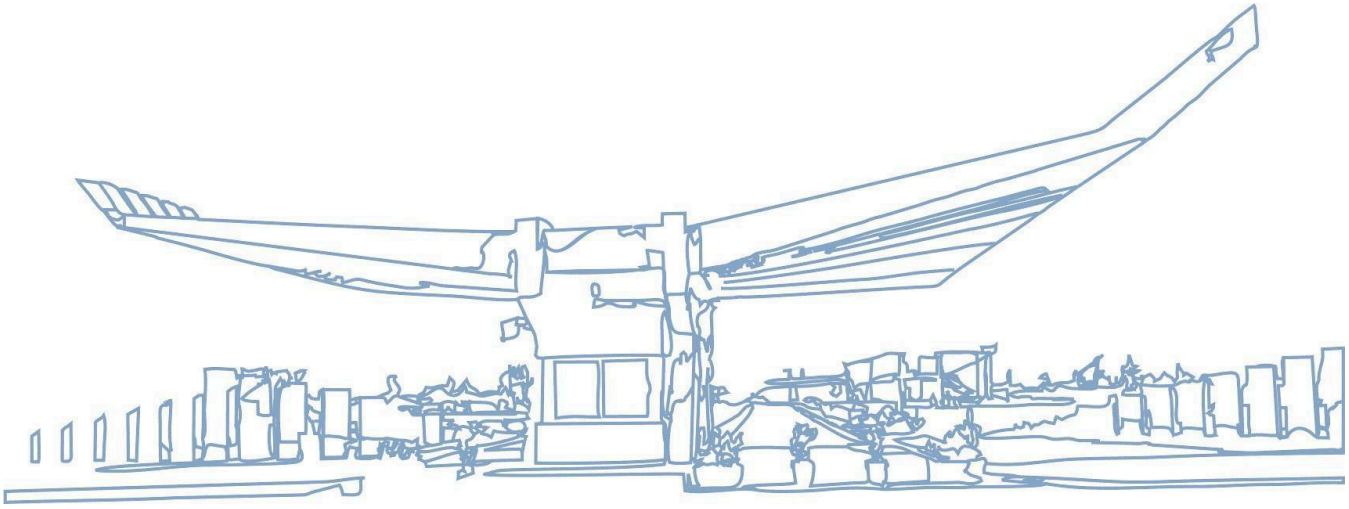
**Software Project Management**

**[Airline Management System - Shqipe Airline]**




**Team Members:**

**Alvin Kollcaku**

**Regi Loshi**

**Tea Meraj**

**Maida Daulle**

**Melis Derveni**

**Aleksander Pacani**

**Rexhens Balla**

**[Shqipe Airline]**
**Requirements Specification**

The Shqipe Airline Management System is a comprehensive software solution designed to optimize airline operations by simplifying flight booking, reservation management, flight tracking, and administrative control. It enhances efficiency for passengers while providing powerful management tools for airline administrators. The system integrates secure payment processing, detailed reporting, and effective communication to ensure a seamless and reliable user experience.

EPOKA UNIVERSITY

## Table of Contents

# 1. Executive Summary

## 1.1 Project Overview

This project focuses on developing a comprehensive airline ticket booking system aimed at enhancing the flight booking experience for passengers. The system is designed to provide a user-friendly platform for booking flights, managing reservations, processing secure transactions, generating detailed reports, and facilitating effective communication among users. The goal is to ensure a seamless, efficient, and streamlined experience for all involved.

**Intended Audience**

The intended audience for this software includes:

- **Passengers**: Individuals who need to search for flights, book tickets, manage their bookings, check flight status, and provide feedback on their flight experiences. They will interact with the system through a user-friendly interface that supports all stages of the booking process.
- **Administrators**: Airline staff responsible for managing user accounts, including account creation, modification, deletion, password resets, and role/permission management. Administrators will also oversee system security, reporting, and overall maintenance.
- **Flight Planners (Air Control Department)**: Personnel in charge of managing flight schedules, including creating, modifying, and deleting flights. Flight planners will also update flight details (e.g., flight status, pricing, and departure times) to ensure that all data is accurate and up to date.

## 1.2 Purpose and Scope of this Specification

**Purpose**

The purpose of this specification is to detail the functional and technical requirements for the development of our airline ticket booking software. This document aims to provide a clear and comprehensive guide for everyone involved in the project, ensuring that all parties understand the scope and limitations of the project.

**Scope**

**In Scope**

This document addresses requirements related to the development of the airline ticket booking software,

specifically:

**Passenger Services**:

- Account creation and login functionality.
- Flight search, booking, and management.
- Payment processing integration and feedback system for passengers to rate their flight experience.

**User Registration and Authentication**:

- Secure account management for passengers.
- Administrative controls for managing user accounts.

**Transactions Reporting**:

- Access to revenue performance reports.
- Data visualization tools for revenue analysis.

**Flight Planning**:

- Creation, modification, and deletion of flights.
- Management of flight details including coordinates, aircraft information, and routes.

**Management and Reporting**:

- Oversight of user accounts and system performance.
- Generation of reports and analytics for flight bookings.

**Customer Support and Communication**:

- Provision of customer support.

**Out of Scope**

The following items are outside the scope of these specifications:

**Future Phases of Development**:

- Any enhancements or modifications planned for future phases of the project, beyond the initial

deployment.

- Requirements related to the integration with third-party systems or services that are not part of the current phase.
- Modifications related to future legislative mandates not covered in the current phase.

These out-of-scope items will be considered in the planning of future phases, but their requirements will be documented separately and addressed in subsequent development cycles.

## *2. Product/Service Description*

**Background Information**

The airline ticket booking software is designed to address the evolving needs of the airline industry and its customers. As the travel industry becomes increasingly digitized, there is a growing demand for efficient, user-friendly, and secure booking solutions. This software aims to enhance the travel experience for passengers while providing robust management tools for airline staff and departments. The following factors influence the product and its requirements:

A. **User Experience**:
   - **Ease of Use**: Passengers expect a seamless and intuitive booking process. A user-friendly interface that simplifies flight searches, bookings, and payments is essential to meet customer expectations and enhance satisfaction.
   - **Accessibility**: The software must be accessible across various devices, including desktops, tablets, and smartphones, to cater to a broad audience.

B. **Security**:
   - **Data Protection**: With the increasing risk of cyber threats, safeguarding personal and payment information is crucial. The software must implement robust security measures to protect user data and ensure compliance with data protection regulations.
   - **Authentication**: Secure user authentication processes are necessary to prevent unauthorized access and ensure the integrity of user accounts.

C. **Real-Time Information**:

- **Flight Availability and Pricing**: Providing real-time data on flight availability, pricing, and seat options is vital for passengers to make informed decisions. This requires integration with airline databases and real-time data feeds.
- **Dynamic Updates**: The software should handle dynamic updates, such as changes in flight schedules or prices, to ensure users have the most current information.

