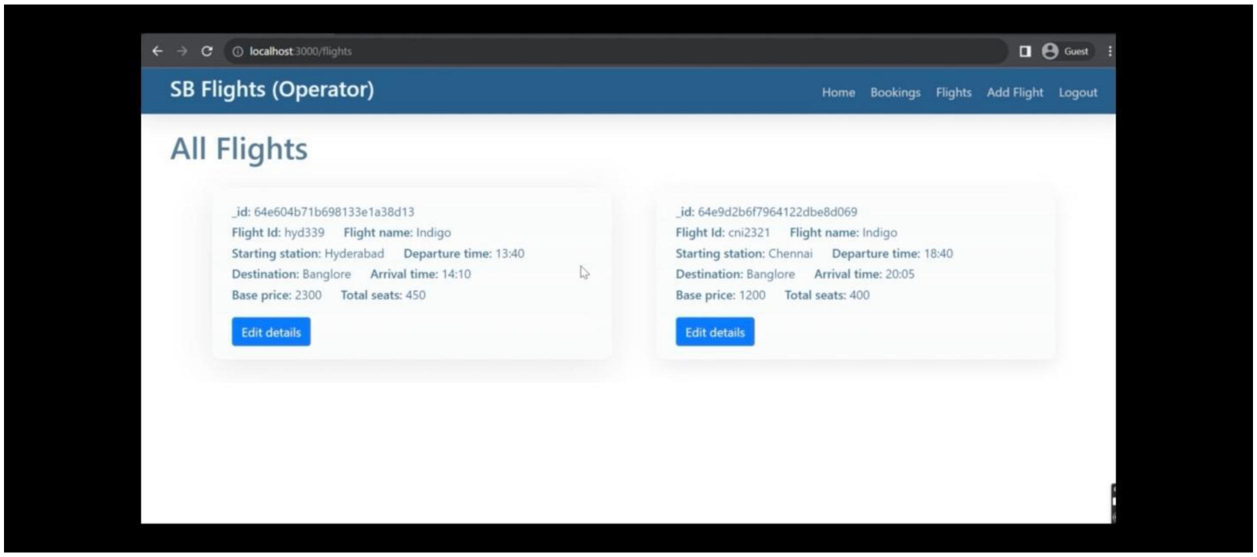
UserId 64e9d2e0f7964122dbe8d098 **Username** alex **Email** alex@gmail.com

Flight Operators

Id 64e8ce302bb50798fe630779 **Flight Name** spicejet **Email** spicejet@gmail.com

Id 64e8d11154e48a90d1c0f26b **Flight Name** Indigo **Email** indigo@gmail.com

Id 64e9d38e5d17bcb51a27b36a **Flight Name** Air Vistara **Email** vistara@gmail.com

