SkyWings: An Efficient Airline Reservation System

Team Members:

• Simiyon Vinscent Samuel L

• Sundareswaran R

• Vijayaraaghavan K S

Guide:

• Mrs. Vijayalakshmi A

Institution:

Department of Computer Science and Engineering, SSN College of Engineering

Date:

5th May 2025

Table of Contents

Project Overview

Problem Statement

Traditional airline reservation systems are often cumbersome, lacking user personalization, real-time updates, and intelligent automation. These limitations result in a poor user experience and operational inefficiencies.

Objective

To develop a full-stack web-based airline reservation system that automates flight booking, seat allocation, and user management, with added features like dynamic pricing, analytics, and chatbot support.

Project Scope

The SkyWings system enables:

- User registration, authentication, and profile management
- Flight search, dynamic booking, and cancellation
- Automated seat allocation and overbooking handling
- Admin dashboard for managing flights and schedules
- AI chatbot for user assistance
- Analytical reports using dynamic visualizations

This system is intended for use by small to medium-sized airline operators.

Existing System

Airlines typically use legacy software or third-party booking engines with limited automation and minimal user experience optimization. These systems lack features like real-time pricing, smart seat selection, and flexible booking flows.

Overview of Existing Solutions

Many airline companies use platforms like Amadeus, Sabre, or Galileo. These are expensive, complex to integrate, and not customizable for smaller operators. Open-source options are limited in modularity and often lack support for AI or analytics.

Limitations of the Current Systems

- High integration and license costs
- No open-source flexibility for enhancements
- Poor user interface and user experience
- No dynamic pricing or real-time updates
- Limited administrative control for small airline owners

Need for the Proposed Web Application

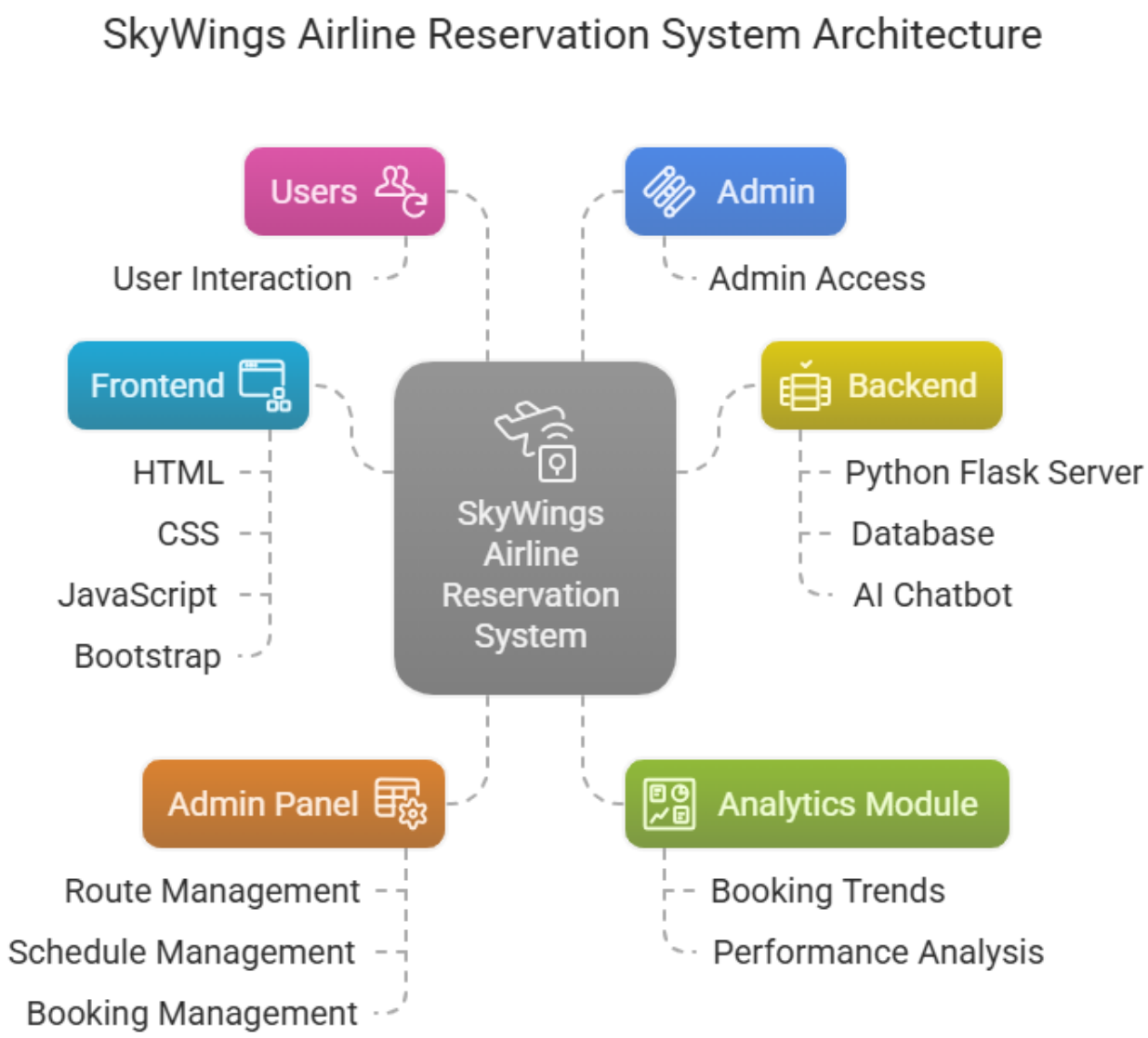
To offer an open-source, scalable, and AI-assisted booking platform that enhances customer convenience, reduces manual errors, and empowers administrators with real-time insights and control.

Proposed System

System Overview

SkyWings is a modular and responsive web application that automates airline reservation processes, integrating backend logic for dynamic seat allocation, pricing algorithms, and a custom-built AI chatbot for user support.

Diagram:



Key Features and Advantages

- Smart seat allocation and dynamic pricing
- Admin dashboard for route and schedule control
- Chatbot support using NLP
- Loyalty rewards module
- Integrated analytics dashboard
- Live weather-based flight scheduling
- Email and QR code for ticket generation

System Workflow

1. User registers/logs in
2. User searches for flights
3. Seats displayed with dynamic pricing
4. User books and pays
5. Seat allocation confirmed
6. Admin manages schedules and data
7. AI chatbot assists users throughout

Diagram:

Module-Wise Explanation and Workflow Diagrams

1. User Module

Overview:
This module manages all interactions related to user identity—account creation, authentication, session tracking, and personalized features. It's the gateway to accessing services like booking flights or managing itineraries.

Why It Matters:
A secure, reliable user system is essential to prevent unauthorized access, protect user data, and maintain booking integrity. It sets the foundation for a personalized experience.

Key Features:

- User registration with email, username, and password.
- Login authentication with session management using Flask.
- Password handling (can include hashing for production-grade security).
- Profile view showing booking history.

Technical Workflow:

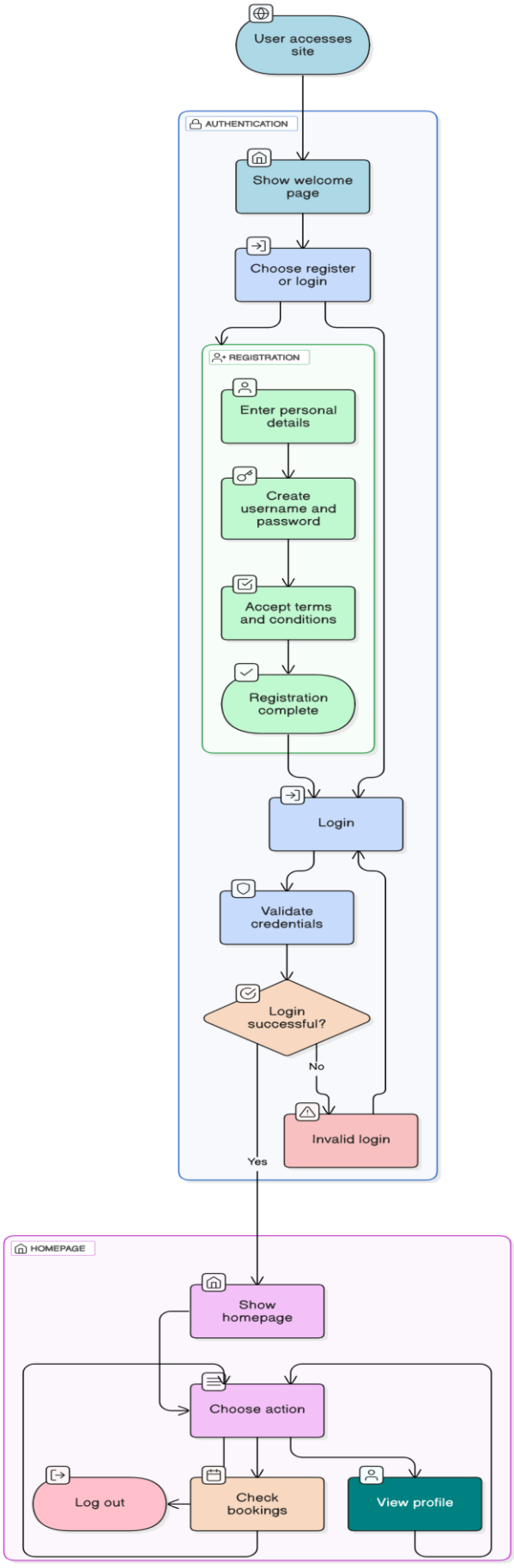
1. User accesses the registration or login page.
2. On registration:
    - Input is validated for format and completeness.
    - Data is stored in the users table.
3. On login:
    - Credentials are matched against database entries.
    - A session variable is initiated to persist login state.
4. Once logged in:
    - The user is redirected to the homepage with access to features like booking, profile view, and logout.

Future Enhancements:

- Password encryption using bcrypt or SHA-256
- OTP or two-factor authentication
- OAuth login integration (Google/Facebook)

Diagram:

User Module Flow in Airline Reservation System



2. Flight Search & Booking Module

Overview:
This module handles the flight discovery and initial booking phase. Users can input search parameters to retrieve a list of available flights from the database.

Why It Matters:
It's the primary functional interface for users to find and compare flight options. It determines user satisfaction and efficiency in discovering relevant options.

Key Features:

- Search by source, destination, and date.
- Filtered SQL queries return matching flights.
- Results display includes departure, arrival, fare, and availability.
- Selection of preferred flight for booking continuation.

Technical Workflow:

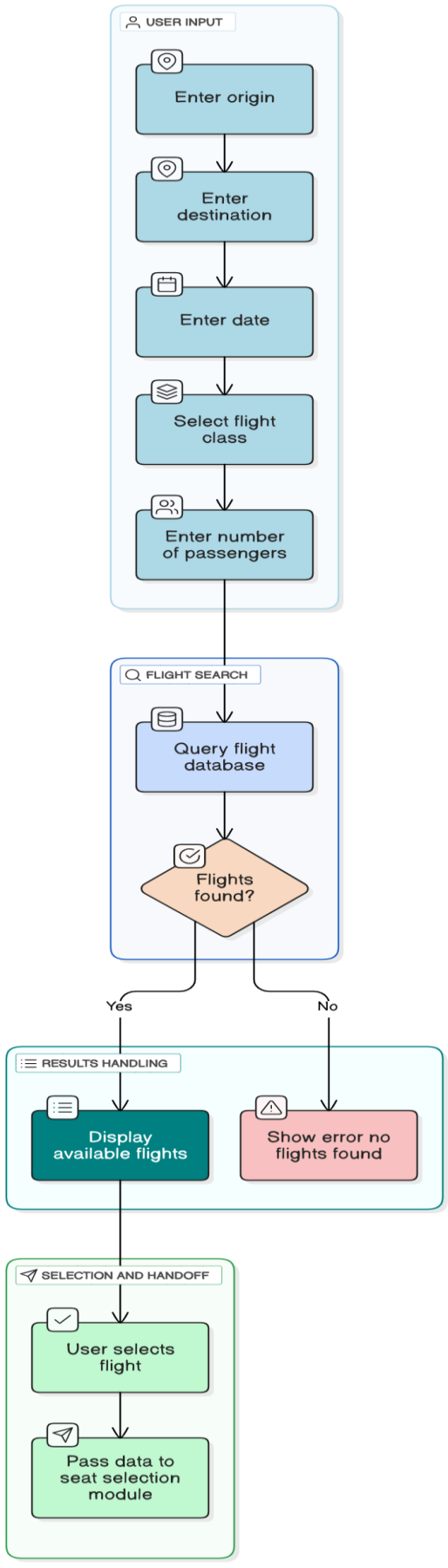
1. User submits search criteria via form.
2. Flask backend processes the input and queries the flights table.
3. Matching results are displayed using HTML templates.
4. User selects a flight, and details are stored in session/local state for next steps.

Future Enhancements:

- Multi-city or round-trip options
- Live seat availability updates
- Integration with real-time flight APIs (for advanced deployment)

Diagram:

Flight Search and Booking Module Flow



3. Seat Allocation Module

Overview:
This module provides a visual and logical system for users to select their preferred seat. It ensures each seat is booked once and integrates pricing based on seat class or availability.

Why It Matters:
Seat selection is a user-expectation standard. It also supports dynamic pricing models, which are industry best practices for revenue optimization.

Key Features:

- Dynamic seat map rendered in the front-end (A/B/C rows)
- Real-time locking to prevent double booking
- Seat preferences like window, middle, aisle
- Pricing modifiers based on seat position or occupancy rate

Technical Workflow:

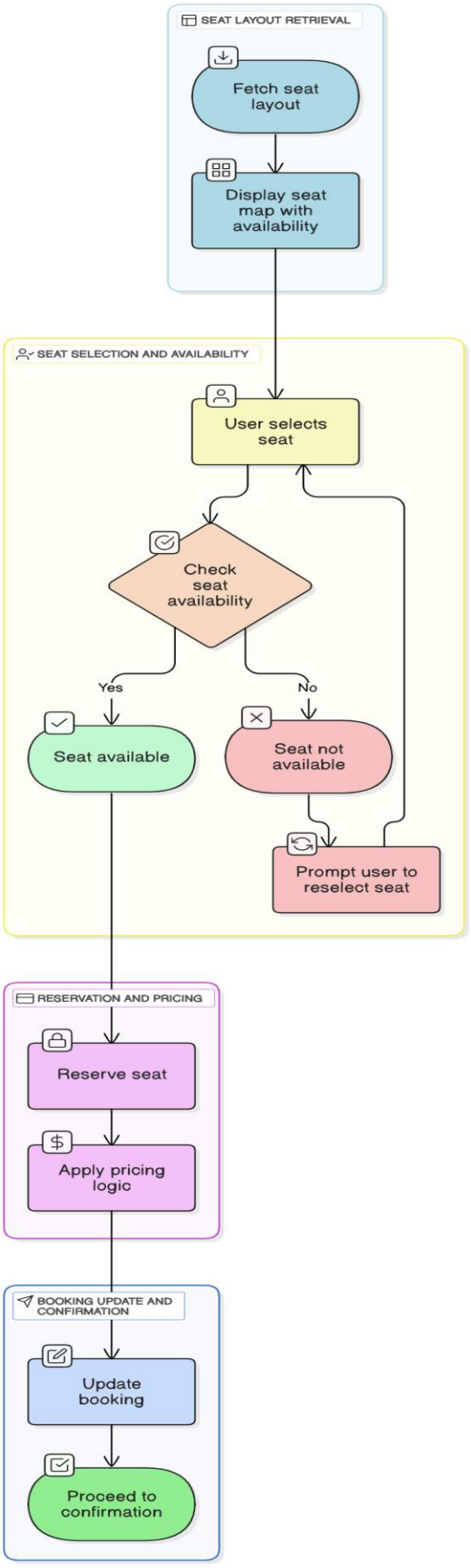
1. System fetches current seat map from database.
2. User selects a seat.
3. Backend checks availability.
    - If available, locks the seat temporarily.
    - If not, prompts re-selection.
4. Price is recalculated if dynamic pricing is enabled.
5. User proceeds to payment and seat status updates to "booked."

Future Enhancements:

- Seat filters (class, extra legroom)
- Real-time sync with multiple users
- Accessibility-aware seat suggestions
-

Diagram:

Airline Seat Allocation Flow



4. Payment & Confirmation Module

Overview:
Simulates the payment process and finalizes bookings. While it does not integrate with real gateways in the prototype, the logical flow ensures system integrity.

Why It Matters:
It ensures users receive a clear confirmation and that bookings are atomic—either fully successful or not recorded at all.

Key Features:

- Fare computation including base price, taxes, and modifiers
- Final confirmation before commit
- Transaction simulation
- Booking ID and confirmation screen

Technical Workflow:

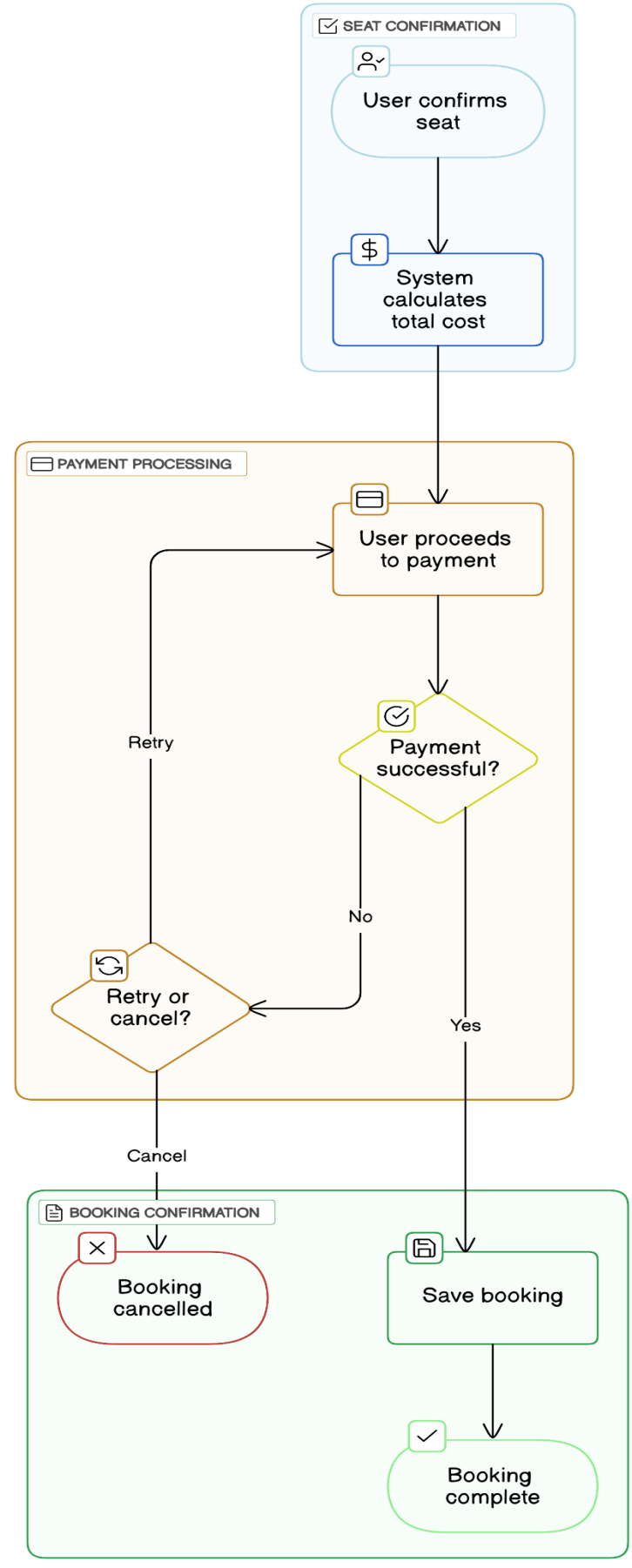
1. Total cost is computed with surcharges.
2. User reviews booking summary and confirms.
3. System creates a booking entry in the bookings table.
4. Receipt/confirmation is generated and shown.