**D. Operational Efficiency**:

- **Administrative Control**: Airline staff need effective tools to manage user accounts, reservations, and flight details. This includes functionalities for creating, modifying, and deleting accounts and flights, as well as handling special requests and changes.
- **Financial Reporting**: Detailed revenue reports and data visualization tools to monitor and analyze financial performance. This helps in making informed business decisions and identifying trends.

**E. Regulatory Compliance**:

- **Legislative Requirements**: The software must comply with industry regulations and standards, such as data protection laws and aviation regulations. This ensures the system operates within legal frameworks and avoids penalties.
- **Accessibility Standards**: Compliance with accessibility standards ensures the software is usable by people with disabilities, broadening its user base and adhering to legal requirements.

**F. Customer Support**:

- **Multi-Channel Support**: Providing customer support through various channels, including phone, and email is essential for addressing passenger inquiries and issues promptly.

**Reasons for Specific Requirements**

The aforementioned factors directly influence the specific requirements detailed later in the specification. For instance, the need for a user-friendly interface drives the requirement for comprehensive search and booking functionalities. Security considerations necessitate robust authentication mechanisms and data protection measures. Real-time information requirements lead to the integration of dynamic data feeds and updates. Operational efficiency and regulatory compliance shape the administrative and reporting features, while customer support needs influence the communication tools and notification systems.

By understanding these general factors, stakeholders can appreciate the rationale behind each requirement, ensuring the final product aligns with industry demands and user expectations.

## *2.1 Product Context*

**Independence and Self-Contained Nature**

The airline ticket booking software can function as an independent, self-contained system with its primary purpose being to facilitate the booking of airline tickets for passengers.

**Interfacing with Related Systems**

While the core functionality of the software is self-contained, it often needs to interface with a variety of related systems to provide a comprehensive service. These relationships include:

**Payment Gateways**

- **Description:** Integration with payment processing services like PayPal, Stripe, or direct credit card processors.
- **Function:** Enables secure and efficient handling of payments, supporting multiple currencies and payment methods.

**Travel Management Systems**

The system will integrate flight bookings with hotel booking and car rental services, providing a seamless experience for travelers. It will help users book their accommodation and rental vehicles alongside their flights, ensuring that all travel needs are addressed in one platform. The integration will streamline the booking process, reduce the time spent on separate bookings, and provide an efficient, all-in-one solution for passengers. Additionally, the system will offer features such as:

- Booking and managing hotel reservations, including details like room types, amenities, and pricing.
- Integration with car rental services, allowing users to select rental options based on their travel plans and preferences.

**Security and Compliance Systems**

- **Description:** Connecting with systems for data security, fraud detection, and regulatory compliance.

- **Function:** Ensures the software adheres to industry standards and protects user data.

### *2.2 User Characteristics*

**Staff**

- **Experience:**
  - Moderate experience with booking flights, typically for business travel.
  - Familiar with corporate travel policies and budget constraints.
- **Technical Expertise:**
  - Comfortable with using computers and mobile devices for booking travel.
  - Generally prefer user-friendly interfaces with some support for complex bookings.
- **Other General Characteristics:**
  - Interested in cost-effective options.
  - Value clear policies on changes and cancellations.
  - Require detailed invoicing and receipts for reimbursement purposes.
  - Often need assistance with group bookings and coordination for team travel.

**Leisure Travelers (Passengers)**

- **Experience:**
  - Wide range of experience, from occasional travelers to frequent vacationers.
  - Typically book flights for vacations, family visits, and personal trips.
- **Technical Expertise:**
  - Varies from novice to proficient with online booking platforms.
  - Comfortable with using both websites and mobile apps.
- **Other General Characteristics:**
  - Budget-conscious, looking for deals and special offers.
  - Flexible with travel dates to find better deals.
  - Prefer easy-to-navigate interfaces and straightforward booking processes.

**Business Travelers (Non-Staff)**

- **Experience:**
  - High experience with frequent travel for business purposes.
  - Focus on efficiency and convenience in booking travel.
- **Technical Expertise:**
  - Proficient with using technology for booking flights and managing itineraries.
  - Often use travel management apps and software.
- **Other General Characteristics:**
  - Prioritize efficiency, reliability, and convenience.
  - Interested in options that offer flexibility and ease of booking.
  - Require flexible booking policies and easy management of bookings.
  - Value quick support for changes and cancellations.

## *2.3 Assumptions*

**Assumptions Affecting Requirements**

**Equipment Availability**

- **Assumption:** Users have access to modern computers, tablets, or smartphones.
- **Impact:** The system must be compatible with multiple device types and screen sizes.
- **Change Needed if Unavailable:** The system would need a simplified version for older or less capable devices.

**Operating System**

- **Assumption:** Users are using up-to-date versions of major operating systems (Windows, macOS).
- **Impact:** The software should be compatible with the latest and several previous versions of these operating systems.
- **Change Needed if Unavailable:** Broader compatibility testing and support for older operating system versions would be required.

**Internet Access**

- **Assumption:** Users have reliable internet access.
- **Impact:** The system can be primarily web-based, requiring constant internet connectivity.
- **Change Needed if Unavailable:** Offline capabilities or downloadable versions of the system might need to be developed.

**Browser Compatibility**

- **Assumption:** Users have access to and use modern web browsers (Chrome, Firefox, Safari, Edge).
- **Impact:** The system should be tested and optimized for these browsers.
- **Change Needed if Unavailable:** Additional support for older or less common browsers would be required.

**Technical Expertise**

- **Assumption:** Users have a basic level of technical expertise and can navigate web applications.
- **Impact:** The user interface should be intuitive and user-friendly, but it can assume a baseline understanding of web usage.
- **Change Needed if Unavailable:** Additional user training resources, tutorials, or a simplified interface would be necessary.

**Payment Methods**

- **Assumption:** Users have access to common online payment methods (credit/debit cards, digital wallets like PayPal).
- **Impact:** The system can integrate standard payment gateways.
- **Change Needed if Unavailable:** Support for alternative payment methods or manual payment processes would be required.

**Security Standards**

- **Assumption:** The system will comply with standard security protocols (SSL/TLS, PCI-DSS for payments).
- **Impact:** Ensures user data protection and secure transactions.
- **Change Needed if Unavailable:** Development of custom security measures or additional compliance checks.

**Regulatory Compliance**

- **Assumption:** The system adheres to relevant aviation and data protection regulations (GDPR, CCPA, etc.).
- **Impact:** Ensures legal compliance and user trust.
- **Change Needed if Unavailable:** Significant modifications to data handling, storage, and user consent processes.

## *2.4 Constraints*

This section outlines the limitations and restrictions that impact the development, deployment, and operation of the airline ticket booking software.

1. **System Performance:**
    - The system must be capable of handling a high volume of concurrent users, especially during peak travel seasons. The system's performance may be constrained by the processing power and resources available during these periods.
2. **Data Security and Privacy Regulations:**
    - The software must adhere to industry regulations and other regional data protection laws.
3. **Budget and Time Constraints:**
    - The project is subject to a fixed budget and timeline. These constraints may affect the scope of features that can be implemented and the depth of testing conducted.
4. **Third-Party Integrations:**
    - The integration with third-party services such as payment gateways, hotel booking platforms,

and car rental systems may impose technical or business constraints. These may include API limitations, service reliability, or licensing costs.