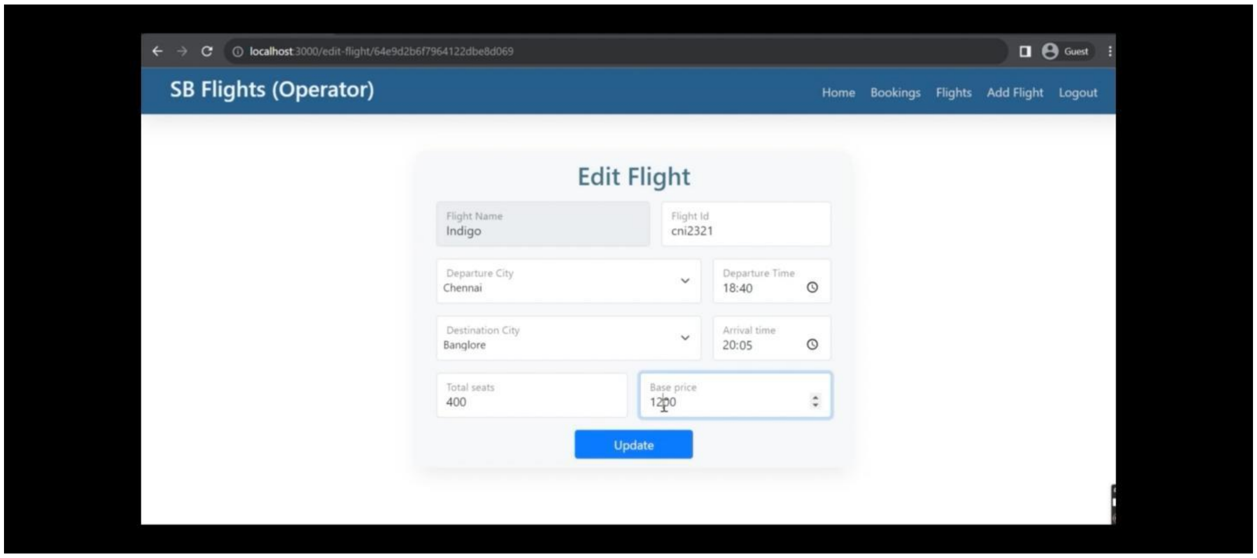


SB Flights (Operator)

All Flights

_id: 64e604b71b698133e1a38d13
Flight Id: hyd339 **Flight name:** Indigo
Starting station: Hyderabad **Departure time:** 13:40
Destination: Banglore **Arrival time:** 14:10
Base price: 2300 **Total seats:** 450

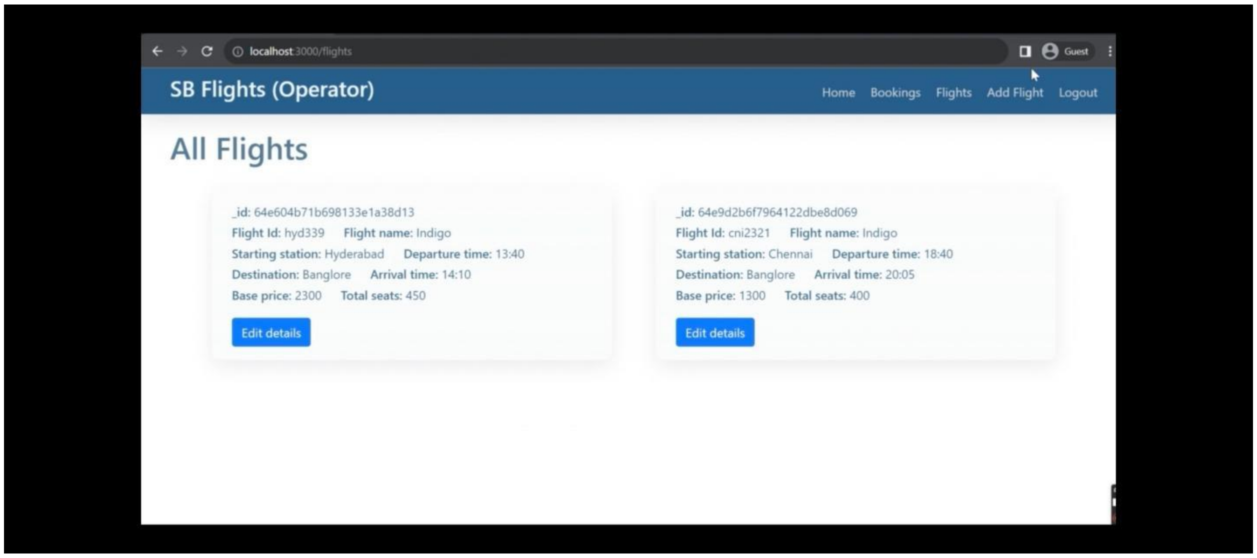
_id: 64e9d2b6f7964122dbe8d069
Flight Id: cni2321 **Flight name:** Indigo
Starting station: Chennai **Departure time:** 18:40
Destination: Banglore **Arrival time:** 20:05
Base price: 1200 **Total seats:** 400