Future Enhancements:

- Real payment gateway integration (e.g., Stripe, Razorpay)
- Email/SMS confirmation
- Refund and cancellation logic

Diagram:

Payment and Confirmation Module Flow



5. Admin Module

Overview:
Back-office system enabling administrative oversight and operations. This includes managing flights, monitoring occupancy, and modifying schedules or prices.

Why It Matters:
Without a dedicated admin backend, the system cannot be maintained or adjusted post-deployment.

Key Features:

- Secure admin login (separate from users)
- Add/modify/delete flights
- View live bookings by route or time
- Access analytics for decision-making

Technical Workflow:

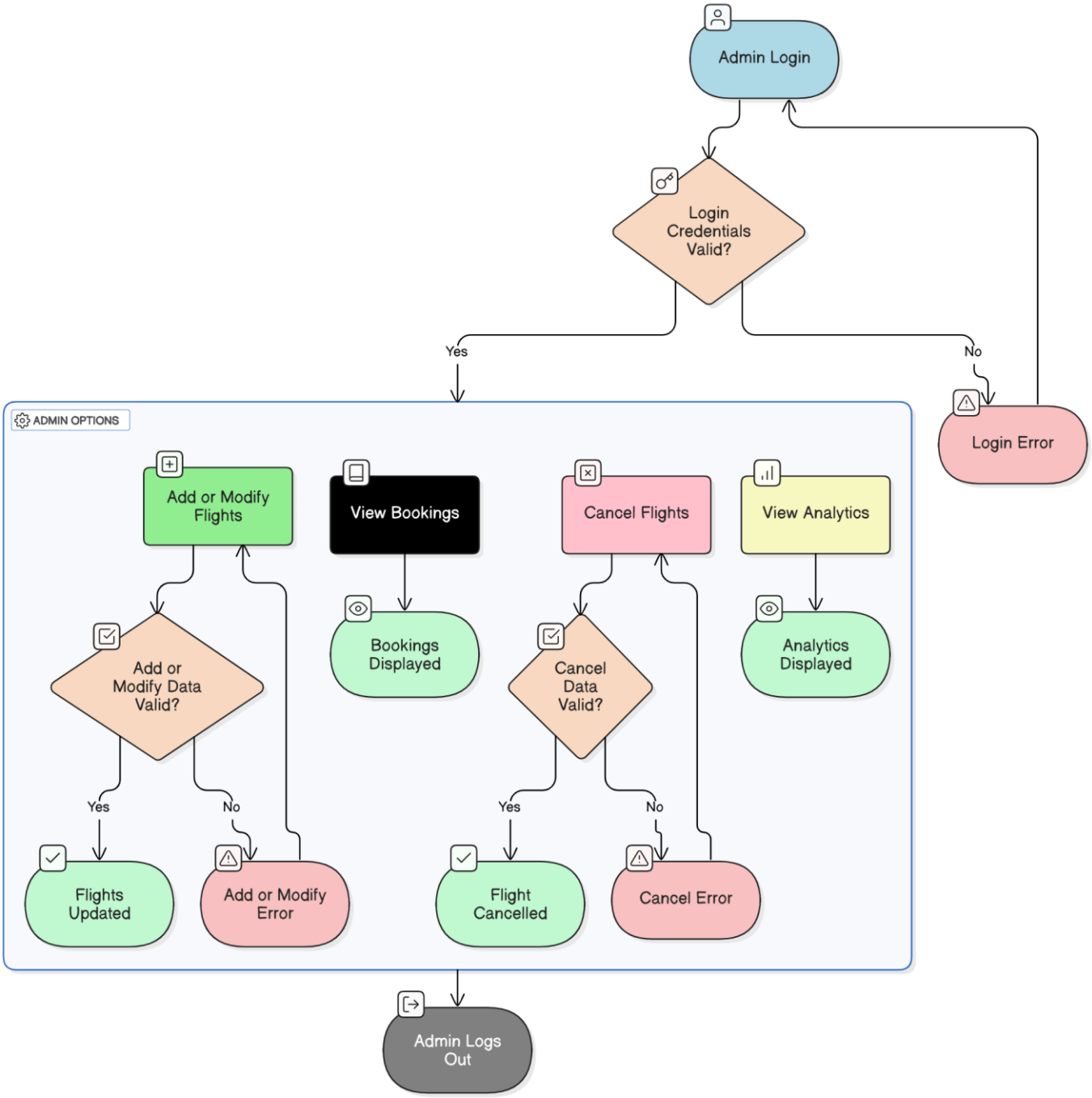
1. Admin logs in through a dedicated portal.
2. Main dashboard displays management options:
    - Add flights: Inputs flight ID, route, timings, seats, fare.
    - Update/delete existing schedules.
    - View all user bookings sorted by date or route.
3. Link to Analytics Module for deeper insights.

Future Enhancements:

- Role-based access (super-admin, manager)
- System logs and audit tracking
- Multi-admin collaboration

Diagram:

Airline System Admin Module Flowchart



6. AI Chatbot Module

Overview:
A simple AI chatbot integrated for user assistance. Handles basic queries and guides users through booking steps, cancellation, and FAQs.

Why It Matters:
Adds self-service support. Reduces dependency on human agents for common questions.

Key Features:

- Pattern-based response matching
- Fallback to “rephrase” if input is unclear
- Intent recognition (greeting, help, refund, booking)

Technical Workflow:

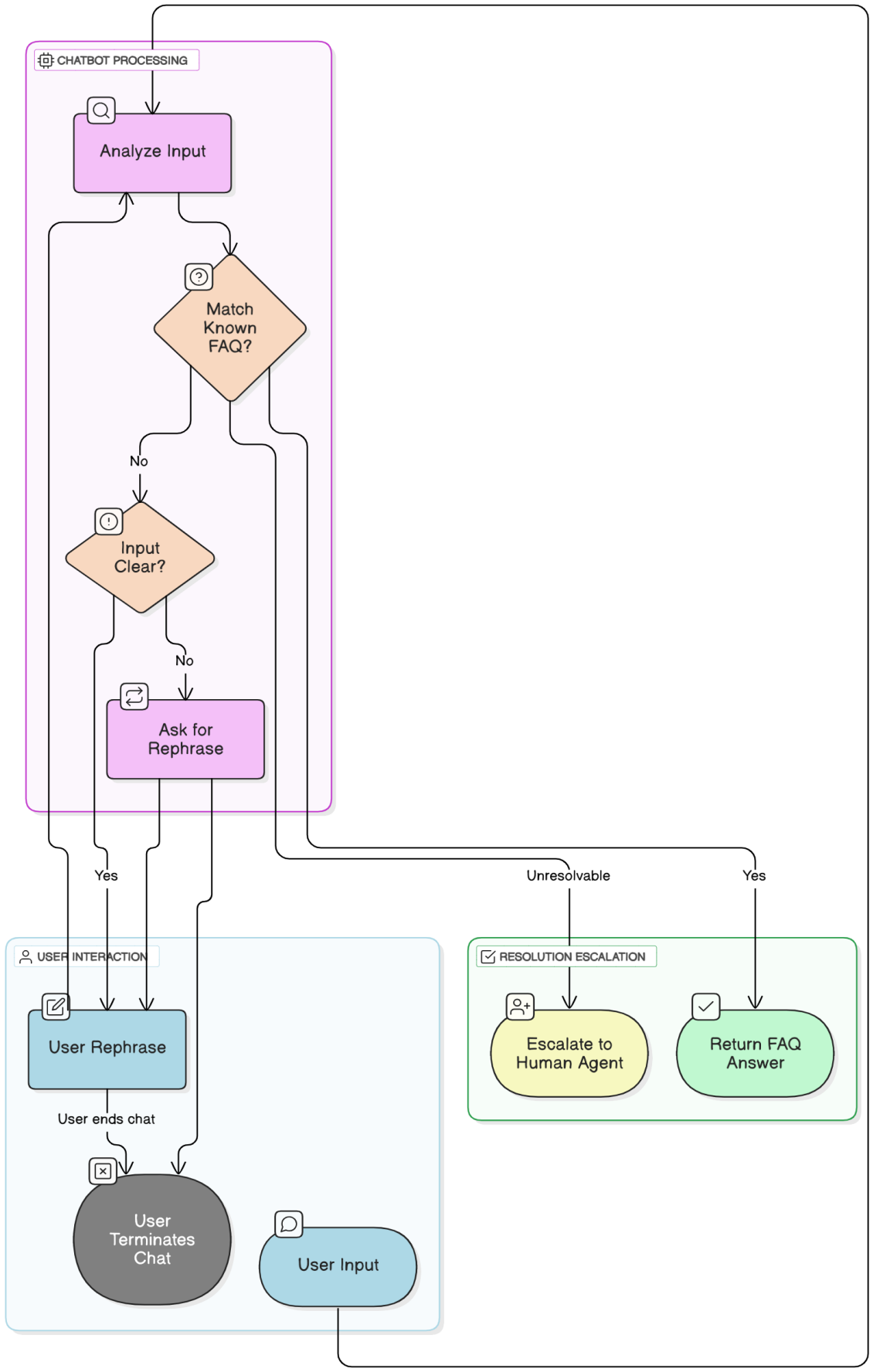
1. User inputs text into chatbot box.
2. Backend scans for keyword matches (e.g., “how to book”, “refund policy”).
3. A relevant predefined response is sent back.
4. If input is unmatched, chatbot prompts for clearer input.

Future Enhancements:

- NLP upgrades using spaCy or NLTK
- Multi-turn conversation memory
- Live agent escalation handoff

Diagram:

AI Chatbot Module Flow for Flight Booking Website



7. Analytics Module

Overview:
Provides visual insight into operational performance. Helps admins understand booking behavior, revenue flow, and demand hotspots.

Why It Matters:
Data visualization is critical for strategic planning. It also validates user growth and system performance over time.

Key Features:

- Booking trends over time
- Top 5 routes by popularity
- Occupancy heatmaps
- Revenue breakdown by date and route

Technical Workflow:

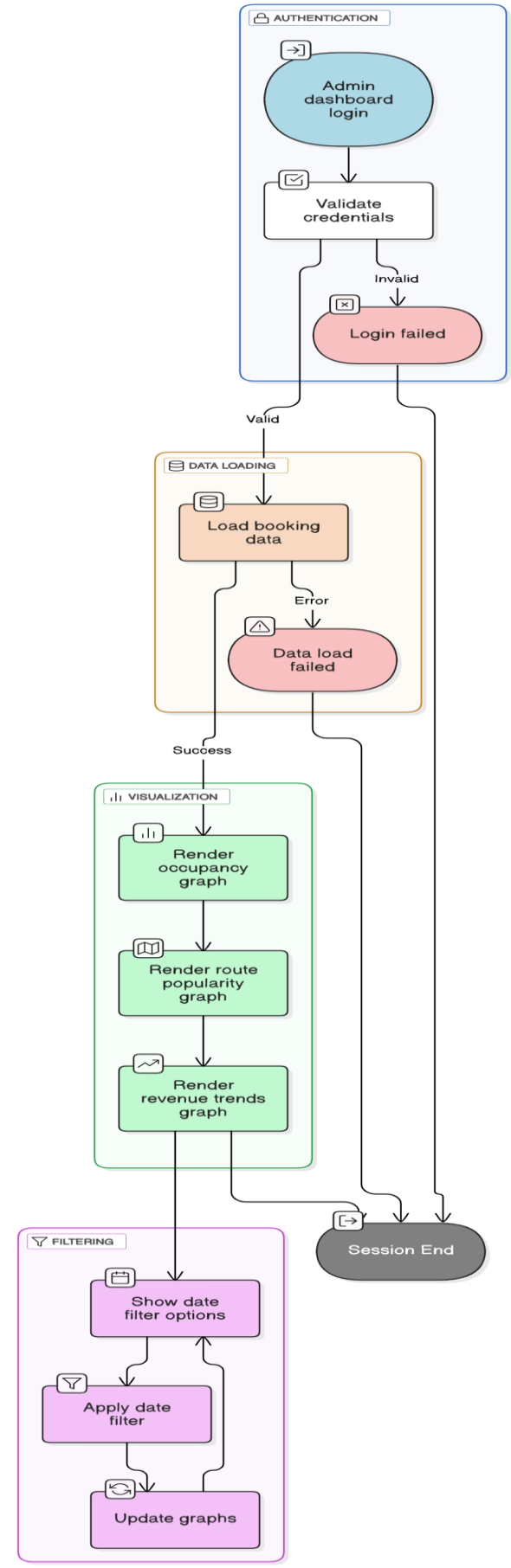
1. System extracts data from `bookings`, `flights`, and `users`.
2. Uses JavaScript charting (e.g., Chart.js) to display insights.
3. Admin can filter charts by date range, route, or booking type.
4. Visual outputs: bar charts, pie charts, line graphs.

Future Enhancements:

- Export to Excel/CSV
- Predictive analytics (e.g., demand forecasting)
- Real-time KPIs

Diagram:

Analytics Module Workflow for Airline Reservation System



8. Weather Module

Overview:

Integrates real-time weather data to enhance flight management and user experience. This module fetches current weather conditions for departure and arrival airports, influencing flight status and providing timely alerts to users.

Why It Matters:

Weather conditions are critical in aviation. By proactively monitoring and responding to weather changes, the system ensures safety, optimizes scheduling, and keeps users informed.

Key Features:

- Real-Time Weather Data: Fetches up-to-date weather information for relevant airports.
- Flight Status Updates: Automatically adjusts flight statuses (e.g., Delayed, Canceled) based on adverse weather conditions.
- User Notifications: Sends alerts to users regarding weather-induced changes to their flights.
- Admin Dashboard Integration: Displays weather conditions and related flight impacts for administrative oversight.

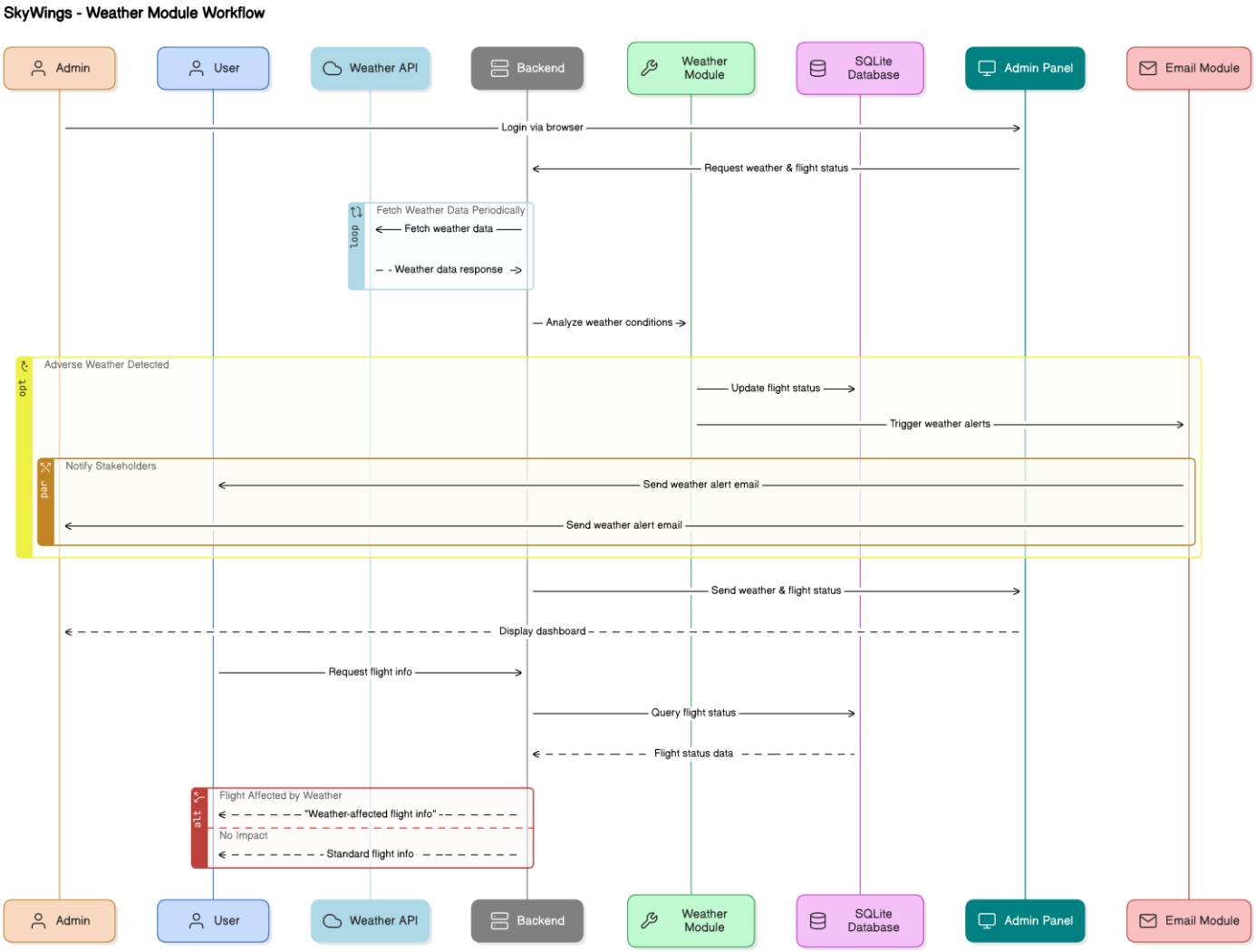
Technical Workflow:

1. Data Retrieval: Periodically fetch weather data from a reliable API (e.g., OpenWeatherMap) for all airports involved in scheduled flights.
2. Condition Analysis: Evaluate weather parameters (e.g., visibility, wind speed) against predefined thresholds.
3. Status Adjustment: If adverse conditions are detected, update the flight's status in the database accordingly.
4. Notification Trigger: Initiate alerts to affected users via the Email Module.
5. Dashboard Update: Reflect changes in the admin interface for monitoring and further action.