5. **Scalability:**
    - While the system is designed to scale, hardware limitations and the complexity of managing large amounts of data could present constraints, particularly during the initial phase of deployment.

### *2.5 Dependencies*

This section outlines external factors or systems that the software relies on to function correctly.

1. **Third-Party Services:**
    - The system depends on third-party services like PayPal, Stripe, and hotel/car rental APIs. Any changes, downtimes, or updates to these services may impact the functionality of the software.
2. **Database Management System (DBMS):**
    - The software relies on a MySQL database for storing user data, booking information, payment details, etc. The functionality of the system depends on the stability and performance of the DBMS.

## *3. Requirements*

### *3.1 Functional Requirements*

***Landing Pages -*** Landing pages are specifically designed to guide visitors through targeted content with the goal of facilitating specific actions or conversions.

- **Home -** The landing page that introduces the website and provides navigation to all other sections.

- **Job Application + Job Application Form -** A section where users can browse and apply for available job positions. An interactive form for applicants to submit their personal details and job preferences.
- **About Us -** A page providing information about the company's background, mission, and values.
- **Aircraft Details -** Displays specifications, features, and visual galleries of available aircraft.
- **Minishop -** Allows users to browse and select items for purchase.
- **Shopping Cart -** Selected products are stored for review before checkout.
- **Payment -** Processes user payments securely for items in the shopping cart.

***Passenger/User -*** A passenger is a customer who books flights.

***Authentication & Account Management***

- **Log in / Sign up:** Allows users to access or register their accounts.
- **Profile management:** Lets users update personal information and preferences.
- **Edit Details (Name, Surname, Birthday, Email):** Permits users to modify their personal information.
- **Change Password:** Lets users change their current password to a new one.

***Flight Booking & Management***

- **Search for Flights (by destination, departure, arrival, date):** Allows passengers to find flights based on their travel preferences.
- **System Integration (Connect with weather data systems):** Integrates with weather data systems to receive real-time weather updates, enabling the system to adjust flight plans as necessary.
- **Book Flights:** Enables passengers to select flight class, seat number, and optionally customize meal preferences, add extra baggage (for an additional fee), and purchase travel insurance.
- **View Ticket Details and Status (confirmed, pending, canceled):** Provides passengers with access to booking information and the current status of their tickets.

***Payments & Transactions***

- **Secure Payment Processing (Credit/Debit Cards):** Ensures users can securely make payments using credit cards.
- **Payment Summary:** Displays an overview of the total amount.
- **Payment Methods:** Lists available payment options, such as credit cards, debit cards, and other supported methods.
- **View Invoices:** Allows users to access and download their invoices and e-tickets, with copies sent to their registered email.
- **Purchases Summary Table:** Presents a detailed table of all completed purchases for easy tracking and reference.

***Flight & Travel History***

- **Filter flights by status:** Allows users to filter flights based on their status (e.g., confirmed, pending, canceled) for easier tracking and management.

***Customer Service & Feedback***

- **Submit Feedback and Rate Flight Experience:** Passengers can provide feedback and rate their flight experience, helping airlines improve services.
- **View All:** Allows users to view all feedback and ratings they've submitted.
- **Delete:** Provides users the option to remove their previously submitted feedback or rating.
- **Select Flight Rate Comment:** Enables users to select and edit specific comments from their feedback or rating submissions.

***Additional Features***

- **Book hotels and rental cars through integrations:** Users can book hotels and rental cars directly through integrated services, offering a seamless travel experience.

***Admin -*** The admin oversees the entire airline management system.

***User & Account Management***

- **Manage User Roles and Permissions:** Allows the admin to assign, modify, and remove user roles and permissions, managing access to various system features. Admins can also add new users, provide details and roles, and delete users if necessary.
- **Search Users by All Features:** Enables the admin to search for users based on multiple attributes.

***System Monitoring & Reporting***

- **Statistics:** Displays key metrics including revenue, flights, and popular destinations.
- **Filter By:** Allows filtering of data by time periods such as last week, last month, last 6 months, and last year.
- **Download and Generate PDF:** Enables downloading and generating comprehensive PDF reports containing all relevant data.

**Access Detailed Reports:** Provides in-depth reports including:

- Flight Reports (delays, cancellations, and trends)
- Financial Reports (revenue, losses, and expenses)

**Dashboard:** Visualizes key performance indicators such as tickets sold, revenue growth, and customer growth for the last month using charts and graphs.

**View Flights:** Shows lists of completed flights, active flights, canceled flights, and total revenue generated.

**Feedback Management:** Displays passenger feedback with search functionality by flight, ID, or keyword comments.

**Transaction Management:** Allows viewing of all transactions, updating transaction status (pending, completed, canceled), deleting transactions, and searching transactions by customer name, airline, status, or route.

***Air Control Department -*** This department is responsible for managing flights and air traffic.

*Flight Management*

- **Add Flights:** The department can schedule new flights by entering necessary flight details.
- **Update Flights:** The department can modify existing flight information and update flight status as needed.
- **Remove Flights:** The department can cancel or delete flights when required.
- **Assign Flight Numbers, Routes, and Schedules:** The department assigns flight numbers, defines routes, and sets flight schedules to optimize operations and efficiency.
- **Update Flight Capacity and Available Seats:** The department manages flight capacity and maintains accurate records of available seats for passengers.

*Dashboard*

- **View All Flights:** Allows users to access and browse the complete list of scheduled flights.
- **View All Passenger Records:** Enables users to access and review all passenger information and booking details.

*System Integration*

- **Connect with weather data systems for real-time updates:** The department connects with weather data systems to receive real-time updates on weather conditions, helping adjust flight plans as needed.

*Flight Booking & Management*

- **Search for Flights (by destination, departure, arrival, date):** Allows passengers to find flights based on their travel preferences.
- **System Integration – Weather Data:** Integrates with weather data systems to receive real-time updates on weather conditions, enabling the department to adjust flight plans accordingly.
- **View Ticket Details and Status (confirmed, pending, canceled):** Provides passengers with access to their booking information and the current status of their tickets.

## *3.2 Non-Functional Requirements*

**Performance Requirements**

- The system should be able to handle at least 1000 concurrent users without performance degradation.
- Flight search results should be displayed within 3 seconds after user input.
- Payment processing should be completed within 5 seconds for a seamless user experience.

**Security Requirements**

- Data encryption must be applied to all sensitive information (passwords, payment details).
- The system must implement role-based access control (RBAC) to restrict staff permissions.
- Implement automatic session timeouts for idle users after 10 minutes.

**Usability Requirements**

- A clear and intuitive user interface (UI) with minimal learning curve should be provided.

**Maintainability & Support**

- Support for API integrations with third-party services like payment gateways and hotel booking and vehicle and whether platforms.

**Compliance & Legal Requirements**

- The system should comply with aviation regulations from organizations like IATA and FAA.
- Secure handling of financial transactions as per PCI-DSS requirements.
- User data privacy should be maintained as per GDPR and CCPA.