SB Flights (Operator)

Edit Flight

Flight Name: Indigo
Flight Id: cni2321
Departure City: Chennai
Departure Time: 18:40
Destination City: Banglore
Arrival time: 20:05
Total seats: 400
Base price: 1200


localhost:3000/flights

SB Flights (Operator)

Home Bookings Flights Add Flight Logout

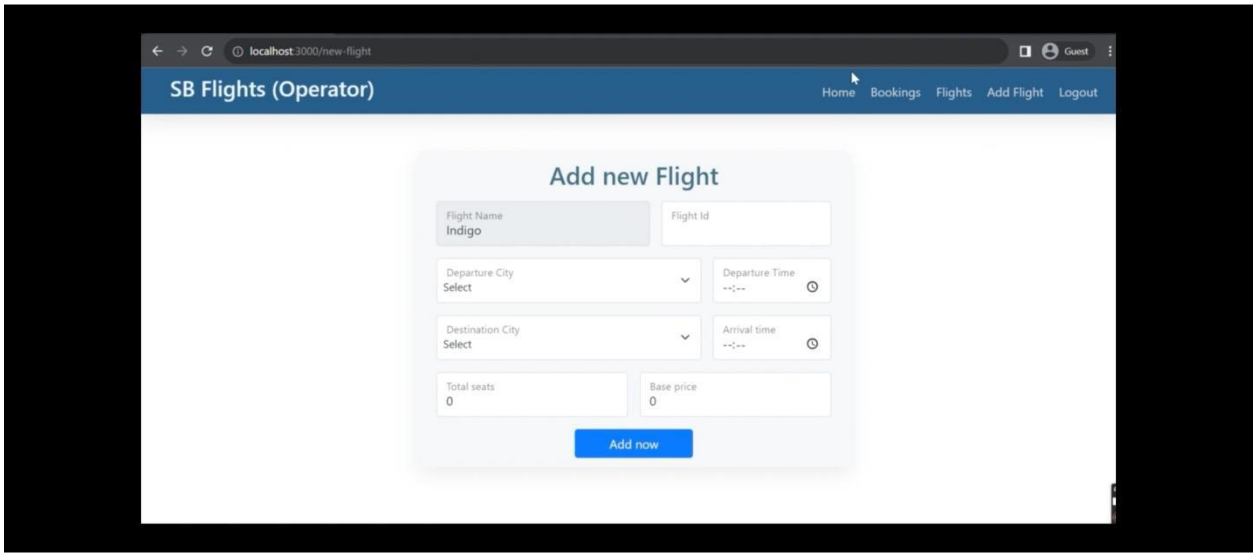
All Flights

_id: 64e604b71b698133e1a38d13
Flight Id: hyd339 **Flight name:** Indigo
Starting station: Hyderabad **Departure time:** 13:40
Destination: Banglore **Arrival time:** 14:10
Base price: 2300 **Total seats:** 450

Edit details

_id: 64e9d2b6f7964122dbe8d069
Flight Id: cni2321 **Flight name:** Indigo
Starting station: Chennai **Departure time:** 18:40
Destination: Banglore **Arrival time:** 20:05
Base price: 1300 **Total seats:** 400