Future Enhancements:

- Predictive Analytics: Incorporate machine learning to forecast weather impacts on flight schedules.
- User Preferences: Allow users to set preferences for receiving weather-related notifications.
- Expanded Data Sources: Integrate multiple weather data providers for redundancy and accuracy.

Diagram:



9. Email Module

Overview:

Manages all automated email communications within the SkyWings system, including booking confirmations, flight updates, and promotional messages. Utilizes daemon processes to handle email dispatch asynchronously, ensuring non-blocking operations.

Why It Matters:

Timely and reliable communication enhances user trust and satisfaction. Asynchronous email handling ensures that the main application remains responsive, especially during high-load scenarios.

Key Features:

- Automated Notifications: Sends emails for booking confirmations, cancellations, flight status changes, and more.
- Asynchronous Processing: Employs daemon threads or background tasks to send emails without delaying user interactions.

- Template Management: Utilizes standardized templates for consistent and professional communication.
- Error Handling: Implements retry mechanisms and logging for failed email attempts.

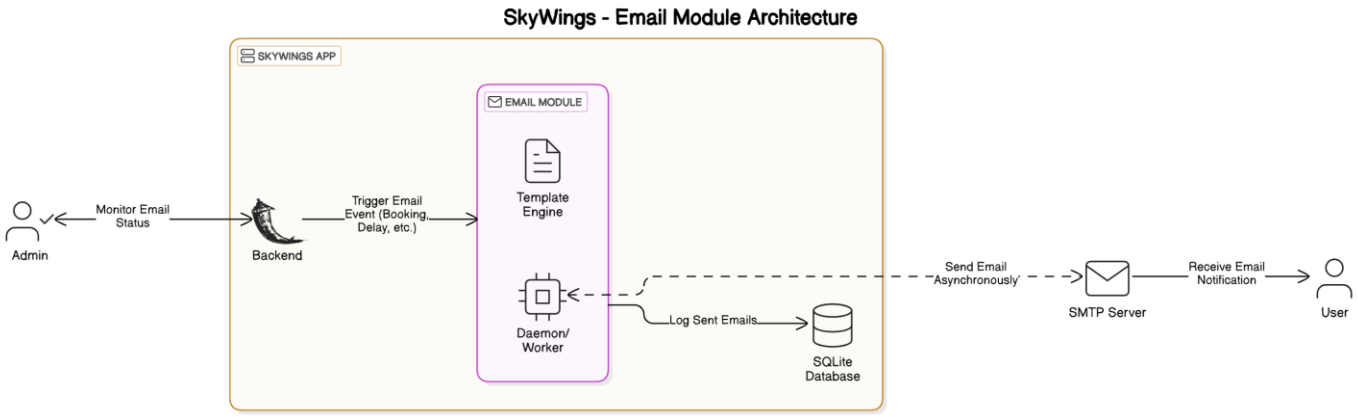
Technical Workflow:

1. Event Detection: Monitor for events that require email notifications (e.g., new booking, flight delay).
2. Queueing: Add email tasks to a queue managed by a background worker or daemon thread.
3. Email Composition: Generate email content using predefined templates and event-specific data.
4. Dispatch: Send emails via an SMTP server using libraries like Flask-Mail.
5. Logging: Record the status of each email sent, including successes and failures, for auditing and troubleshooting.

Future Enhancements:

- Advanced Queue Management: Integrate with task queues like Celery or RQ for more robust background processing.
- User Preferences: Allow users to customize their email notification settings.
- Analytics Integration: Track email open rates and engagement for marketing insights

Diagram:



System Design

System Architecture

The SkyWings Airline Reservation System follows a modular, scalable architecture that separates concerns across multiple layers—frontend, backend, data persistence, and an AI-driven service layer. This design promotes maintainability, performance, and clarity in operation.

Frontend Layer

- Technologies Used: HTML, CSS, JavaScript, and Bootstrap.
- Responsibilities: Provides a responsive and user-friendly interface for passengers, admins, and agents to interact with the system. Includes forms for registration, search, booking, and dashboards for both users and administrators.

Backend Layer

- Framework: Python with the Flask micro-framework.
- Responsibilities: Acts as the core logic handler, processing all API requests, user authentication, booking transactions, seat selection, and profile management. It serves dynamic content to the frontend and interacts with the database and chatbot module.

Database Layer

- Database Engine: SQLite.
- Design Considerations**:**
    - Relational schema using foreign keys to maintain data consistency.
    - Normalized structure for efficient storage and query performance.
    - Supports user records, flight schedules, bookings, payment history, seat maps, and admin configurations.

API Layer

- Architecture**:** RESTful APIs.
- Endpoints**:** Includes routes for user registration/login, flight search, seat booking, payment processing, admin controls, and chatbot services.
- Usage: Enables asynchronous operations between client and server, improving responsiveness and modular integration.

AI Integration

- Chatbot Module**:**
    - Implemented as a rule-based NLP engine in Python.
    - Provides real-time responses to frequently asked user queries about flights, bookings, and general system use.
    - Integrated directly into the Flask app and accessible through a dedicated chat interface.

Database Design

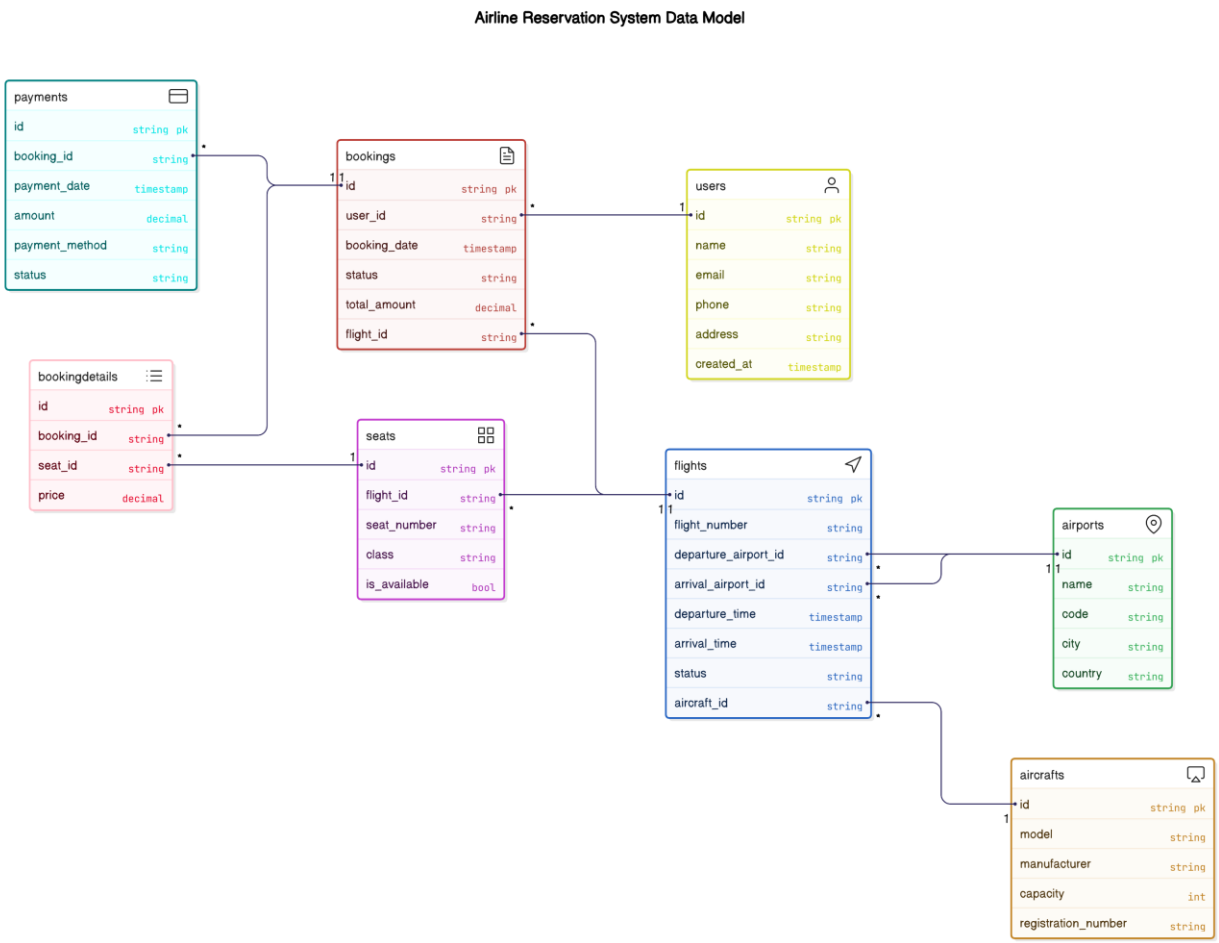
The SkyWings system employs a relational database architecture using SQLite to ensure portability and simplicity while supporting complex queries and constraints. The schema has been normalized to minimize redundancy and ensure data integrity through the use of foreign keys.

Key Entities and Relationships

- Users: Stores user credentials, profile details, and preferences. Includes admin distinction and loyalty tracking.
- Flights: Contains scheduling data, route information, pricing by class, and flight status updates.
- Aircraft: Details aircraft models, registration, and seating capacity across travel classes.
- Airports: Maintains geographical and identity data of departure and arrival points.
- Seats: Dynamic allocation per flight; maps seat class, availability, and booking status.
- Bookings: Core table linking users to their reservations. Tracks status, payment, and metadata.
- Booking Details: Stores passenger-specific data per booking including seat and personal info.
- Payments: Manages payment details, status, and method for each booking.

Design Features

- Relational Integrity: Enforced via foreign key constraints (e.g., bookings → users, flights).
- Scalability: Designed to accommodate additional entities like loyalty programs, baggage management, and reviews in future versions.
- Extensibility: Schema allows flexible integration with analytics, email notifications, and third-party APIs.



Technology Stack

The project is built with a full-stack approach using widely adopted and easily deployable technologies. The stack was chosen to maximize development speed, cross-platform compatibility, and maintainability.

Frontend

- HTML, CSS, JavaScript: Core technologies for layout, styling, and interactivity.
- Bootstrap: Used for responsive design, prebuilt components, and grid system.

Backend

- Python (Flask Framework)**:**
  - Lightweight web framework for routing, session management, and API development.
  - Handles business logic, authentication, seat allocation, and integrates with the chatbot module.

Database

- SQLite**:**
  - Chosen for its simplicity, zero-configuration nature, and ease of embedding within Python apps.
  - Suitable for prototyping and educational environments with low to moderate concurrency.

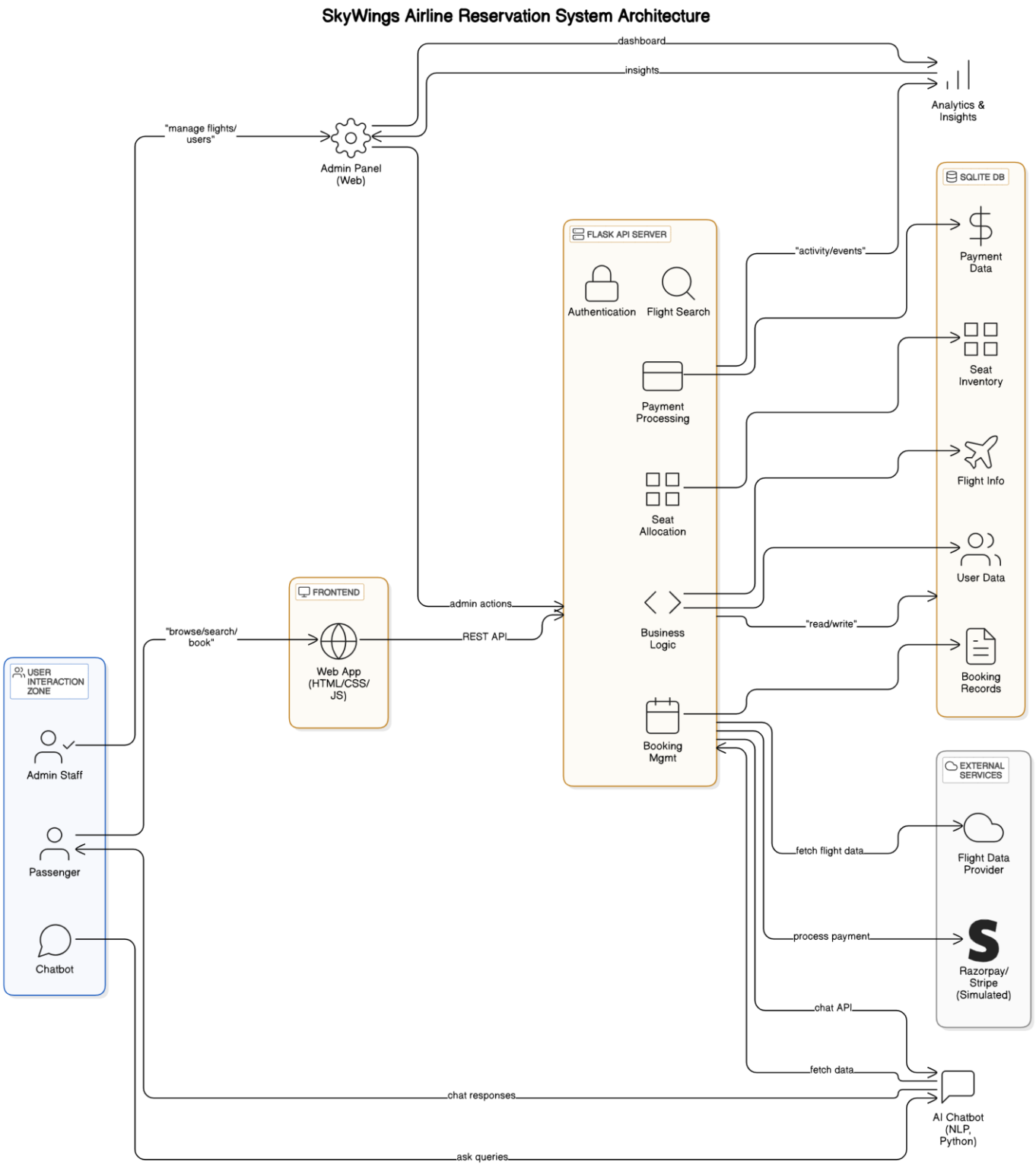
AI Component

- Custom Chatbot**:**
  - Python-based rule-driven chatbot using basic NLP techniques.
  - Integrated via a RESTful endpoint and rendered on the frontend chat UI.
  - Helps users with common queries related to bookings, schedules, and policies.

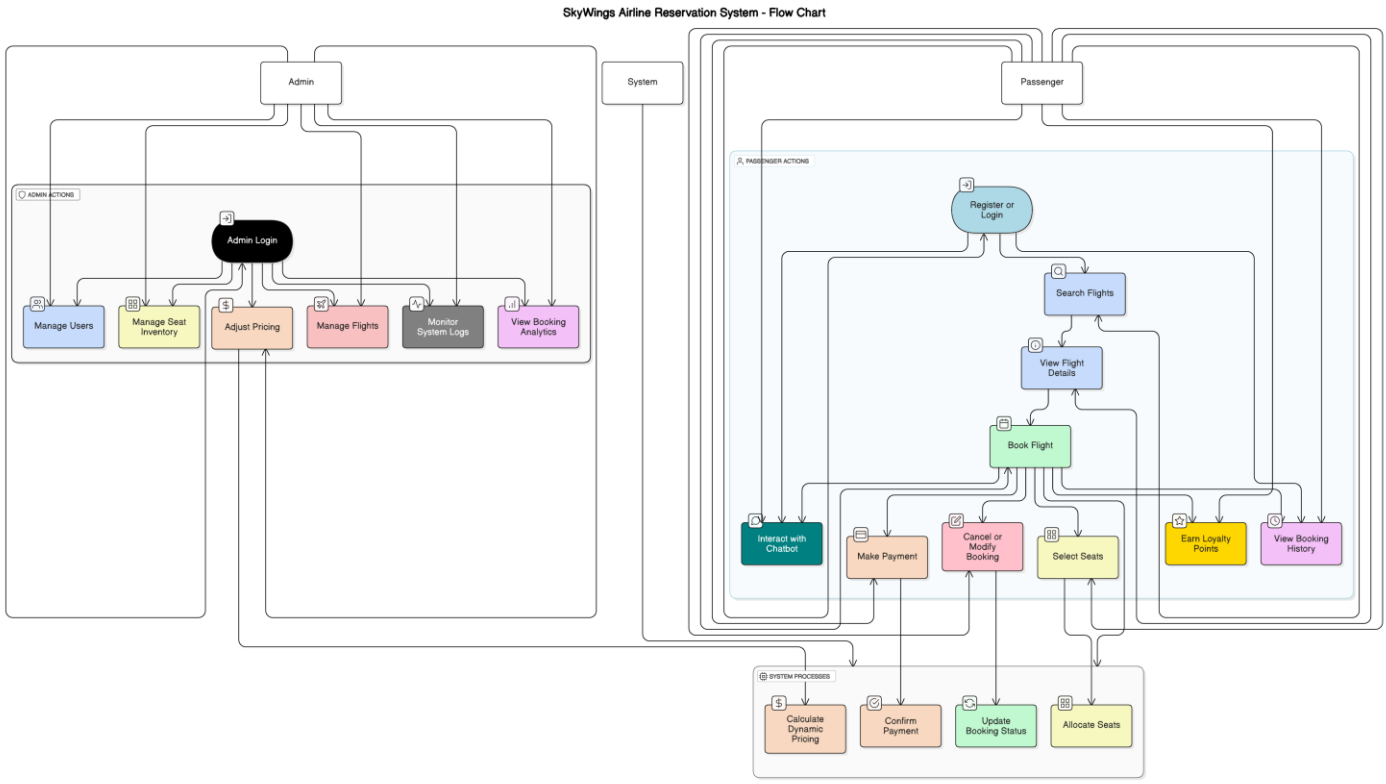
Other Tools

- Version Control**:** Git and GitHub for source management, collaboration, and deployment tracking.
- Stripe (or similar)**:** Integrated for payment processing via secure APIs (if implemented in your project).
- Jinja2**:** Templating engine used in Flask to dynamically render pages with user-specific data.