**Backup & Disaster Recovery**

- Redundant database storage to prevent critical data loss.

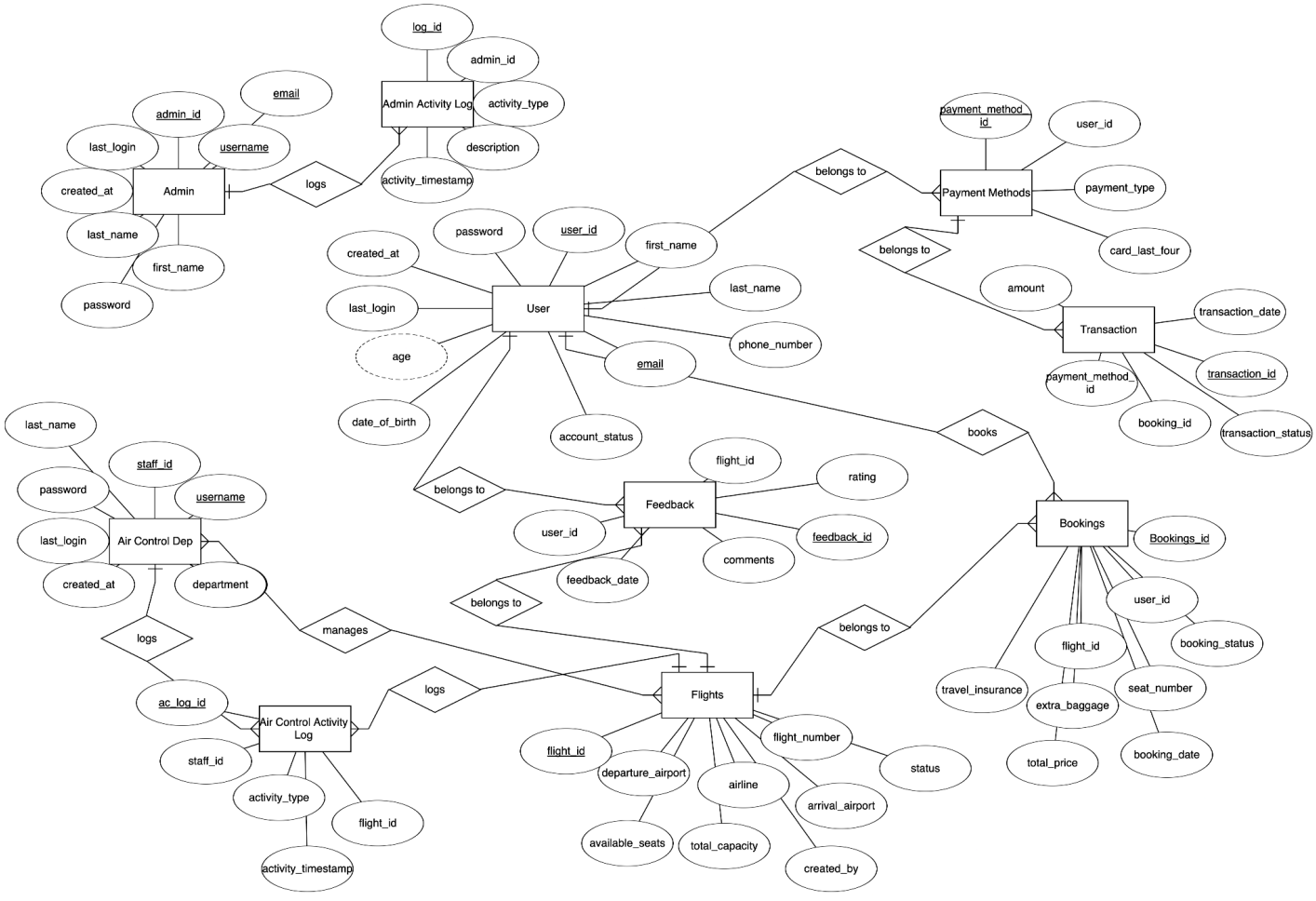
**Interoperability**

- The system should be able to integrate with third-party services such as:
    - Payment gateways (Visa, MasterCard)
    - Weather forecasting APIs
- Standardized APIs should be provided for future expansion.

# *4. Diagram/Schemas*

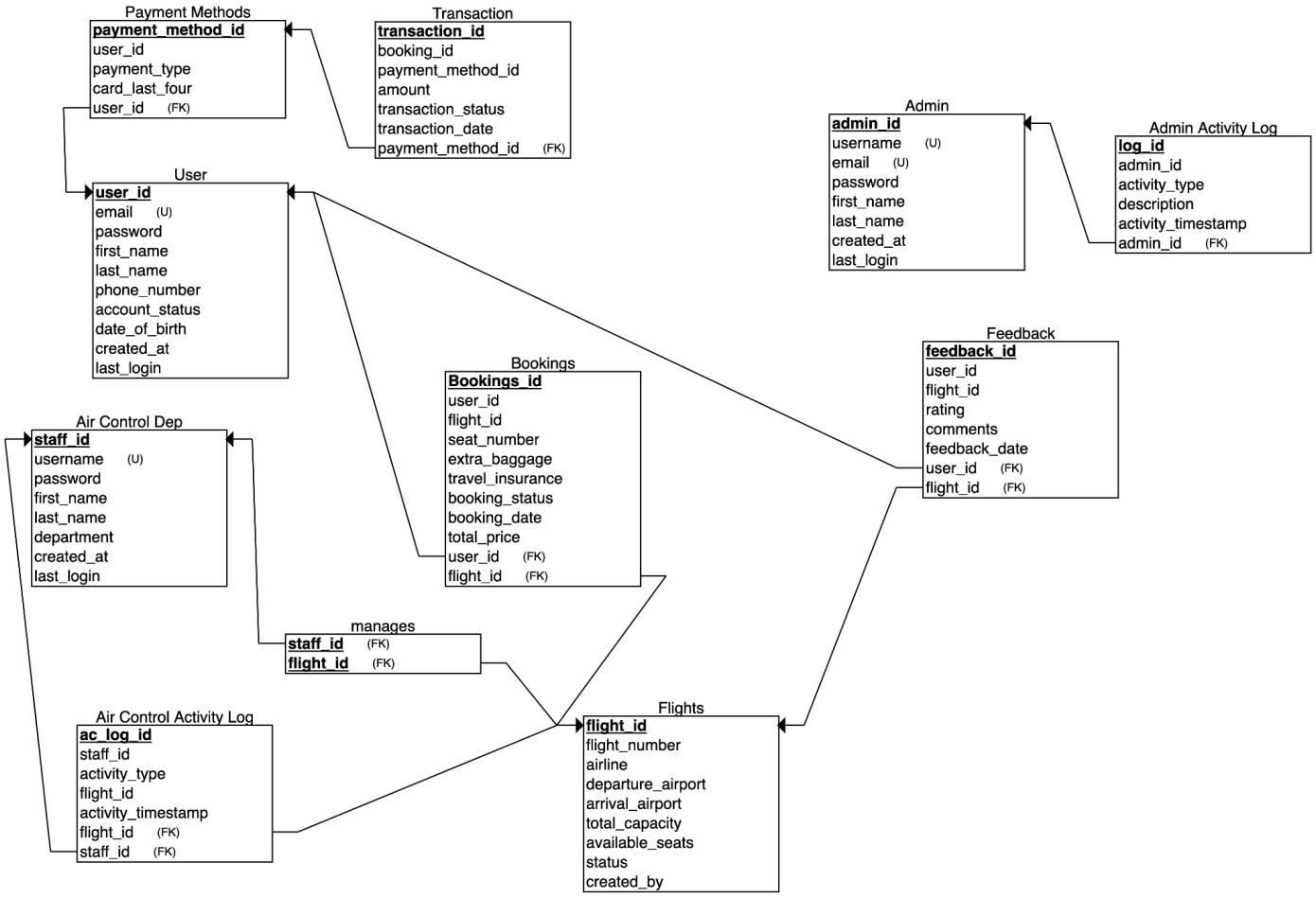
## *4.1 Entity Relationship Diagram (ERD)*