Edit details


localhost:3000/new-flight

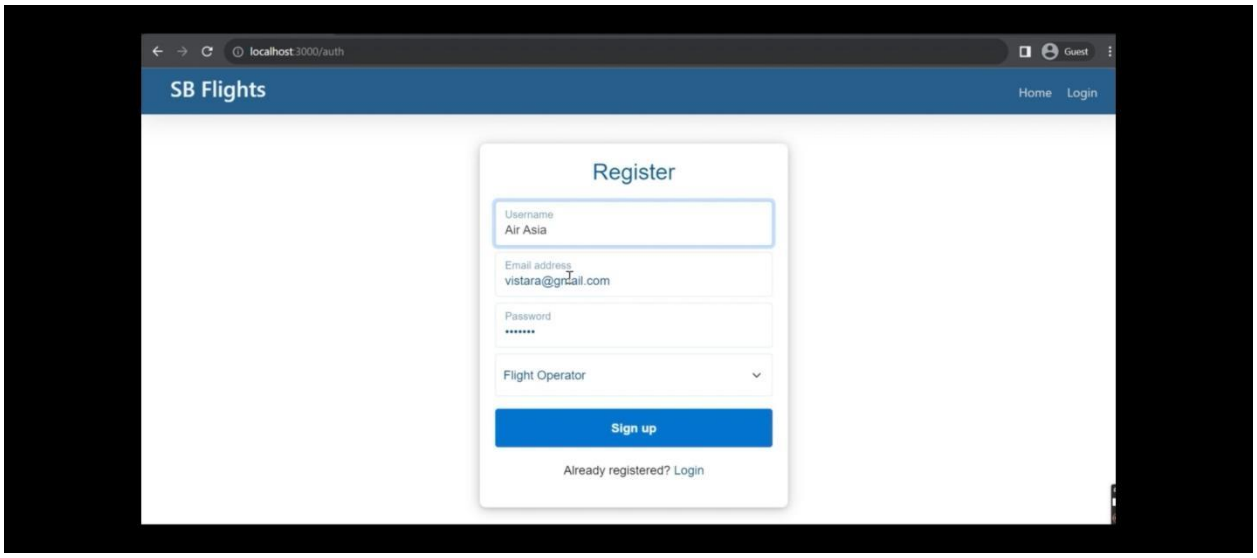
SB Flights (Operator)

Home Bookings Flights Add Flight Logout

Add new Flight

Flight Name: Indigo
Flight Id:
Departure City: Select
Departure Time: --:--
Destination City: Select
Arrival time: --:--
Total seats: 0
Base price: 0

Add now


localhost:3000/auth

SB Flights

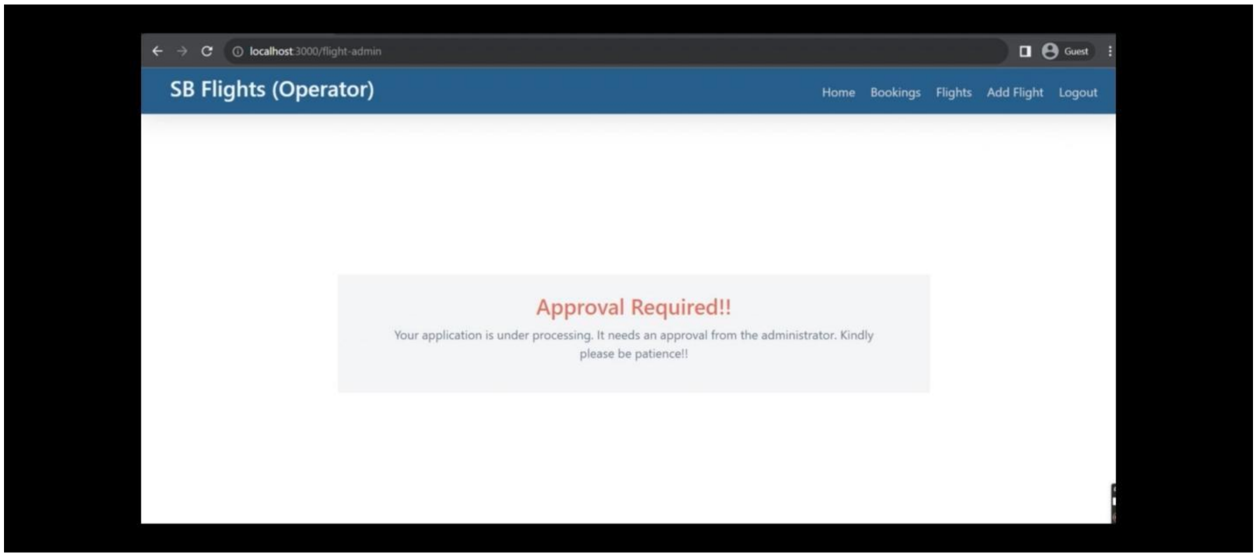
Home Login

Register

Username: Air Asia
Email address: vistara@gmail.com
Password: •••••••
Flight Operator