Architecture Diagram:



User Case Diagram:



Implementation

Development Tools & Environment

- Visual Studio Code
- GitHub for version control
- Flask server on localhost
- SQLite for lightweight DB deployment

Frontend Implementation Details

- Responsive design using Bootstrap
- Dynamic content loading via JS
- User-friendly booking interface

Backend Implementation Details

- Python Flask routes handle booking, seat checks, payments
- Custom seat allocation and pricing logic

API and Database Integration

- REST APIs link frontend forms to backend DB operations
- Uses SQL queries to fetch/update data in real-time

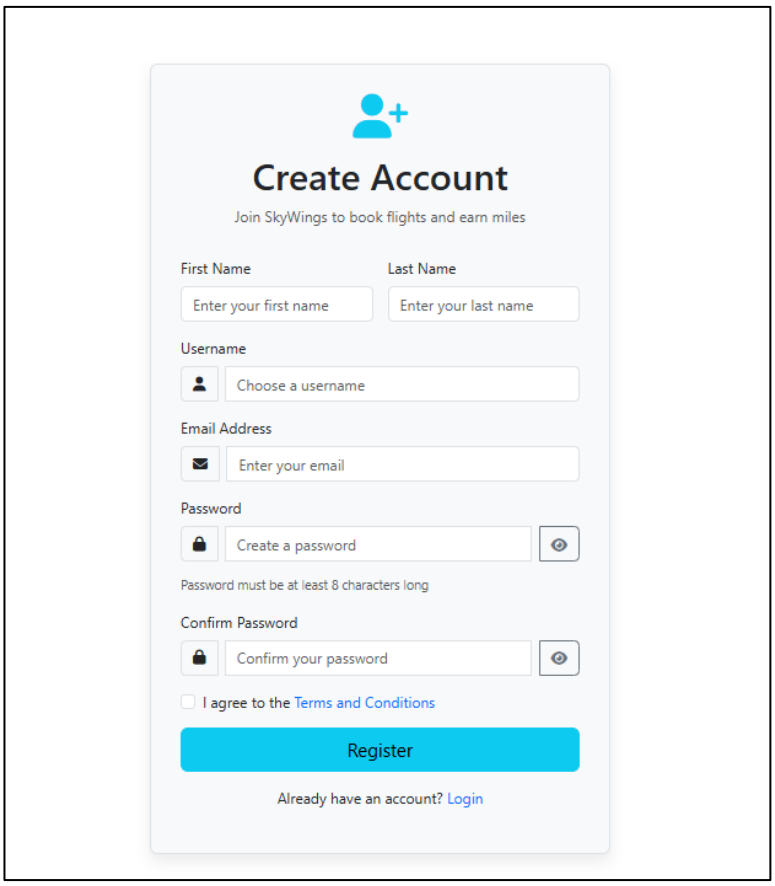
User Authentication and Sessions

- Session-based login management
- Secure user registration and access control

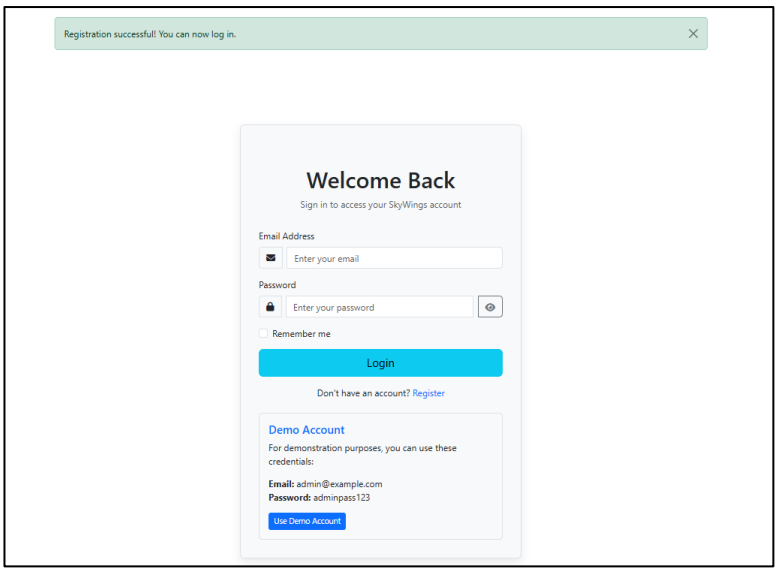
Screenshots of the outputs

User-Facing Interface:

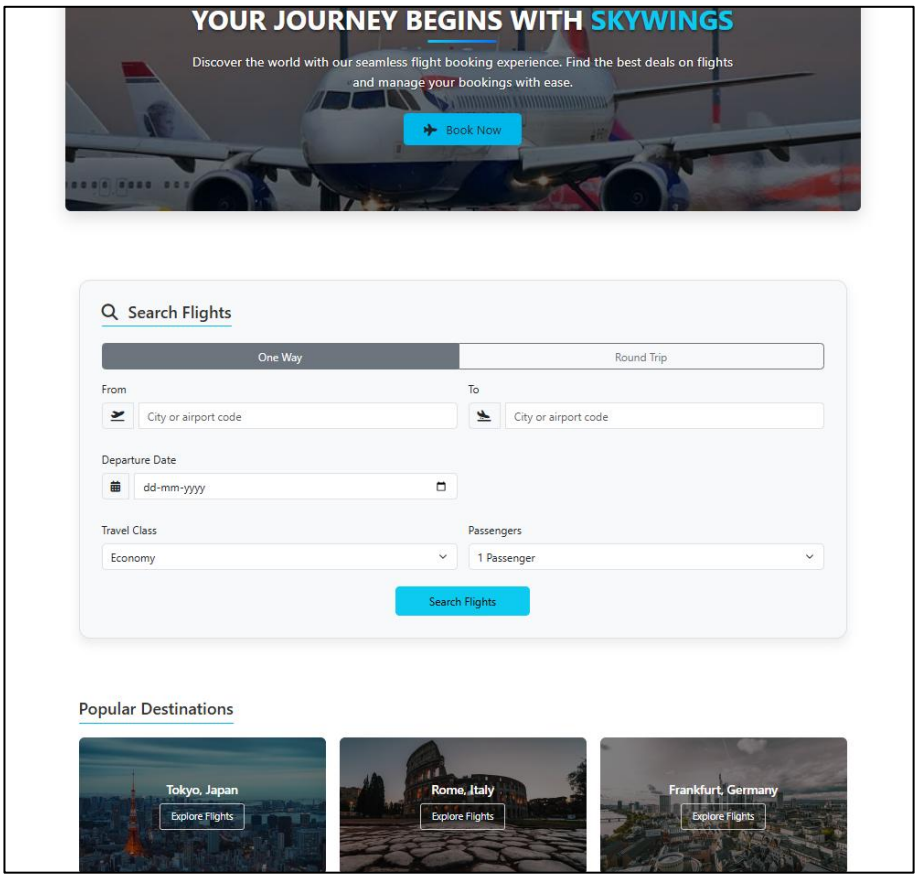
Registration page:



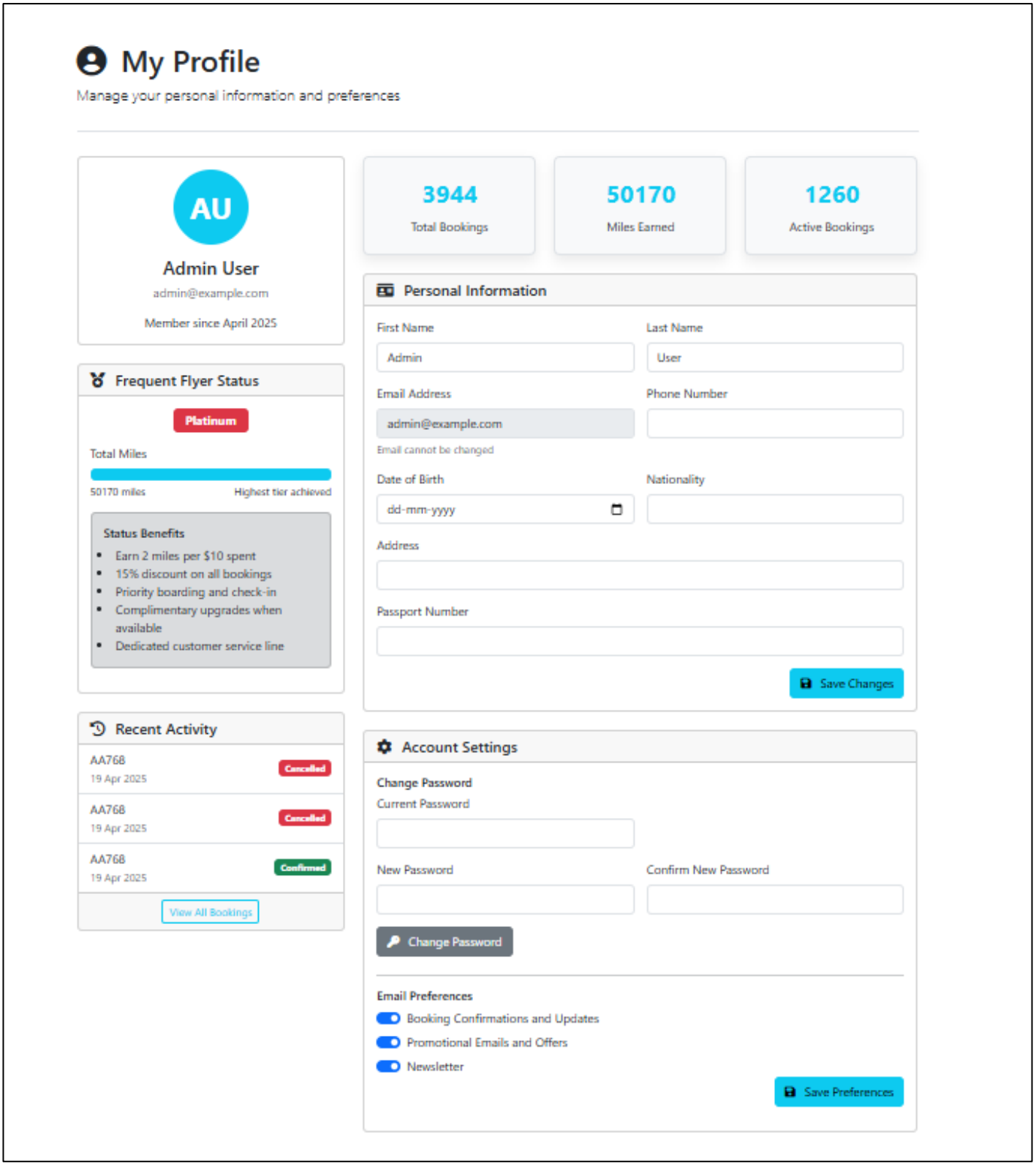
Login page:



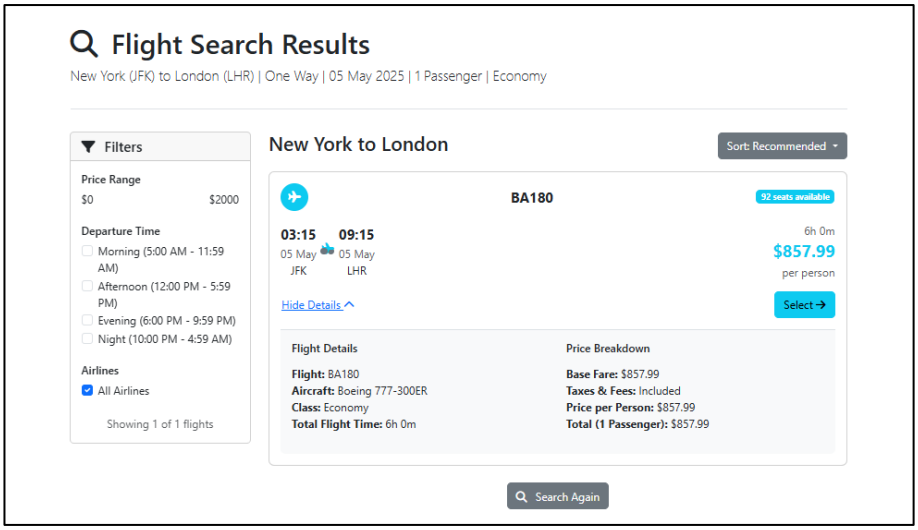
Search template:



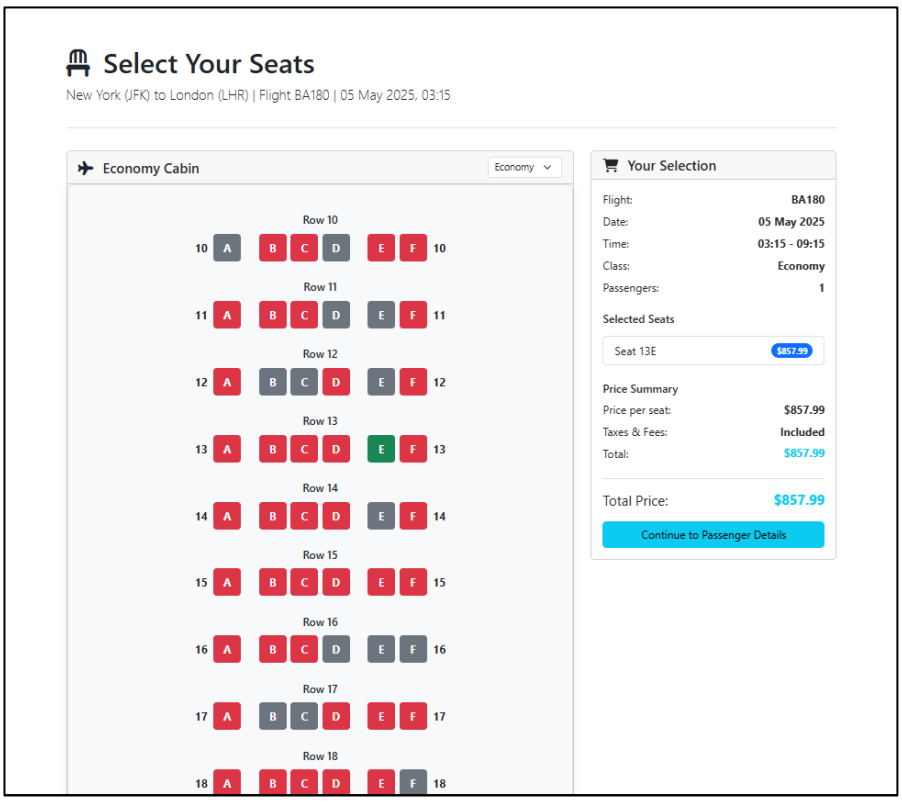
Profile Page:



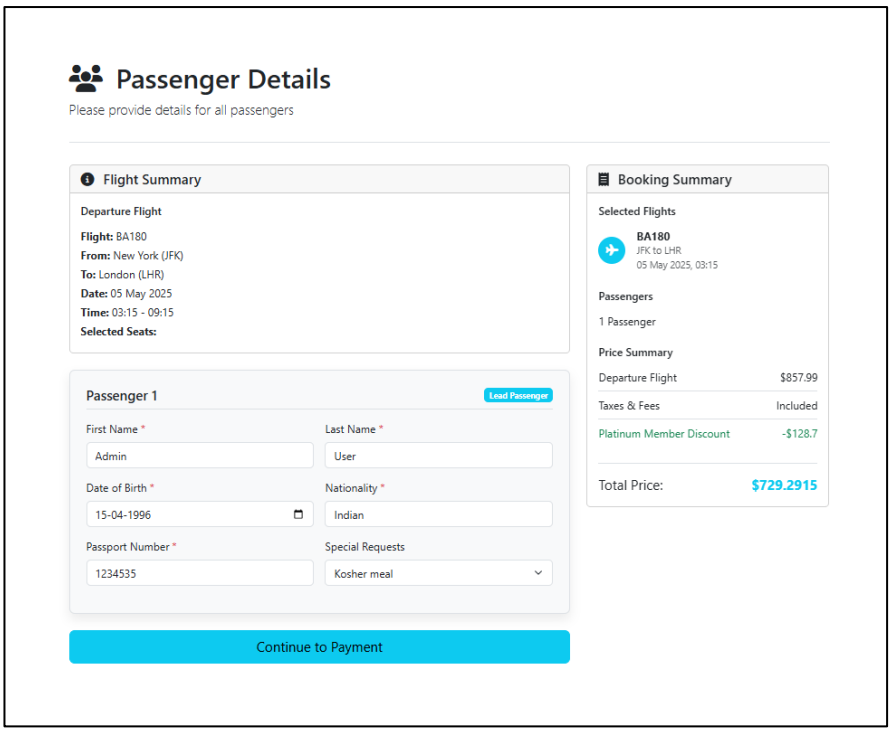
Search result:



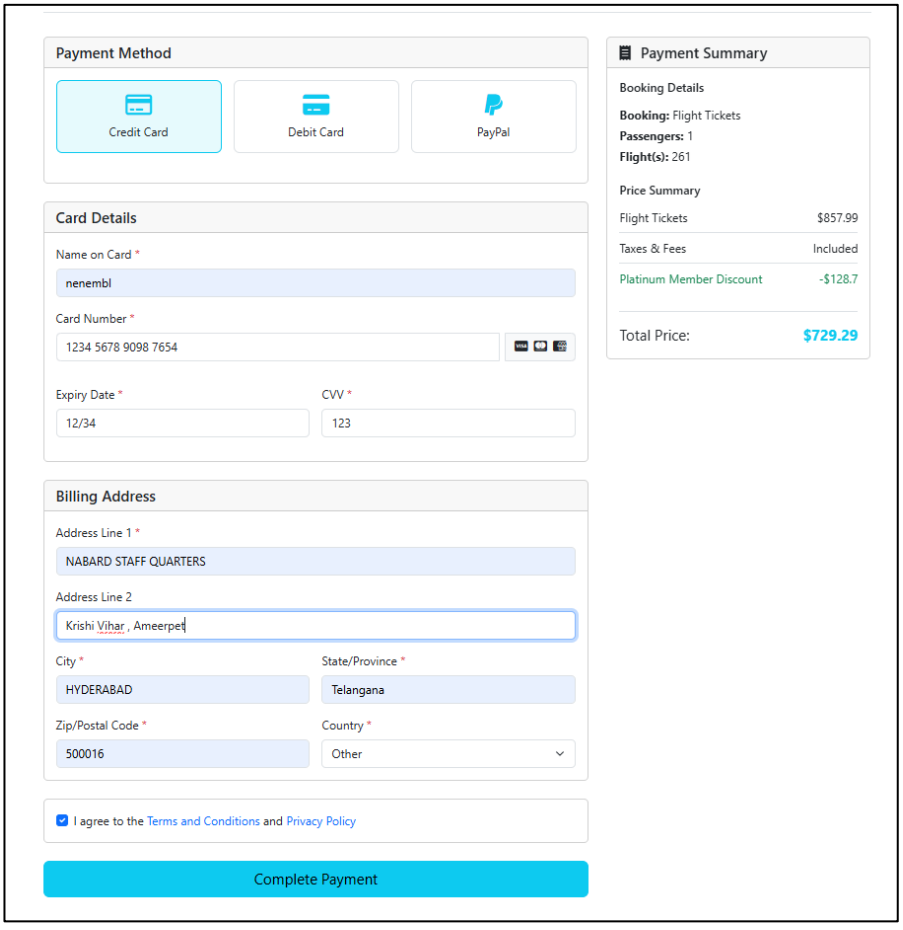
Seat selection:



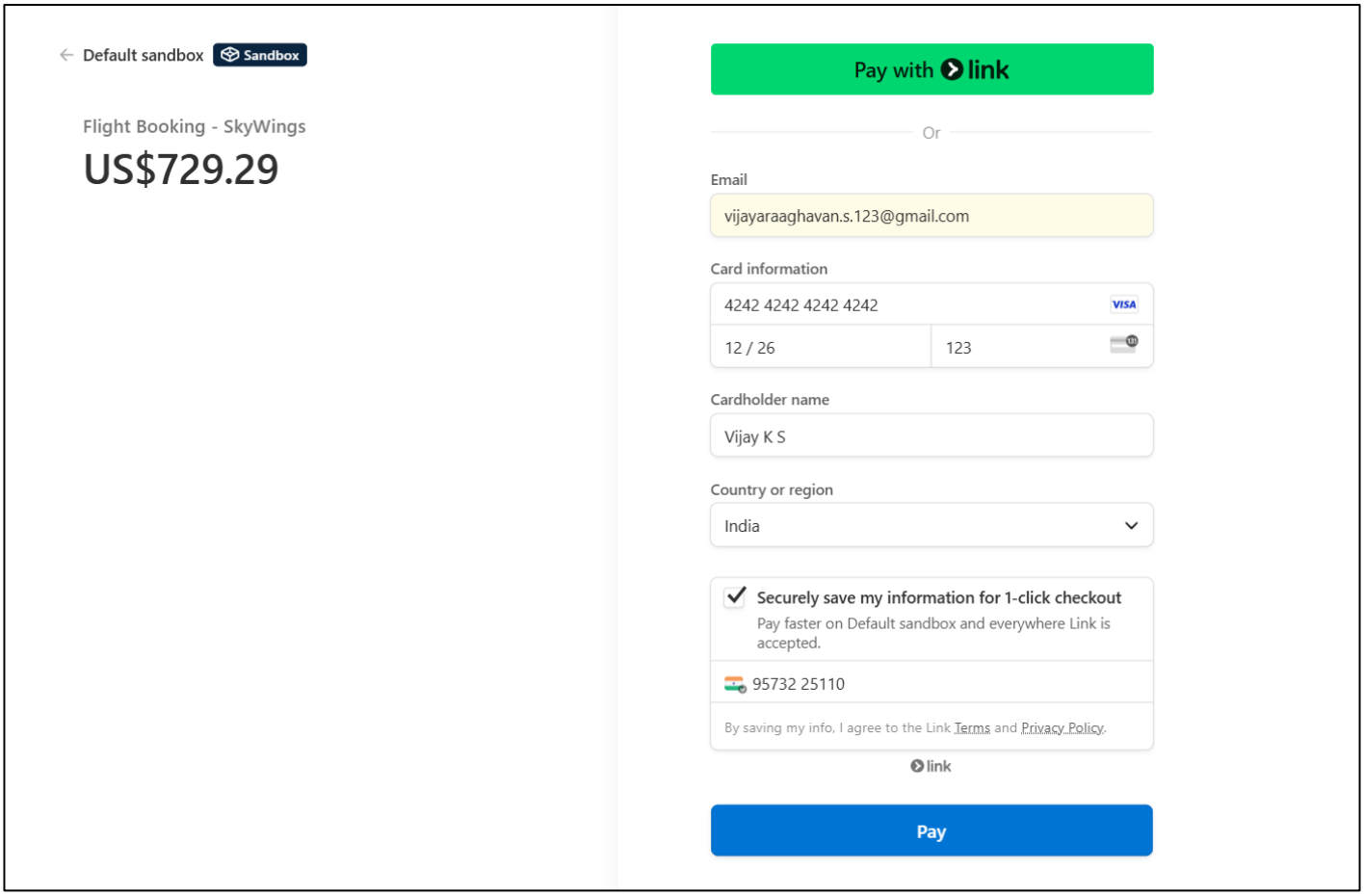
Passenger details entry:



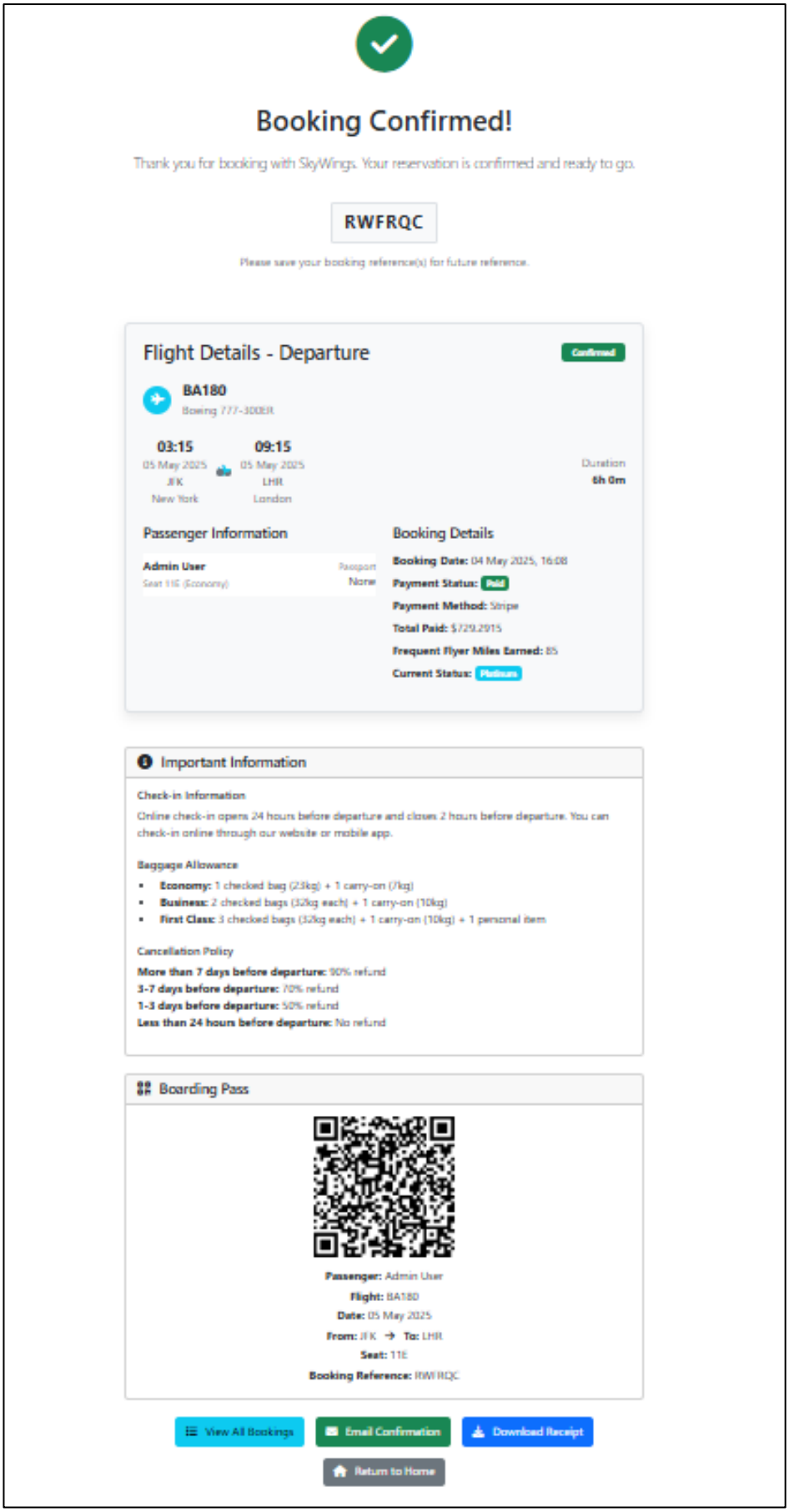
Payment information:



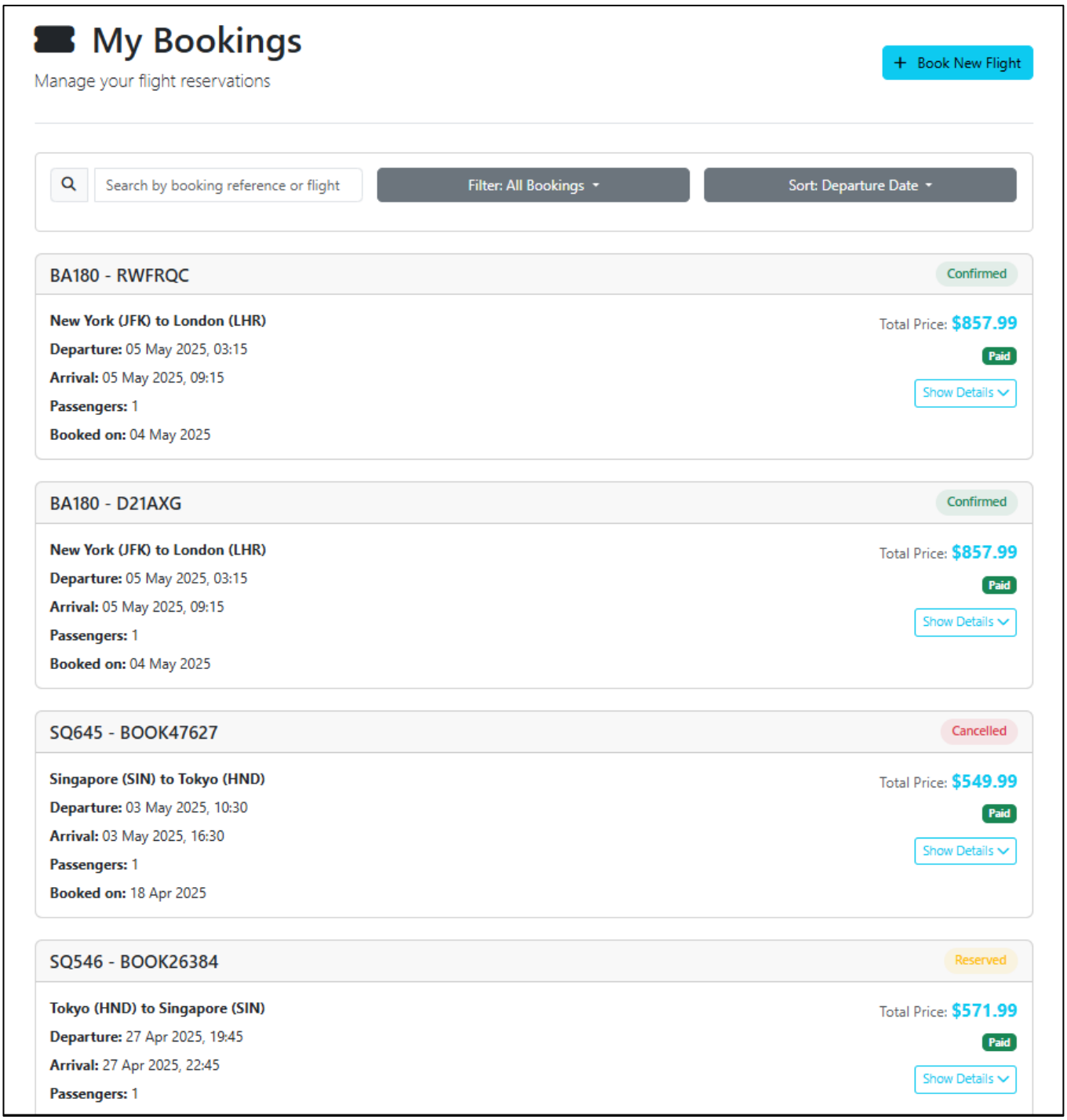
Payment processing:



Booking confirmation:

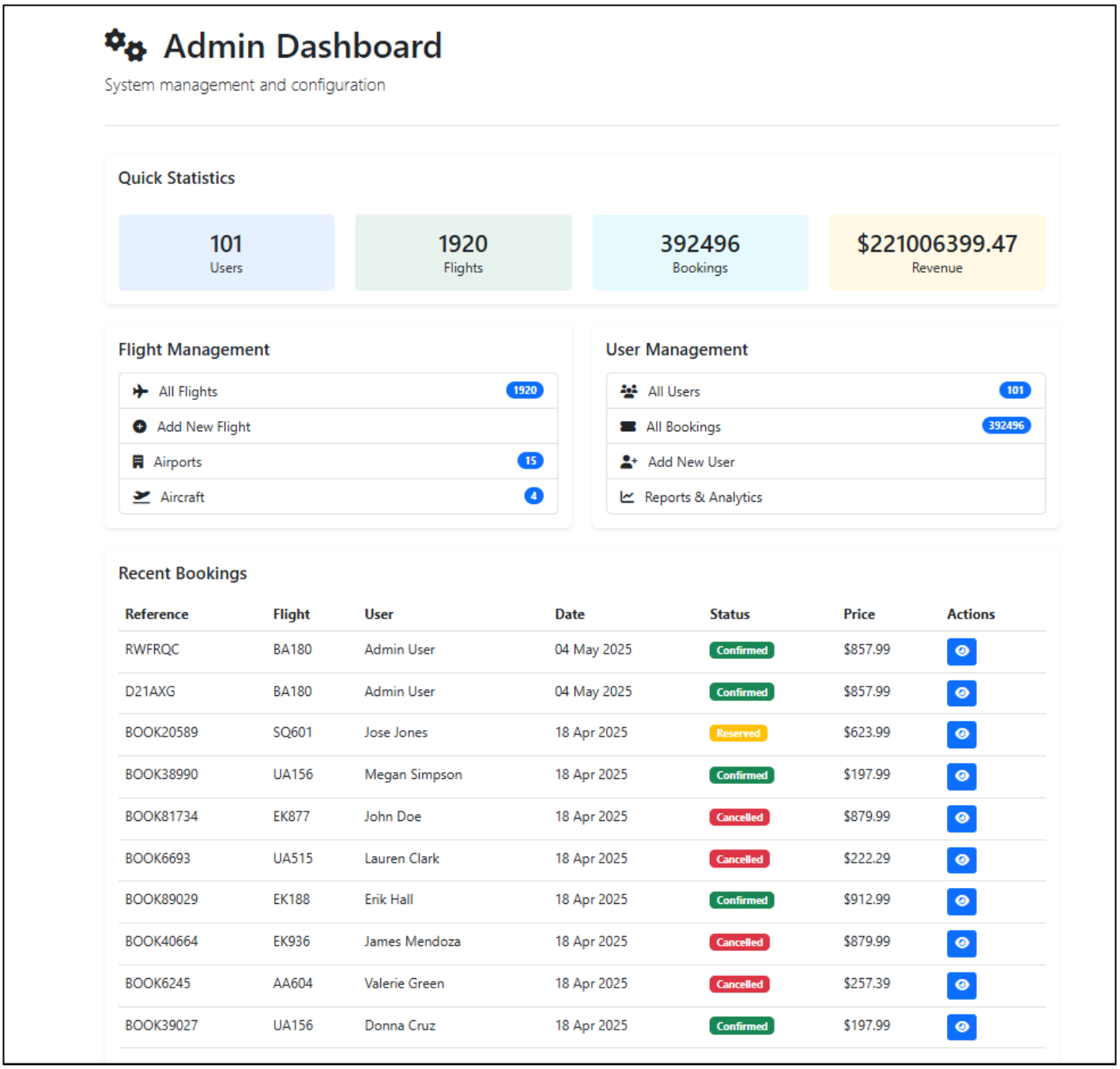


Booking history:



Admin Interface:

Admin dashboard:



Analytical Reports:

Reports & Analytics

Detailed insights into your flight booking system

Back to Dashboard

Last 30 Days

Total Users
101
Admins: 1 | Regular: 100

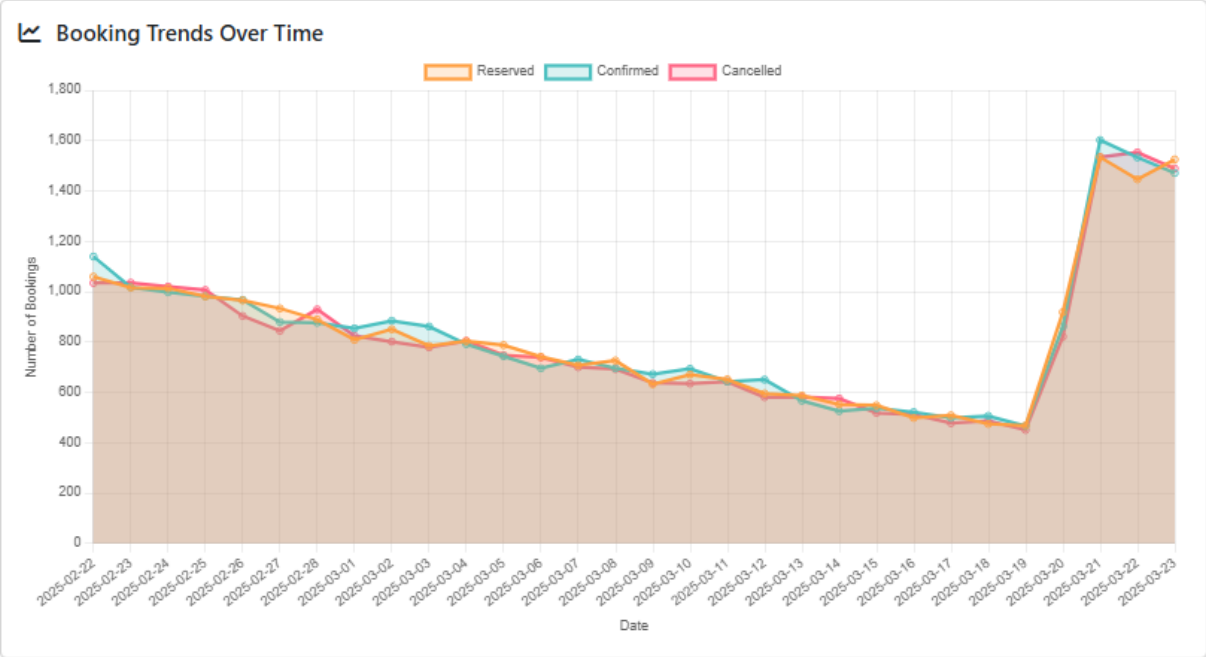
Total Bookings
392496
Reserved: 131055 | Confirmed: 130711 | Cancelled: 130730

Total Flights
1920
Scheduled: 225 | Departed: 1695

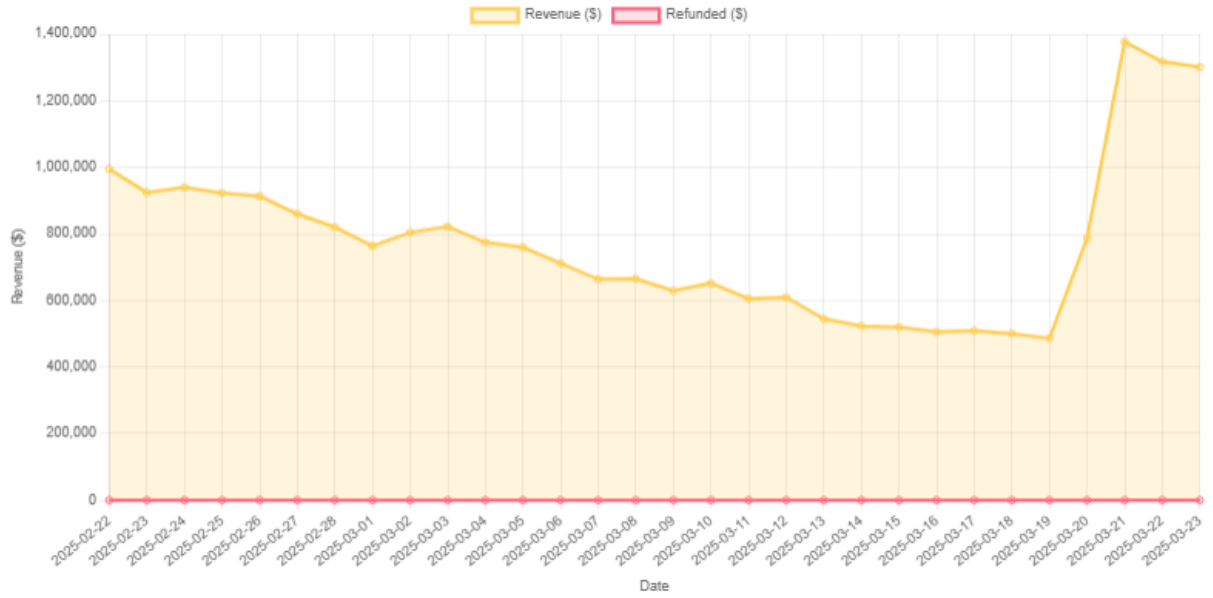
Total Revenue
$73364734.89
From confirmed bookings

Frequent Flyer Distribution

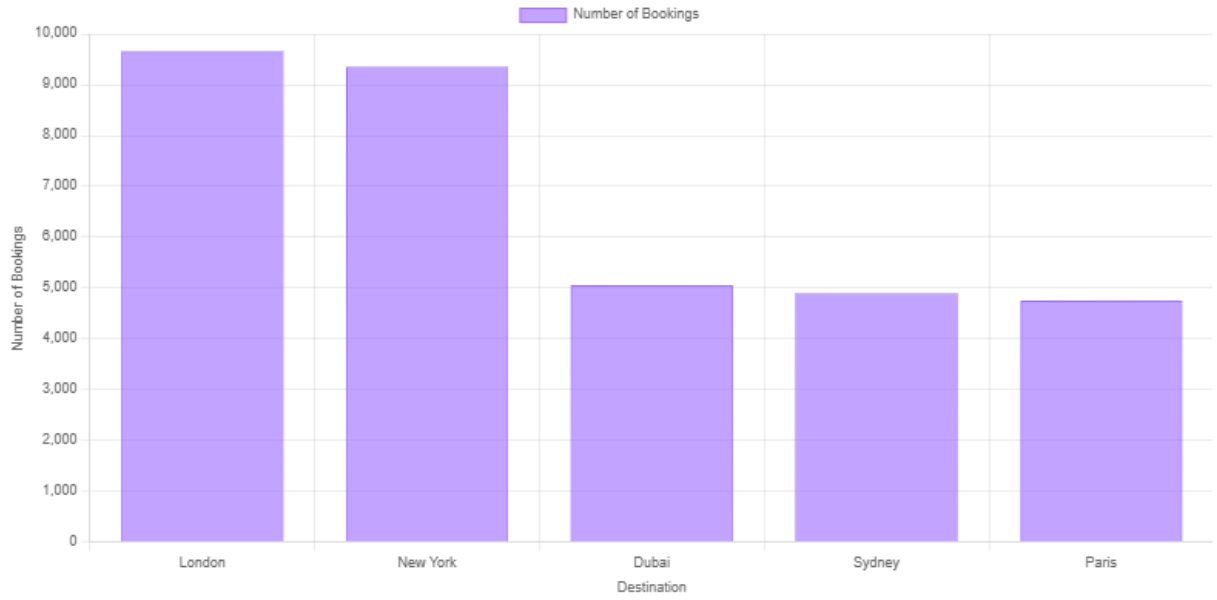
Gold: 32 users **Platinum**: 28 users **Silver**: 21 users **Standard**: 20 users



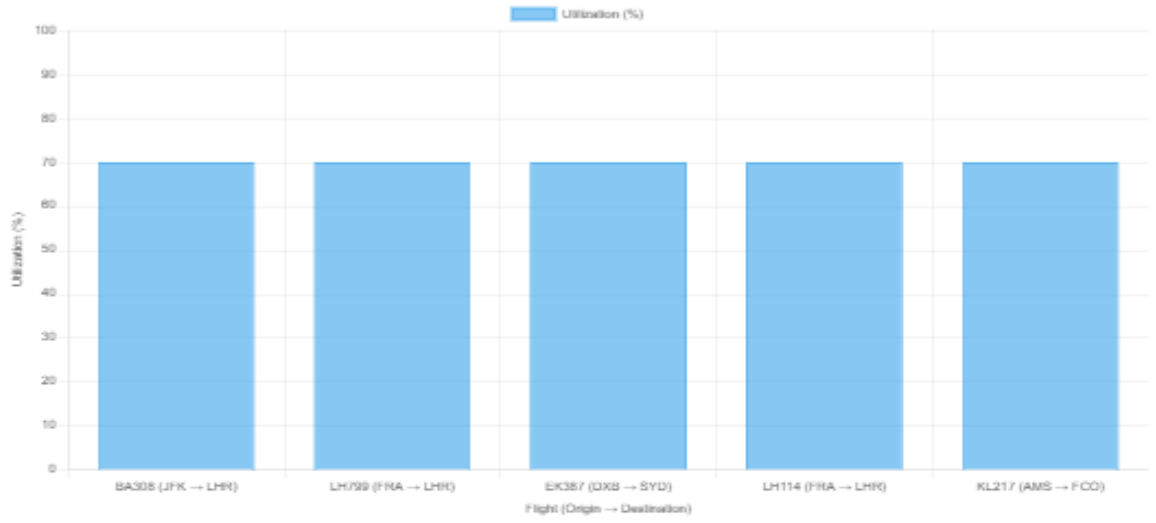
$ Revenue Trends Over Time



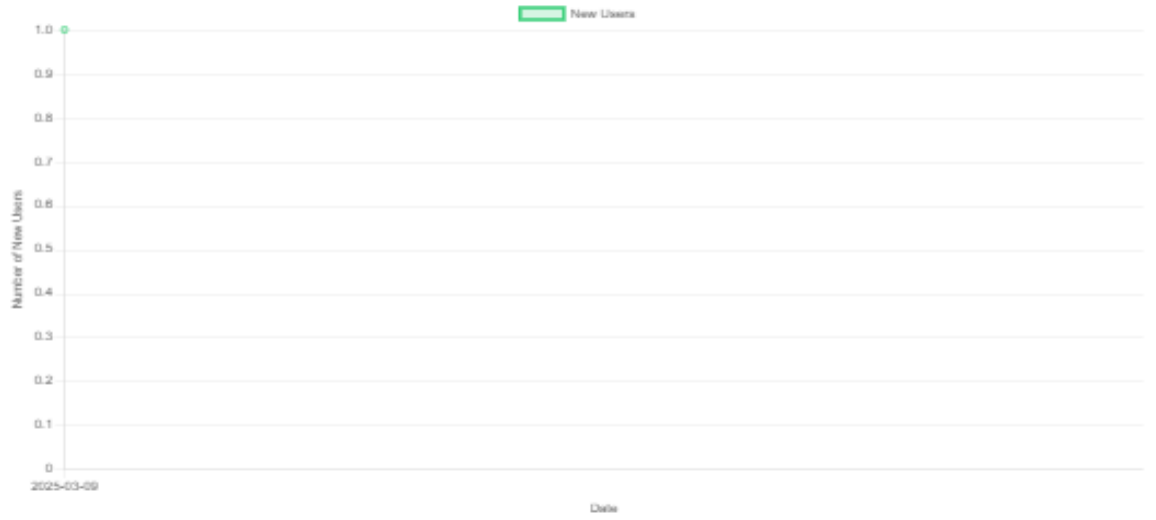
Top Destinations by Bookings



Flight Utilization (Top 5 Flights)



Increase in Users Over Time

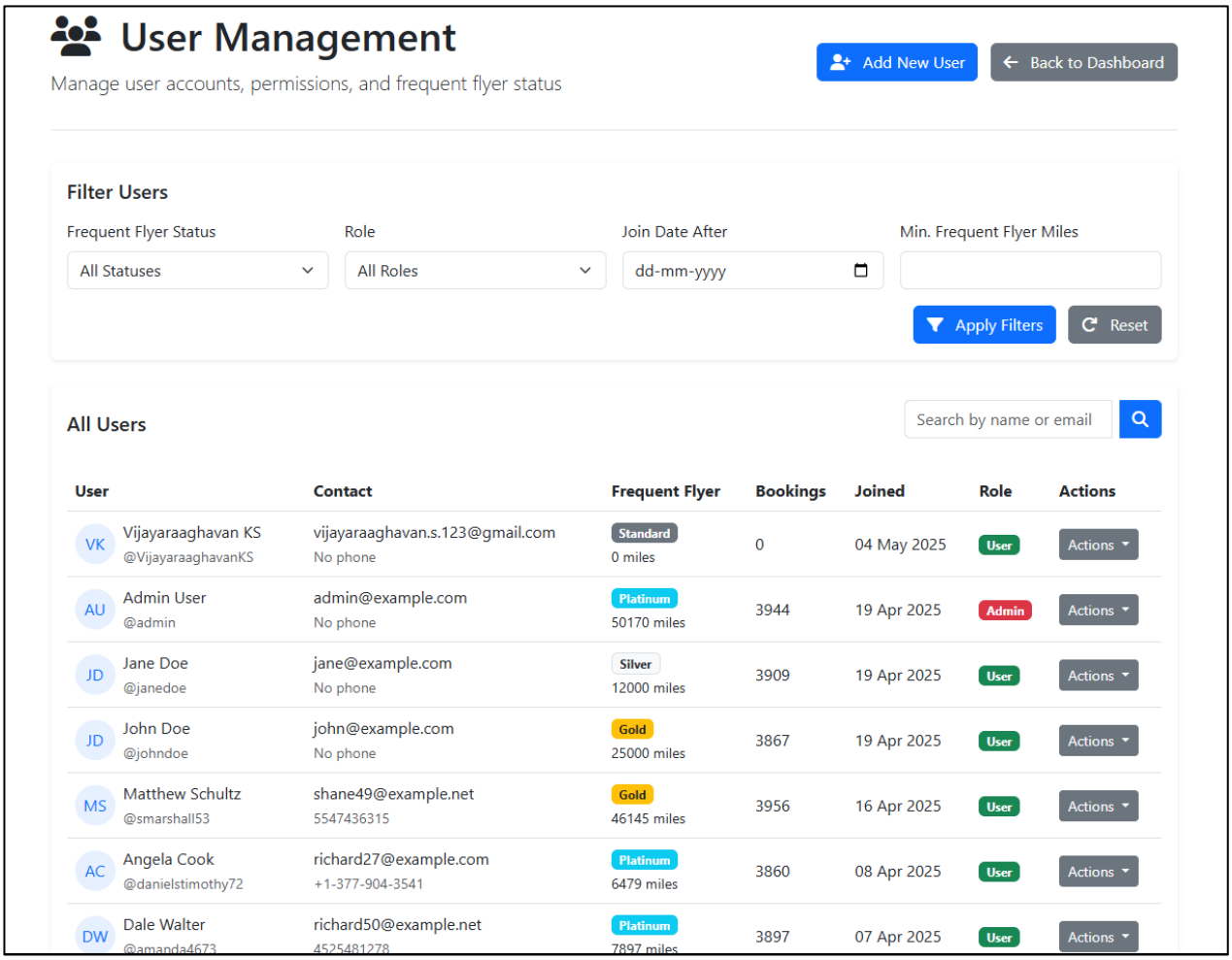


$ Revenue Breakdown by Seat Class

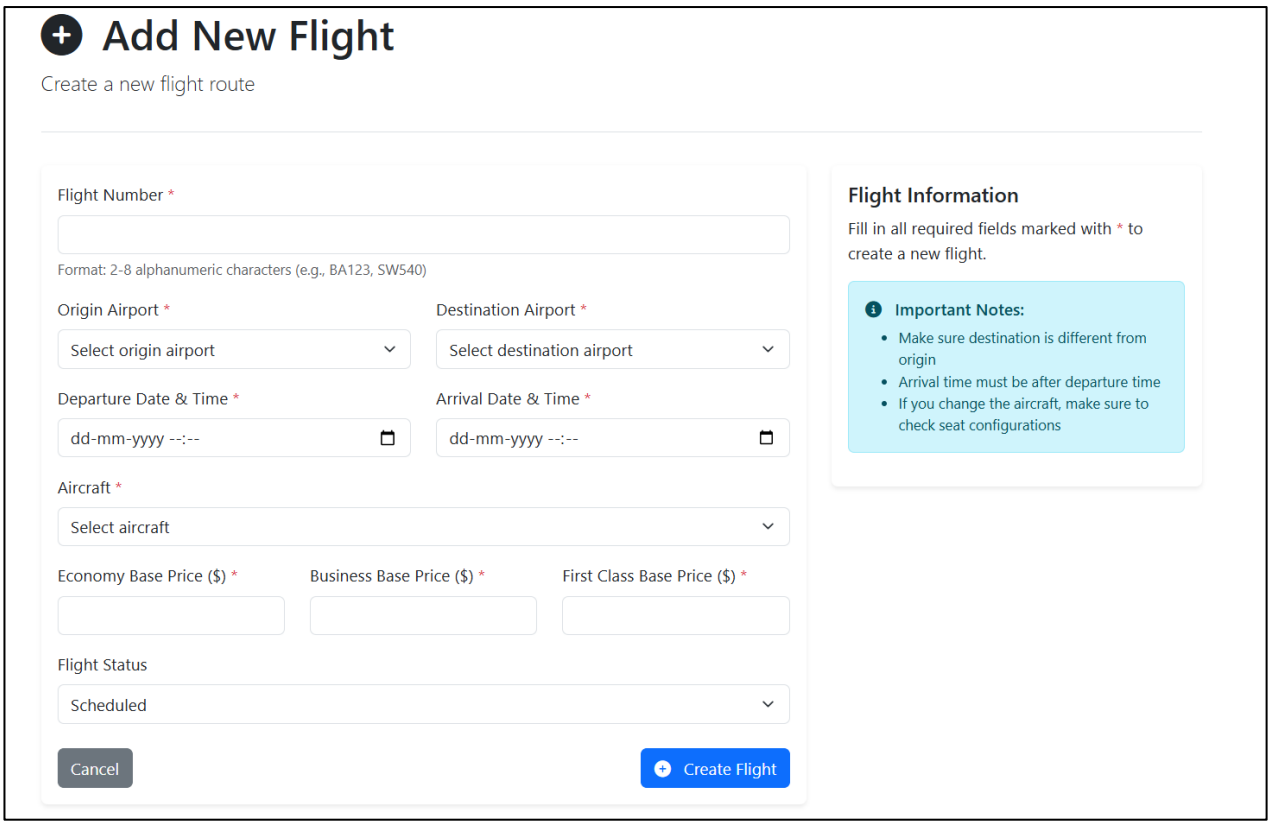
Top Users by Bookings

1. Samantha Hess (@mjohnson7) 784
2. Jason Bennett (@kelsey3320) 780
3. Javier Martinez (@holtsusan42) 780
4. William Navarro (@travis2867) 778
5. Catherine Rodriguez (@crawforddana4) 777
6. Lauren Smith (@thomasgoodman2) 777
7. James Shaw (@dalton3729) 776
8. Danielle Castaneda (@andrea7225) 775
9. Mary Richardson (@johnsonmichael41) 774
10. Robert Williams (@rebecca5634) 774