A visual representation of the relationships between entities in a database. It shows how data entities such as passengers, flights, bookings, and payments relate to each other within your system.



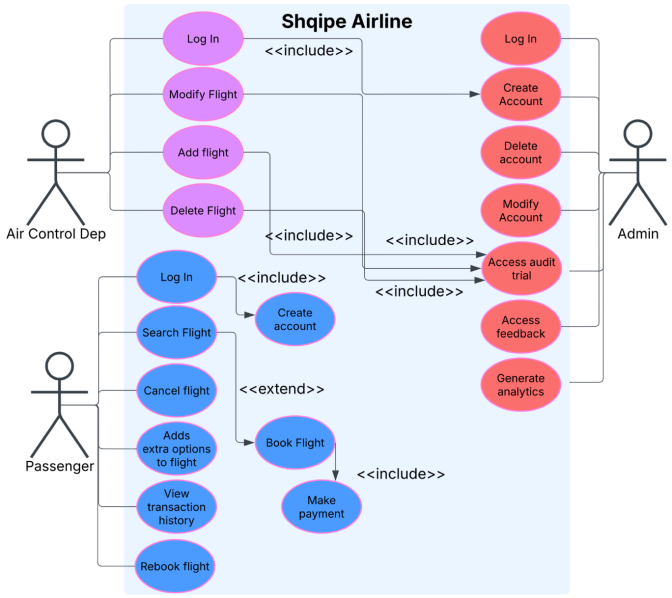
## *4.2 Relational Schema (RS)*

A representation of the database structure, showing how data is organized into tables, along with their attributes (columns) and the relationships between those tables. Each table corresponds to an entity, and each attribute corresponds to a property of that entity. The relationships between the entities (tables) are represented using foreign keys.



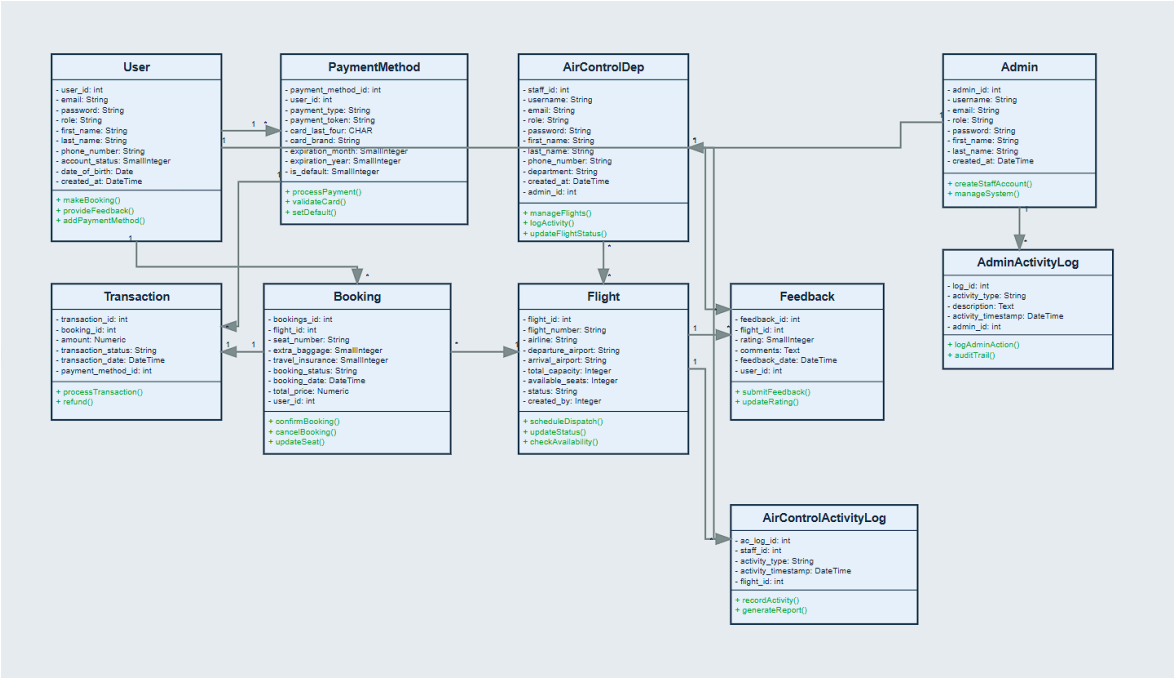

## *4.3 Use Case Diagram*

A representation of the functionality of a system from a user's perspective. It shows the interactions between actors (users or external systems) and the system.