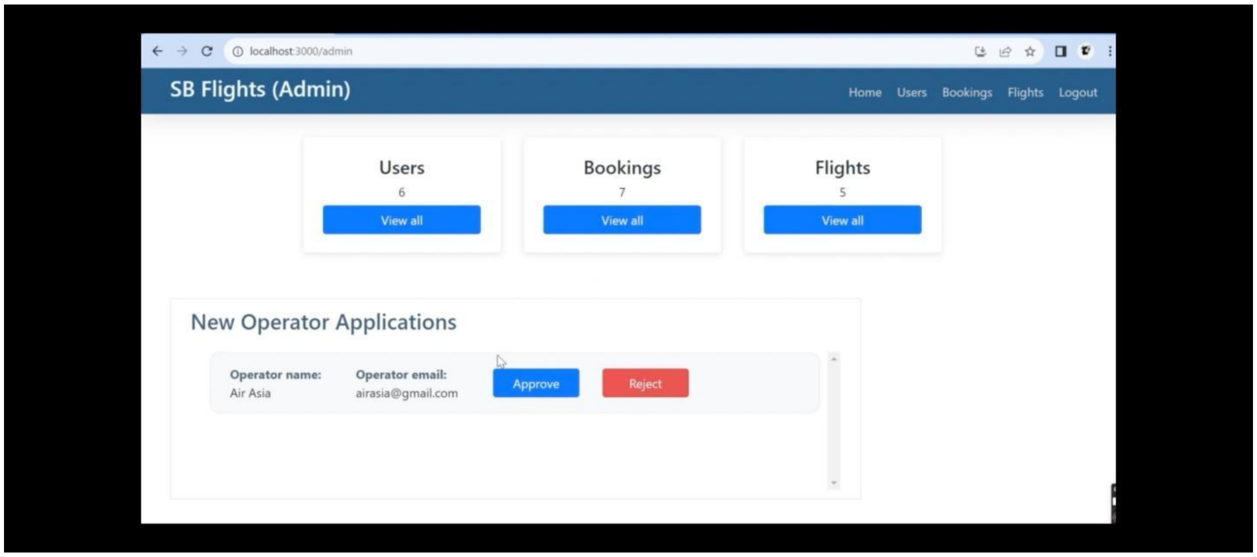
Sign up

Already registered? Login



Approval Required!!

Your application is under processing. It needs an approval from the administrator. Kindly please be patience!!



New Operator Applications

Operator name: Air Asia
Operator email: airasia@gmail.com

---

12. Known Issues

- Occasional delay in fetching flights due to network latency.
- Payment gateway integration could be more seamless.

---

13. Future Enhancements

Future enhancements for a flight booking app can improve the user experience, expand functionality, and stay competitive in the travel industry. Below are detailed future enhancement ideas categorized into different aspects of the app:

---

1. User Experience UX Enhancements

a. Personalization

AIPowered Recommendations:

Suggest flights based on user search history, preferences, and past bookings. Use machine learning to analyze trends and offer tailored suggestions for destinations, airlines, and travel times.

Dynamic Pricing Alerts:

Notify users of fare changes for their saved searches or preferred routes.

Loyalty Rewards:

Introduce points systems for frequent users that can be redeemed for discounts or upgrades.

b. Enhanced Search Features

Multi-City and Multi-Airline Searches:

Allow users to book trips involving multiple stops or different airlines in one search.