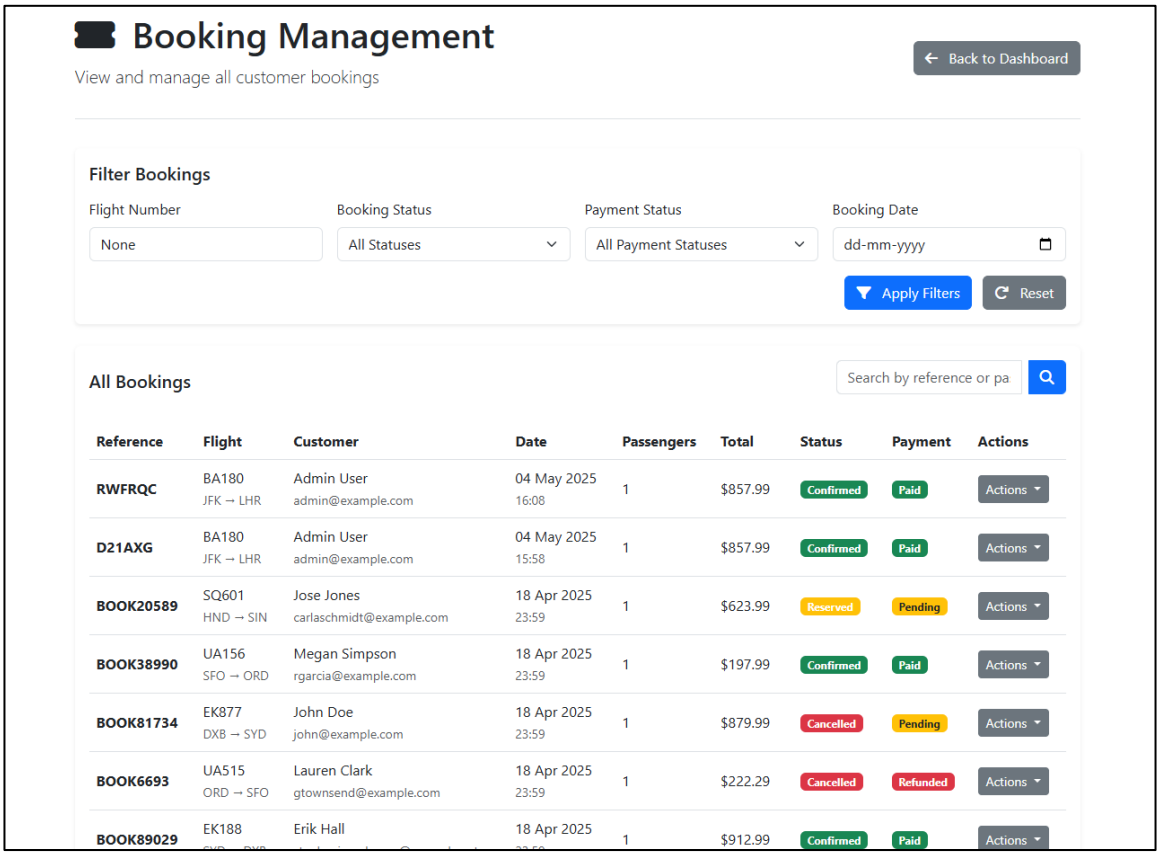
User management:



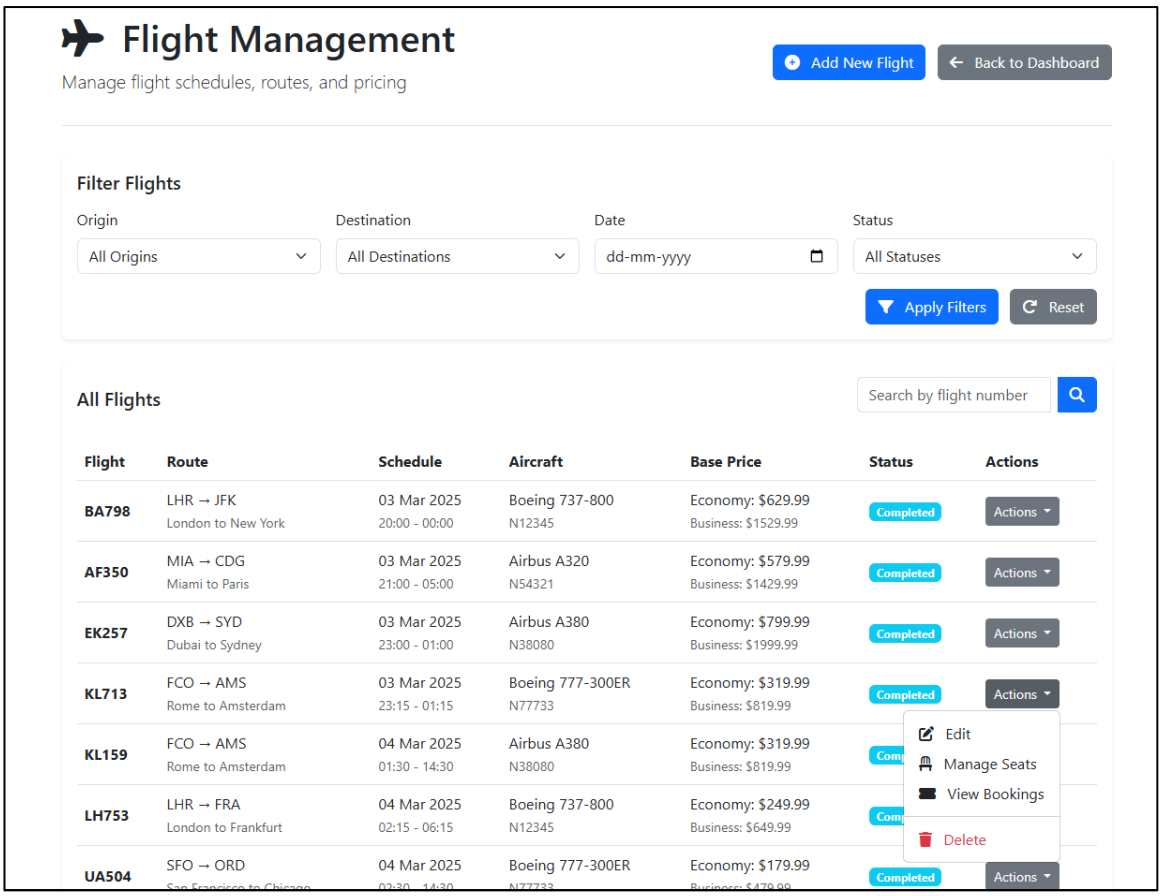
Flight addition interface:



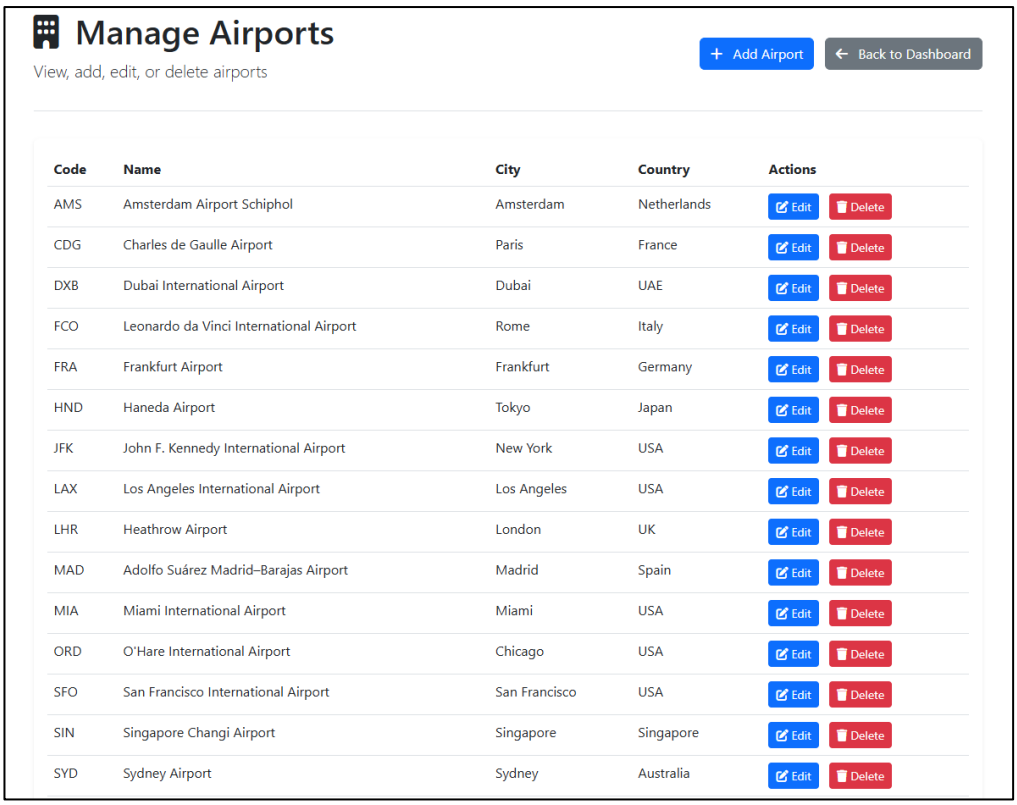
Booking management:



Flight management:

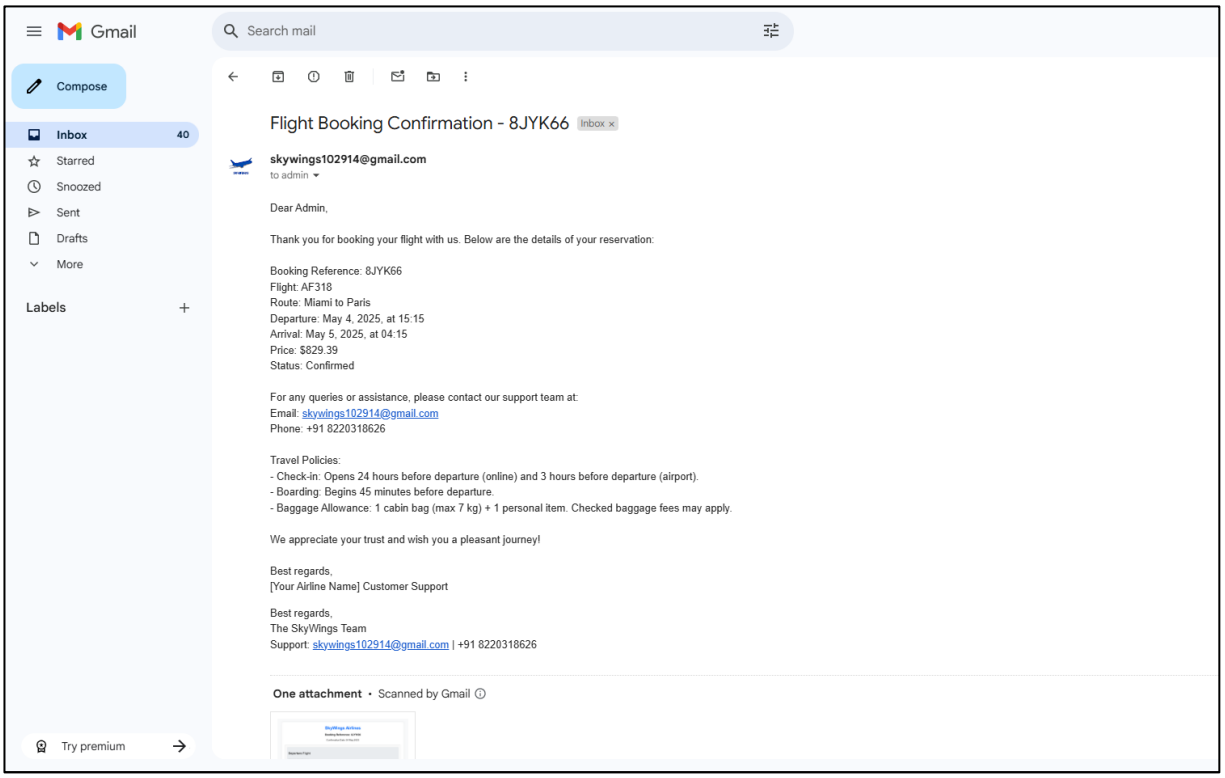


Airport management:

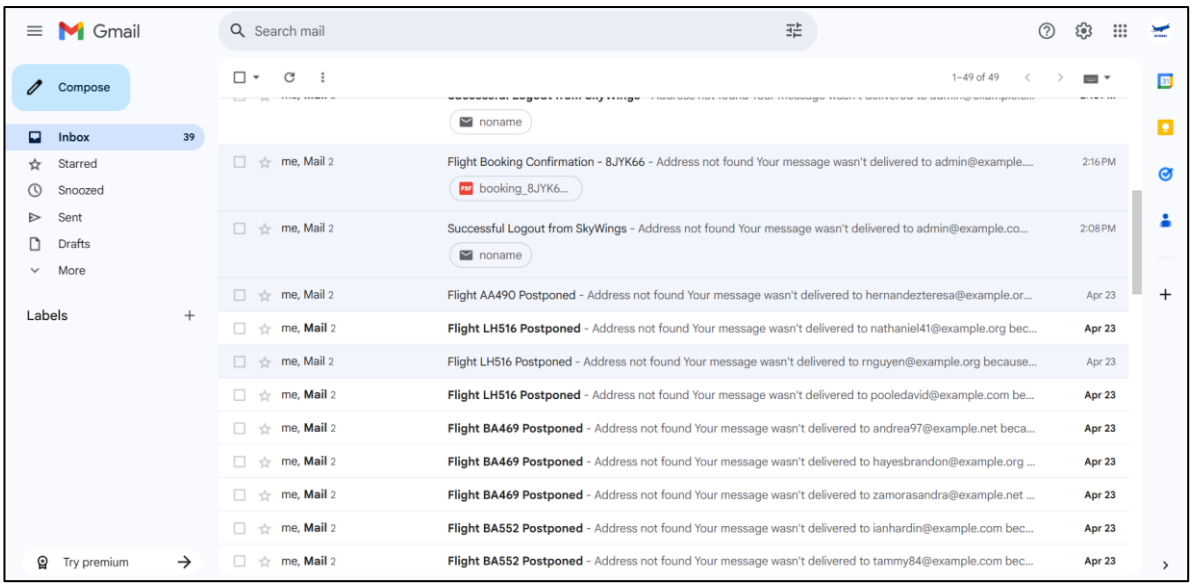


Email :

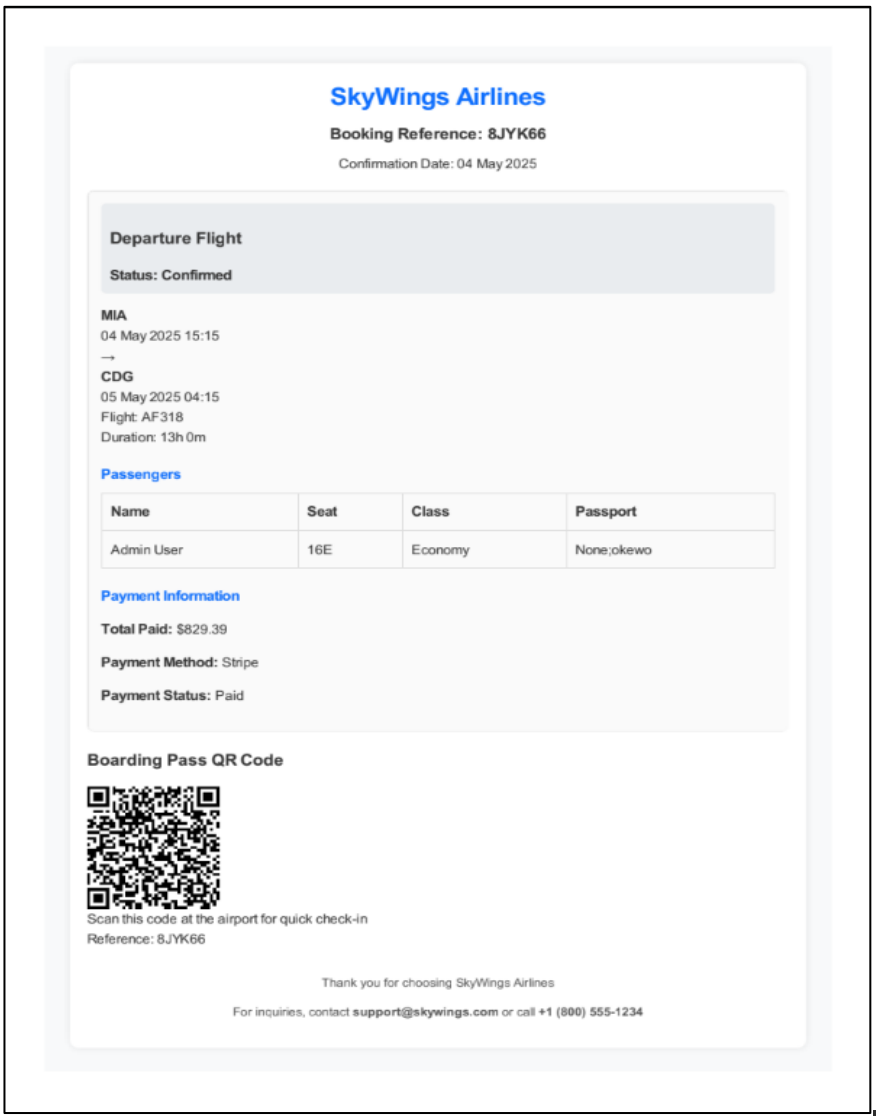
Booking confirmation:



Mail to Customers (Booking Confirmating , Flight Delays , Login , Logout):



Ticket generated:



SkyWings Airlines

Booking Reference: 8JYK66

Confirmation Date: 04 May 2025

Departure Flight

Status: Confirmed

MIA
04 May 2025 15:15
→
CDG
05 May 2025 04:15
Flight: AF318
Duration: 13h 0m

Passengers

Name	Seat	Class	Passport
Admin User	16E	Economy	None;okewo

Payment Information

Total Paid: $829.39

Payment Method: Stripe

Payment Status: Paid

Boarding Pass QR Code

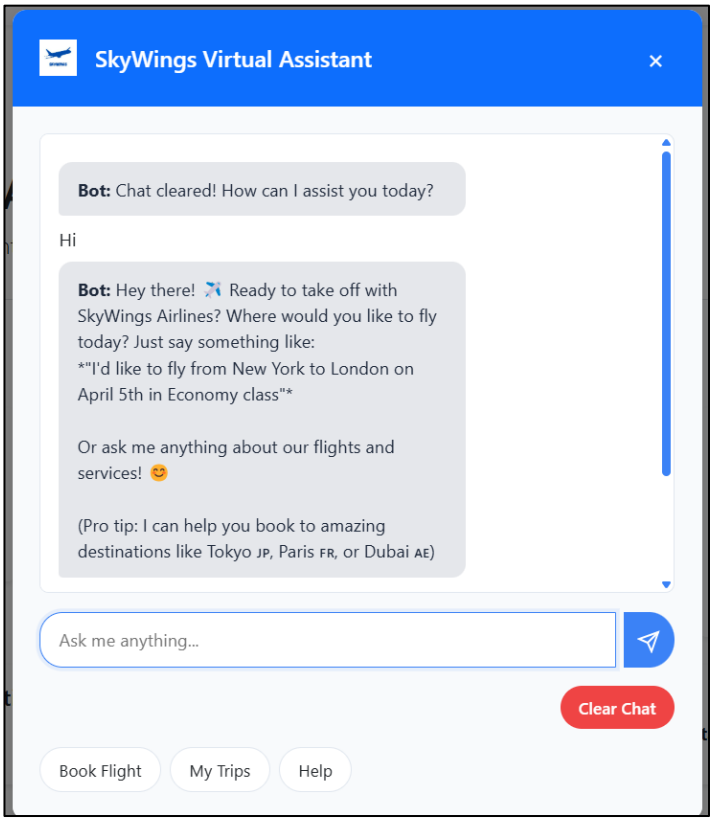
Scan this code at the airport for quick check-in
Reference: 8JYK66

Thank you for choosing SkyWings Airlines

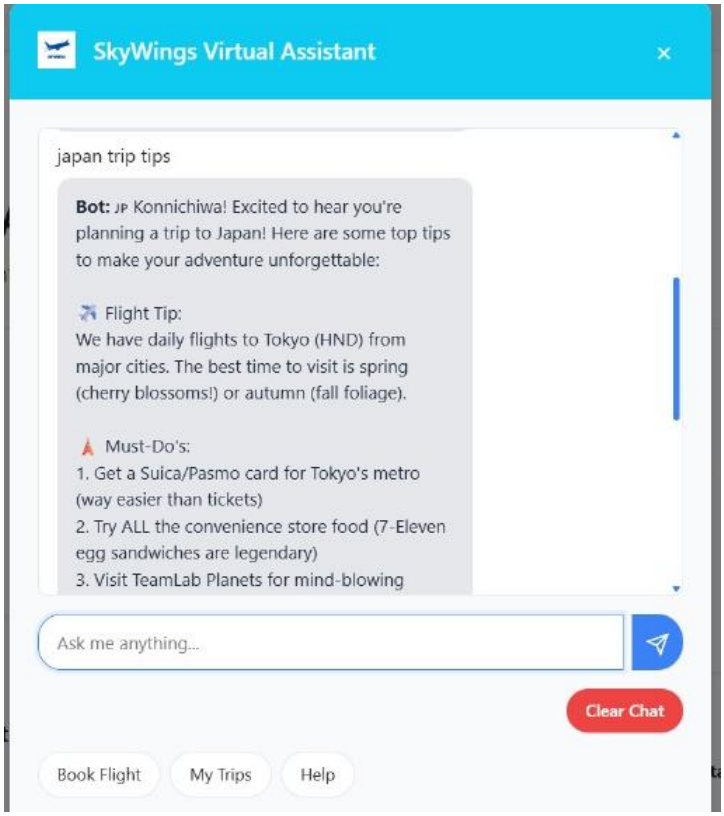
For inquiries, contact **support@skywings.com** or call **+1 (800) 555-1234**

AI Chatbot:

Initial Interaction:



Trip suggestions:

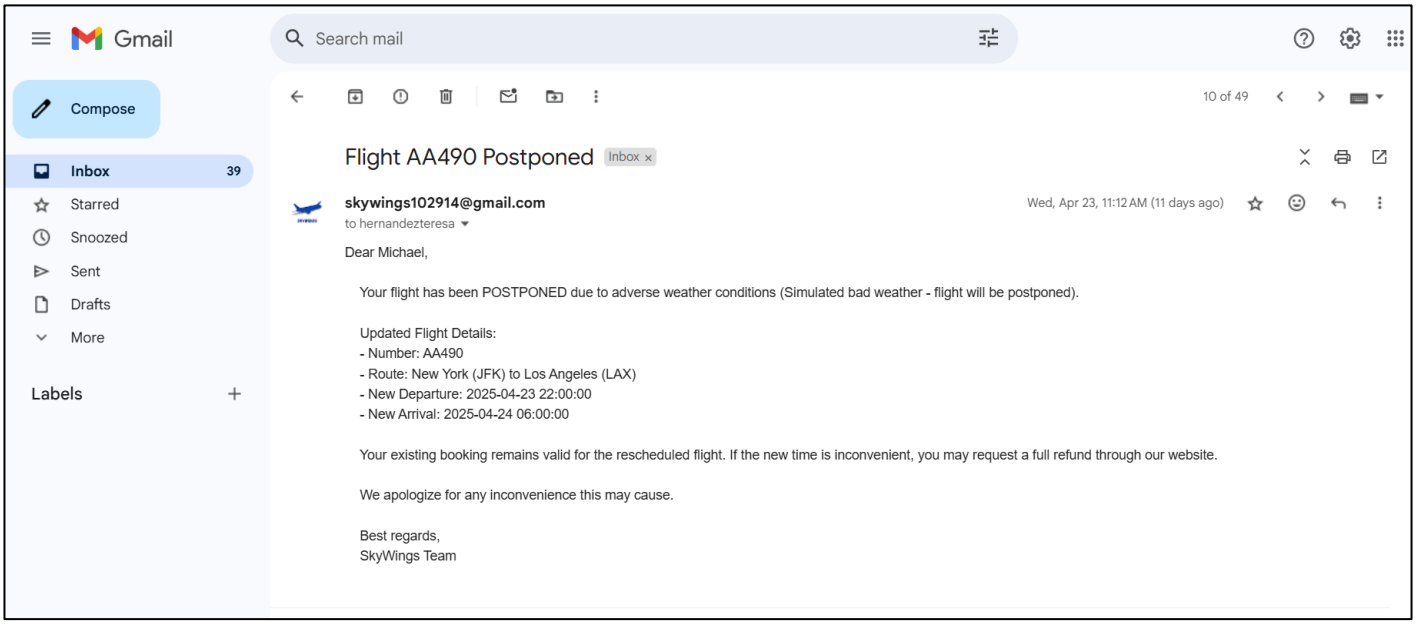


Weather Module:

Weather status checked and updated:

```
  Humidity: 73%
  Visibility: 10000 meters
  Wind: Speed 7.48 m/s, Direction 151°, Gust 9.42 m/s
  Cloud Cover: 100%
  Timestamp: 1746376737
  Sunrise: 1746331410
  Sunset: 1746382322
  Timezone: 7200 seconds
  Location ID: 6693932
  Location Name: Focene
  Country Code: IT
  Response Code: 200
Destination Weather Data:
  Coordinates: (4.7683, 52.3105)
  Weather Condition: overcast clouds
  Icon: 04d
  Temperature: 11.68°C
  Feels Like: 10.5°C
  Temperature Range: 11.68°C (min) to 11.68°C (max)
  Pressure: 1016 hPa (sea level), 1016 hPa (ground level)
  Humidity: 61%
  Visibility: 10000 meters
  Wind: Speed 4.54 m/s, Direction 3°, Gust 6.79 m/s
  Cloud Cover: 97%
  Timestamp: 1746376739
  Sunrise: 1746331458
  Sunset: 1746385860
  Timezone: 7200 seconds
  Location ID: 2747542
  Location Name: Schiphol
  Country Code: NL
  Response Code: 200
2025-05-04 22:08:58,392 [MainProcess:Thread-2 (run_continuously)] - INFO - - Flight KL234: Rome (FCO) to Amsterdam (AMS), Departure: 2025-05-05 06:45:00, Weather S
tatus: Safe
2025-05-04 22:08:58,393 [MainProcess:Thread-2 (run_continuously)] - INFO - Weather monitoring completed: 0 flights updated
2025-05-04 22:08:58,393 [MainProcess:Thread-2 (run_continuously)] - INFO - Weather monitor completed single run
```

Simulated bad weather – Mail sent to customers:



Conclusion

SkyWings is a scalable, AI-enhanced, user-centric reservation platform for modern airline operations.

Summary of Work Done

We developed a web-based airline booking platform, implemented flight and seat management, built a dynamic pricing model, integrated a chatbot, and created an admin panel. The project demonstrates real-world application of full-stack development and system integration.

Key Learnings

- Python Flask development and API integration
- Database schema design
- Use of AI in customer support
- Importance of user interface/UX principles
- Collaborative project workflow with Git

Future Scope

Potential Improvements

- Real-time flight status tracking
- Cloud deployment for scalability
- Internationalization support

Additional Features Planned

- Full-fledged loyalty program
- Integration with payment gateways
- Predictive booking trends using ML

References

- Flask: https://flask.palletsprojects.com
- SQLite: https://sqlite.org/docs.html
- GitHub: https://github.com/VijayaraaghavanKS/Python-T14
- Bootstrap: https://getbootstrap.com
- Email: mailto:skywings102914@gmail.com

Appendix

Code Snippets

Below are representative snippets of critical backend logic for the airline reservation system:

User Authentication (Flask - `auth.py`)

```
@app.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        user = request.form['username']
        password = request.form['password']
        conn = sqlite3.connect('airline.db')
        cursor = conn.cursor()
        cursor.execute("SELECT * FROM users WHERE username = ? AND password = ?", (user, password))
        data = cursor.fetchone()
        if data:
            session['user'] = user
            return redirect(url_for('home'))
        else:
            flash("Invalid credentials")
    return render_template('login.html')
```

Flight Search Logic

```
@app.route('/search', methods=['POST'])
def search():
    source = request.form['source']
    destination = request.form['destination']
    date = request.form['date']
    conn = sqlite3.connect('airline.db')
    cursor = conn.cursor()
    cursor.execute("SELECT * FROM flights WHERE source=? AND destination=? AND date=?", (source,
destination, date))
    flights = cursor.fetchall()
    return render_template('search_results.html', flights=flights)
```

Booking Confirmation

```
@app.route('/confirm', methods=['POST'])
def confirm():
    flight_id = request.form['flight_id']
    seat = request.form['seat']
    user = session['user']
    conn = sqlite3.connect('airline.db')
    cursor = conn.cursor()
    cursor.execute("INSERT INTO bookings (username, flight_id, seat) VALUES (?, ?, ?)", (user, flight_id,
seat))
    conn.commit()
    return render_template('confirmation.html', flight_id=flight_id, seat=seat)
```

API Route Listings

The following routes define the major API endpoints and views in the Flask application:

Method(s)	Route	Description
GET	/	Home page

GET, POST	/chatbot	Chatbot interface and message handling
POST	/clear_chat	Clear chatbot conversation history
GET	/get_chat_history	Retrieve chatbot chat history
GET	/api/airports	Airport autocomplete API
GET	/search_flights	Search for flights by destination
GET, POST	/search	Flight search interface/results
GET	/book_flight/<int:flight_id>	Redirect to search with selected flight
GET	/seat-selection/<flight_id>/<travel_class>/<int:passengers>	Seat selection for a flight
POST	/store-selected-seats/<flight_id>	Store selected seats for a flight
GET	/api/flight/<flight_id>/available-seats/<travel_class>	Get available seats for a flight/class (JSON)
GET	/passenger-details	Passenger details form
POST	/process-passengers	Process passenger details submission
GET	/payment	Payment page
POST	/process-payment	Process payment (Stripe)
GET	/payment-callback	Stripe payment callback/confirmation
GET	/booking-confirmation/<booking_reference>	Booking confirmation page
GET	/my-bookings	View user's bookings
POST	/cancel-booking/<booking_reference>	Cancel a booking
GET	/logout	Log out the user
GET, POST	/login	Login form and login handler
GET, POST	/register	Registration form and handler
GET	/profile	View user profile

POST	/update_profile	Update user profile details
POST	/change_password	Change user password
POST	/save_preferences	Save user preferences
GET	/init-db	Initialize database with sample data
GET	/download-confirmation/<booking_references>	Download booking confirmation as PDF
GET	/email-confirmation/<booking_references>	Email booking confirmation with PDF
GET	/boarding-pass	Display boarding pass for a booking
GET	/admin	Admin dashboard
GET	/admin/flights	Manage flights (admin)
GET, POST	/admin/flights/add	Add a new flight (admin)
GET, POST	/admin/flights/edit/<flight_id>	Edit a flight (admin)
POST	/admin/flights/delete/<flight_id>	Delete a flight (admin)
GET, POST	/admin/flights/<flight_id>/seats	Manage seats for a flight (admin)
GET	/admin/flights/<flight_id>/bookings	View bookings for a flight (admin)
GET	/admin/bookings	View all bookings (admin)
GET	/admin/bookings/<booking_id>	View details of a specific booking (admin)
POST	/admin/bookings/update-status/<booking_id>	Update booking status (admin)
POST	/admin/bookings/update-payment/<booking_id>	Update payment status (admin)
POST	/admin/bookings/cancel/<booking_id>	Cancel a booking (admin)
GET	/admin/airports	Manage airports (admin)
GET, POST	/admin/airports/add	Add airport (admin)
GET, POST	/admin/airports/edit/<airport_id>	Edit airport (admin)

POST	/admin/airports/delete/<airport_id>	Delete airport (admin)
GET	/admin/aircraft	Manage aircraft (admin)
GET, POST	/admin/aircraft/add	Add aircraft (admin)
GET, POST	/admin/aircraft/edit/<aircraft_id>	Edit aircraft (admin)
POST	/admin/aircraft/delete/<aircraft_id>	Delete aircraft (admin)
GET	/admin/users	Manage users (admin)
GET, POST	/admin/users/add	Add user (admin)
GET, POST	/admin/users/edit/<user_id>	Edit user (admin)
POST	/admin/users/delete/<user_id>	Delete user (admin)
POST	/admin/users/toggle-admin/<user_id>	Toggle admin status (admin)
POST	/admin/users/update-ff-status/<user_id>	Update frequent flyer status (admin)
GET	/admin/users/bookings/<user_id>	View bookings for a user (admin)
GET	/admin/user/<int:user_id>	View user profile and bookings (admin)
GET	/admin/reports	Analytics/reports dashboard (admin)
...	Error handlers (404, 500)	Custom error pages

Database Table Structures

SQLite is used as the backend database. The main schema consists of:

Users Table

Field Name	Type	Description
id	INTEGER	Primary key
username	TEXT	Unique username
email	TEXT	Unique email address
password_hash	TEXT	Hashed password
first_name	TEXT	User's first name
last_name	TEXT	User's last name
date_of_birth	DATE	User's birth date
phone_number	TEXT	Contact number
address	TEXT	Physical address
passport_number	TEXT	Passport number

Field Name	Type	Description
nationality	TEXT	Nationality
frequent_flyer_status	TEXT	Status (e.g., Standard, Gold)
frequent_flyer_miles	INTEGER	Total frequent flyer miles
date_joined	DATETIME	Date of registration
is_admin	BOOLEAN	Admin status
booking_notifications	BOOLEAN	Opt-in for booking notifications
promotional_emails	BOOLEAN	Opt-in for promotional emails
newsletter	BOOLEAN	Opt-in for newsletters

Airports Table

Field Name	Type	Description
id	INTEGER	Primary key
code	TEXT	Unique airport code (e.g., JFK)
name	TEXT	Airport name
city	TEXT	City name
country	TEXT	Country name
latitude	FLOAT	Latitude coordinate
longitude	FLOAT	Longitude coordinate

Aircrafts Table

Field Name	Type	Description
id	INTEGER	Primary key
model	TEXT	Aircraft model
registration	TEXT	Unique registration number
economy_seats	INTEGER	Number of economy seats
business_seats	INTEGER	Number of business seats
first_class_seats	INTEGER	Number of first-class seats

Flights Table

Field Name	Type	Description
id	INTEGER	Primary key
flight_number	TEXT	Unique flight number
origin_id	INTEGER	Foreign key referencing Airports
destination_id	INTEGER	Foreign key referencing Airports
departure_time	DATETIME	Flight departure time

Field Name	Type	Description
arrival_time	DATETIME	Flight arrival time
aircraft_id	INTEGER	Foreign key referencing Aircrafts
economy_base_price	FLOAT	Base price for economy seats
business_base_price	FLOAT	Base price for business seats
first_class_base_price	FLOAT	Base price for first-class seats
status	TEXT	Flight status (e.g., Scheduled)
postponed_count	INTEGER	Number of postponements
weather_status	TEXT	Weather condition details
cancellation_reason	TEXT	Reason for cancellation (if applicable)

Seats Table

Field Name	Type	Description
id	INTEGER	Primary key
flight_id	INTEGER	Foreign key referencing Flights
seat_number	TEXT	Seat number (e.g., 12A)
seat_class	TEXT	Seat class (Economy, Business, etc.)
is_booked	BOOLEAN	Booking status

Bookings Table

Field Name	Type	Description
id	INTEGER	Primary key
booking_reference	TEXT	Unique booking reference
user_id	INTEGER	Foreign key referencing Users
flight_id	INTEGER	Foreign key referencing Flights
booking_date	DATETIME	Date of booking
status	TEXT	Booking status (e.g., Reserved)
total_price	FLOAT	Total cost of booking
payment_status	TEXT	Payment status (e.g., Paid, Unpaid)

Booking Details Table

Field Name	Type	Description
id	INTEGER	Primary key
booking_id	INTEGER	Foreign key referencing Bookings
seat_id	INTEGER	Foreign key referencing Seats
passenger_first_name	TEXT	Passenger's first name

Field Name	Type	Description
passenger_last_name	TEXT	Passenger's last name
passenger_dob	DATE	Passenger's date of birth
passenger_passport	TEXT	Passport number
passenger_nationality	TEXT	Nationality
special_requests	TEXT	Special requirements (optional)
price	FLOAT	Seat price for passenger

Payments Table

Field Name	Type	Description
id	INTEGER	Primary key
booking_id	INTEGER	Foreign key referencing Bookings
amount	FLOAT	Payment amount
payment_date	DATETIME	Date of payment
payment_method	TEXT	Method used (e.g., Credit Card)
transaction_id	TEXT	Payment transaction ID (optional)
status	TEXT	Payment status (e.g., Completed)