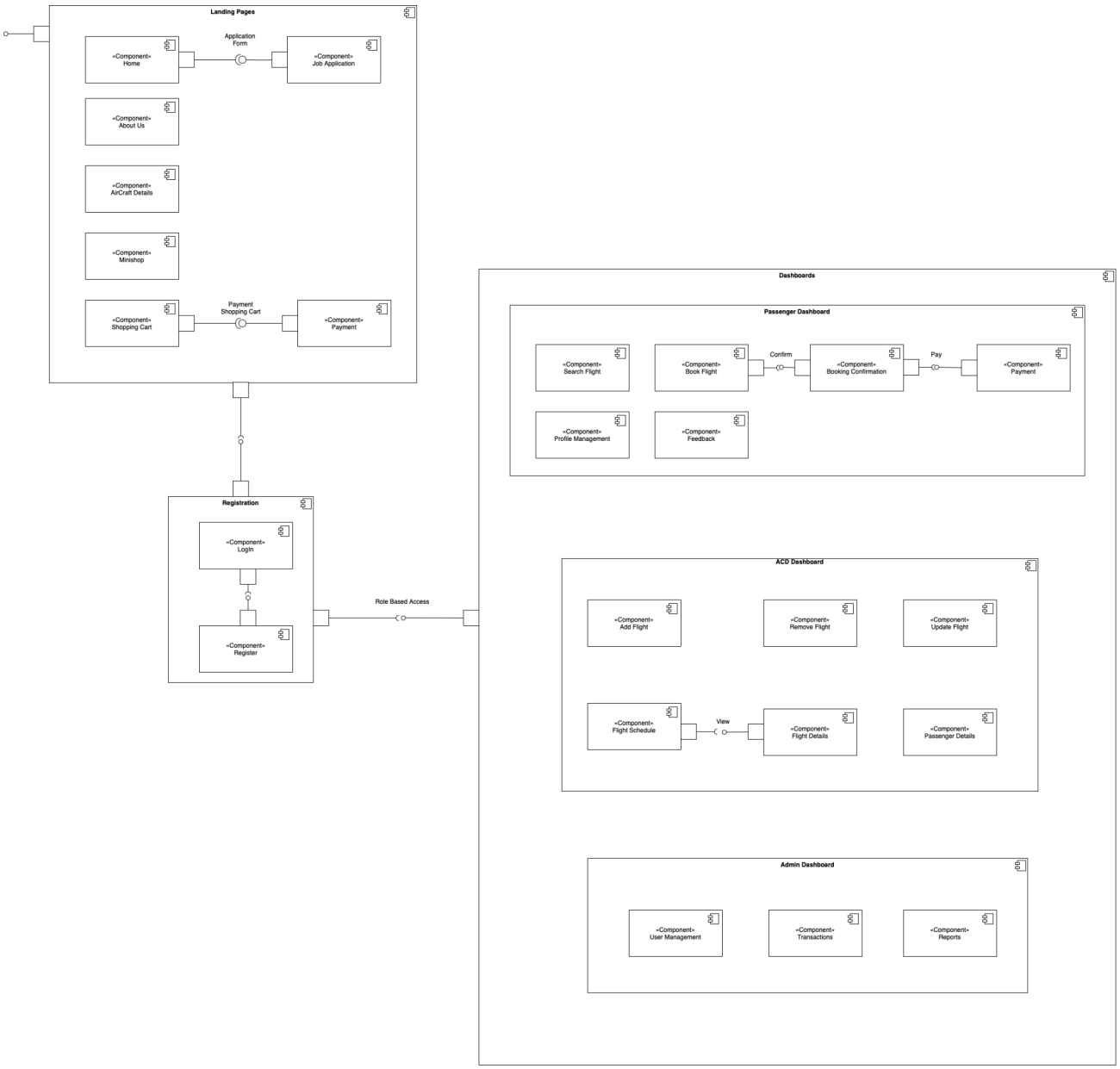

## *4.4 Object Diagram*

The Object Diagram visually represents instances of classes (objects) in a system at a specific moment, showing their attributes and relationships. It helps illustrate the system's state by depicting how objects interact and connect during runtime.



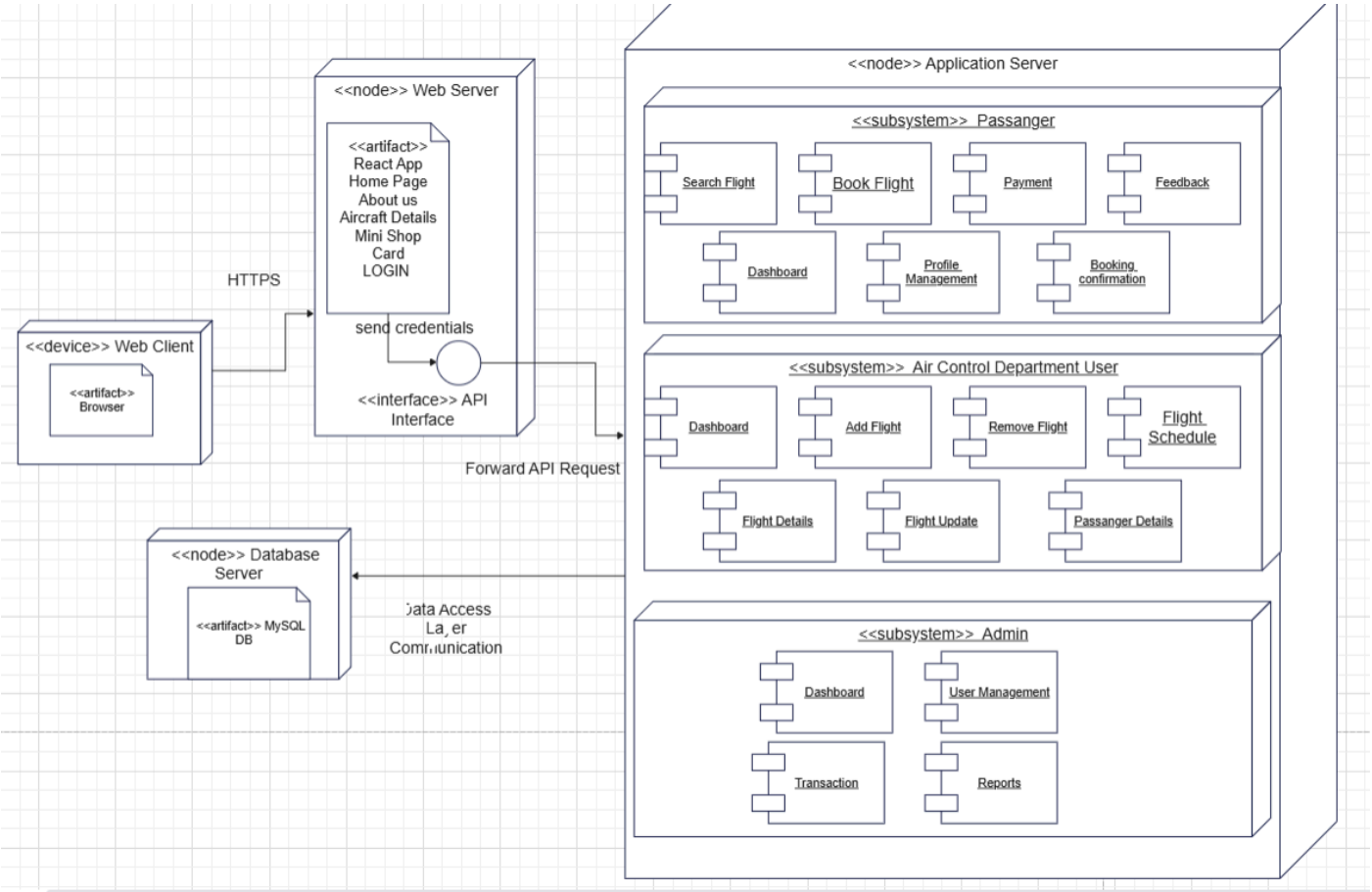
## *4.5 Component Diagram*

The Component Diagram models the organization and dependencies among software components, illustrating how different parts of a system are assembled and interact. It shows components, their interfaces, and relationships, helping to visualize the system's modular structure and deployment.



### *4.6 Deployment Diagram*

The Deployment Diagram shows the physical arrangement of hardware and software in a system, illustrating how software components are deployed across servers, devices, or nodes. It helps visualize the system's runtime architecture and communication between hardware elements.