Flexible Date Searches:

Show the cheapest flights for a range of dates.

Voice Search Integration:

Enable users to search for flights using voice commands.

c. Booking Process Improvements

One-Click Checkout:

Save user payment details securely for faster booking.

Seat Selection Visualization:

Provide a live, interactive seat map for better seat choices.

Bundle Deals:

Offer packages combining flights, hotels, and car rentals.

---

2. Payment and Security Enhancements
a. Payment Options
Installment Plans:
Allow users to pay for flights in installments.

Cryptocurrency Support:

Accept payments via popular cryptocurrencies like Bitcoin or Ethereum.

Region-Specific Payment Gateways:
Integrate local payment systems like UPI □India), WeChat Pay □China), etc.

b. Security Features
Biometric Authentication:

Enable fingerprint or facial recognition for login and payment authentication.

Fraud Detection Systems:
Implement real-time fraud detection using AI to monitor suspicious activity.

---

3. Real-Time Data Integration
a. Flight Status Tracking
Provide real-time updates on flight delays, cancellations, and gate changes.

b. Dynamic Pricing Insights
Show pricing trends and predictions to help users book at the best time.

c. Weather Integration
Include destination weather forecasts during the booking process to help users plan better.

4. Social and Collaborative Features

a. Group Bookings

Allow multiple users to book tickets for the same trip and split payments.

Show synchronized seat selections for group members.

b. Social Sharing

Enable users to share itineraries, wishlists, or trip plans on social media or with

friends.

c. Community-Based Features

User Reviews:

Add reviews and ratings for airlines, airports, and destinations.

Travel Blogs and Guides:

Provide user-generated content and recommendations for destinations.

---

5. Advanced AI and Automation

a. Chatbots

Use AI-powered chatbots to handle customer inquiries, cancellations, or rebookings instantly.

Offer multilingual support to cater to global users.

b. Travel Assistance

AI travel assistants can help plan itineraries, suggest activities, and provide travel tips.

Offer voice-enabled personal assistants to guide users through the app.

6. New Travel Options

a. Multi-Modal Travel

Include alternative modes of transport (e.g., trains, buses, ferries) for a seamless door-to-door travel experience.

b. Subscription Plans
Introduce travel subscriptions for frequent flyers, offering discounted rates or unlimited travel on selected routes.

7. Sustainability and Social Responsibility
a. Carbon Emission Tracking
Show the carbon footprint of each flight and suggest greener alternatives.
Allow users to offset their carbon emissions directly through the app.

b. Promote Sustainable Airlines
Highlight airlines and flights with environmentally friendly practices.

8. Offline and Accessibility Features
a. Offline Access
Allow users to access their itineraries, e-tickets, and booking details without an internet connection.

b. Accessibility Options
Make the app more inclusive by offering features like voice descriptions, larger font sizes, and simplified navigation for visually impaired users.

9. Integration with Other Platforms
a. Calendar Sync
Allow users to add flight schedules directly to their calendars.

b. Wearable Devices
Integrate with smartwatches to show flight updates and boarding information.

c. Third-Party Services
Collaborate with apps like Google Maps for navigation to the airport or Uber for ride-hailing services.

10. Marketing and Loyalty Enhancements
a. Referral Programs

Introduce rewards for users who refer new customers.

b. Seasonal Offers
Provide discounts for specific seasons, events, or holidays.

c. Push Notifications
Use location-based notifications to alert users about exclusive deals or airport-specific services.

11. Regulatory and Legal Enhancements
a. Insurance Integration
Offer travel insurance during the booking process.

b. Legal Compliance
Ensure compliance with GDPR, CCPA, and other global data protection laws.

12. Gamification
Add interactive features like milestones or badges for frequent travelers.
Provide leaderboards for top travelers to engage users.

These enhancements can help create a robust, feature-rich flight booking app that offers a seamless, engaging, and secure travel experience for users.