## *5. Database*

### *5.1 Database Design*

The database design for the Shqipe Airline Management System is crucial for efficiently managing operations such as booking tickets, managing flight schedules, storing customer details, and handling various

logistical aspects of the airline business. The system consists of multiple entities that need to be well-organized to handle large volumes of transactions and maintain data consistency.

**Entities and Relationships:**

**Flights**: This table contains details of the flights, including:

- Flight ID (Primary Key)
- Departure Location
- Arrival Location
- Scheduled Departure Time
- Scheduled Arrival Time
- Aircraft Used

**Passengers**: Stores customer information, including:

- Passenger ID (Primary Key)
- First Name
- Last Name
- Contact Details
- Booking History

**Bookings**: Records each booking made by customers, linking to specific flights and customers.

- Booking ID (Primary Key)
- Passenger ID (Foreign Key referencing Passengers)
- Flight ID (Foreign Key referencing Flights)
- Booking Status (Confirmed, Pending, Canceled)

**Staff**: Manages information related to staff members such as air control department personnel.

- Staff ID (Primary Key)
- First Name
- Last Name
- Role

- Contact Details

**Tickets**: Records details about the tickets purchased.

- Ticket ID (Primary Key)
- Booking ID (Foreign Key referencing Bookings)
- Ticket Class (Economy, Business)
- Ticket Status (Booked, Canceled, Completed)

**Payments**: Tracks payments for bookings.

- Payment ID (Primary Key)
- Booking ID (Foreign Key referencing Bookings)
- Payment Method (Credit Card, PayPal, etc.)
- Amount
- Payment Status (Paid, Pending, Failed)

**Normalization**:

The database schema is normalized to reduce data redundancy and ensure data integrity. The tables are designed following normalization principles:

- **First Normal Form (1NF)**: Each table contains atomic (indivisible) values, ensuring that there are no repeating groups.
- **Second Normal Form (2NF)**: Non-key attributes depend entirely on the primary key, preventing partial dependencies.
- **Third Normal Form (3NF)**: No transitive dependencies exist, ensuring data consistency across tables.

**Indexes**:

To enhance query performance, indexes are created on frequently queried fields such as:

- Flight ID
- Booking ID

- Passenger ID
- Ticket ID
- Payment ID

This helps optimize search operations and joins between tables.

### *5.2 MySQL Implementation*

The MySQL implementation of the Shqipe Airline Management System involves creating tables, establishing relationships, and writing queries to manage and retrieve data efficiently.

**Sample Queries**

1. Retrieve all confirmed bookings with flight details.
2. Retrieve payment status for a specific booking.
3. List available flights between two locations.
4. Count the number of passengers per flight.

## *6. UI/UX Design*

The UI/UX design of the Shqipe Airline Management System is focused on providing users with an intuitive and seamless experience while interacting with the system. This includes flight booking, managing customer details, viewing schedules, and handling payment and ticketing processes. A well-thought-out UI/UX ensures that the users, including passengers and staff, can easily access the information they need and complete tasks efficiently.

### *6.1 Introduction to UI/UX*

User Interface (UI) refers to the visual aspects of the system, including the layout, buttons, colors, and typography. User Experience (UX) focuses on the overall experience a user has when interacting with the system, ensuring that it is intuitive, user-friendly, and enjoyable. In the Shqipe Airline Management System,

the design aims to create a user-friendly interface for customers booking flights and staff managing the airline's operations.

The primary goal is to make it easy for users to book tickets, manage their schedules, and track payment status while ensuring the system is efficient and straightforward to navigate. It is important to balance aesthetics with functionality, ensuring that users can achieve their goals without confusion or frustration.

### *6.2 Design Principles*

To ensure a smooth user experience, several key design principles were followed:

- **Simplicity:** The interface should not be cluttered, and actions should be straightforward. Each page or screen should focus on a single task to avoid overwhelming the user.
- **Consistency:** Consistent use of colors, fonts, and design elements throughout the system helps users understand the interface faster and reduces confusion.
- **Clarity:** All interactive elements (buttons, links, etc.) should be clearly visible, and labels should be intuitive. Information should be organized logically and presented clearly.
- **Feedback:** The system should provide feedback to users when actions are completed (e.g., confirming a flight booking, processing payment).
- **Accessibility:** The design must be accessible to all users.

### *6.3 User Interface Components*

The UI of the Shqipe Airline Management System will include several key components designed to make tasks easier for users:

- **Navigation Bar:** This will allow easy access to various sections of the system, such as booking a flight, viewing upcoming flights, payment processing, and managing customer profiles.
- **Booking Forms:** These will allow users to enter flight details (e.g., departure and destination locations, travel dates) and personal information to book a ticket.
- **Search Filters:** Users can filter flights by various criteria such as date, flight class, and destination to find flights that suit their needs.

- **Booking History:** A component to allow customers to view past bookings and manage upcoming travel.
- **Payment Gateway:** A seamless, secure way for customers to make payments for their bookings, supporting multiple payment methods (credit card, PayPal, etc.).
- **Dashboard:** A dashboard that will help the user to navigate to the desired functionality of the system.

### *6.4 User Experience Considerations*

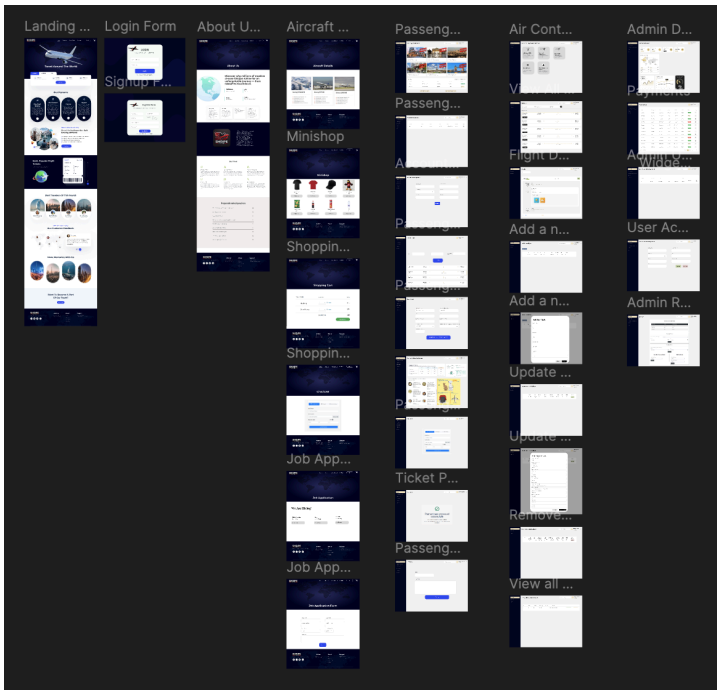
To enhance the UX of the Ship Airline Management System, the following considerations are made:

- **Ease of Navigation:** Simple and intuitive navigation ensures users can easily find what they need.
- **Minimized Input:** Users should be able to book flights with as few steps as possible.
- **Error Prevention and Recovery:** Clear messages will be displayed for user errors (e.g., incorrect flight details, payment failures) with guidance on how to correct the issue.
- **Personalization:** For registered users, the system should offer a personalized dashboard with upcoming flights, historical bookings, and relevant promotions.

### *6.5 Wireframes and Prototypes*

Wireframes and prototypes are essential for visualizing the UI design before development begins. These are low-fidelity representations that focus on layout and functionality rather than visual design.

- **Wireframes:** Basic wireframes will be created for each screen in the system. They will highlight the placement of key UI components such as navigation elements.
- **Prototypes:** High-fidelity prototypes will be developed using tools like Figma to simulate the actual user flow.




### *6.6 Future UI/UX Enhancements*

The design of the Ship Airline Management System will evolve over time to meet the changing needs of users and technology. Potential future enhancements include:

- **Chatbots for Customer Support**: Integrating AI-driven chatbots for answering common queries and helping users navigate the booking process.

## *7. Frontend*

The frontend of the Shqipe Airline Management System is designed to ensure a smooth and intuitive user experience for all users, including customers booking flights, staff managing operations, and administrators overseeing the system. The primary goal of the frontend is to provide users with a visually appealing and highly responsive interface.

The frontend will be built using modern web technologies, ensuring that the system is not only user-friendly but also optimized for performance and scalability. This will ensure that users have a consistent experience when interacting with the system.

## *7.1 Overview of Frontend*

The frontend of the Ship Airline Management System provides a user interface through which customers and staff can access various features and functionalities. It is the part of the system that interacts directly with the user, enabling them to interact with the backend services, such as flight search, booking, payment processing, and flight tracking. Key features of the frontend include:

- **User-Friendly Booking Process:** The system allows users to easily search for flights, choose from available options, and complete their bookings without confusion. Simple forms and minimal input fields help reduce friction in the booking process.
- **Flight and Booking Management:** Users can view flight details, check their booking history, and manage their reservations, including making changes to flight details and checking status updates.
- **Interactive User Dashboard:** The system features a personalized dashboard that provides easy access to upcoming trips, payment statuses, and user information. This helps users keep track of their activities without feeling overwhelmed.
- **Seamless Payment Integration:** The frontend integrates with various payment providers, ensuring users can securely make payments for bookings through a smooth and easy-to-navigate interface.
- **Error Handling and Feedback:** The system provides clear, actionable error messages when users input incorrect information or encounter issues during the booking process. Feedback is provided for each action the user performs, such as booking confirmation or payment success.

## *7.2 Tech Stack*

The frontend of the Shqipe Airline Management System is built using a modern, scalable, and efficient tech stack. This stack ensures the application is both visually appealing and highly functional. Here are the key technologies used:

**CSS**:

- **CSS** is used to style the UI, making it visually appealing and ensuring that the layout is responsive and optimized.

**JavaScript (JS) - Visual Studio Code**:

- **JavaScript** powers the dynamic aspects of the frontend, such as form validation, event handling, and fetching data from the backend. It also enables features like smooth transitions, interactive flight search results, and real-time updates without reloading the page.

**React**:

- **React** is used for building the user interface of the Shqipe Airline Management System. It allows the creation of reusable UI components which makes the development process faster and more maintainable.

**Figma**:

- **Figma** is used for designing wireframes, prototypes, and UI mockups. These tools allow the design team to visualize the layout and user flow before development begins. They also serve as a reference during the frontend development process.

**Jira**:

- **Jira** ensures that each task is assigned, tracked, and completed within the sprint cycle, providing visibility for the design, development, and database teams.

# *8. Backend*

## *8.1 Overview of Backend*

The backend of the Shqipe Airline Management System is the core of the application, responsible for managing data, user authentication, processing requests, and ensuring that all operations in the system run smoothly. It is built using Python, a highly versatile programming language known for its readability, scalability, and rich set of libraries that are ideal for backend development.

In the backend, Python serves as the server-side language, while other tools and technologies handle various tasks, such as API development, database interactions, and managing business logic. The backend interacts

with the frontend through a set of RESTful APIs, enabling seamless communication between the user interface and the server. For secure access, the backend uses Flask-JWT-Extended, which enables token-based authentication and provides role-based access control, allowing different user levels to interact with the system according to their permissions.

Developed using the Flask web framework, the application integrates SQLAlchemy for database management, ensuring smooth interaction with the underlying data.

The backend also supports key operations through structured API endpoints. These include functions for user registration, login, managing flight details, and logging activities performed by air control staff. The system ensures that all sensitive data is handled securely, with tokens validated and expired or revoked tokens properly managed. Additionally, the use of environment variables allows sensitive configurations, like database credentials and secret keys, to be securely stored and accessed.

Furthermore, the application includes integrated Swagger UI documentation, providing an interactive interface for testing and understanding the available API endpoints. Cross-origin resource sharing (CORS) is enabled to facilitate communication between the frontend and backend even when they are hosted on different domains.

Key responsibilities of the backend include:

- **Handling User Requests:** The backend receives requests from the frontend, processes them, and sends back appropriate responses. For example, when a user searches for flights, the backend queries the database, retrieves the relevant flight data, and returns it to the frontend.
- **Database Management:** The backend interacts with the MySQL database to store, retrieve, and update information about users, flights, bookings, payments, and more. MySQL provides a reliable, structured storage solution for the system.
- **User Authentication & Authorization:** The backend ensures that only authorized users can access certain features, such as booking flights or viewing sensitive account details. It manages login, registration and ensures a secure experience for users.
- **Error Handling:** The backend includes mechanisms to gracefully handle errors, ensuring that users receive meaningful error messages when something goes wrong, such as an invalid search query or a failed payment.

- **Performance and Scalability:** The backend is designed to handle a large number of users, concurrent requests, and data, ensuring that the system remains fast and responsive under heavy loads.

By using Python for the backend, the Shqipe Airline Management System ensures that the system is flexible, maintainable, and capable of handling complex tasks in a reliable manner.

### *8.2 Tech Stack*

**MySQL**:

- The database for the Shqipe Airline Management System is structured using **MySQL**, a relational database management system. It stores essential data like user profiles, flight schedules, bookings, and payment records.

**Python - PyCharm**:

- **Python** is used for building the backend of the Shqipe Airline Management System, including API endpoints, business logic, and data processing. As the frontend team creates wireframes and prototypes, the backend team develops the required functionalities in Python to ensure that user actions in the frontend are properly handled.

**Github - Github Link:** https://github.com/ShqipeAirline/ShqipeAirline

- Throughout the development of this project, GitHub was used as the primary platform for version control, collaborative development, and documentation management. All source code, test files, diagrams, reports, and supporting documents were consistently uploaded and maintained in a structured GitHub repository. This ensured that each team member could contribute effectively, track changes in real-time, and maintain a history of every modification made to the project. Branching and pull requests were used for collaborative coding and review, helping maintain code quality and organization across different modules. GitHub also served as the central hub for issue tracking and project progress visibility.

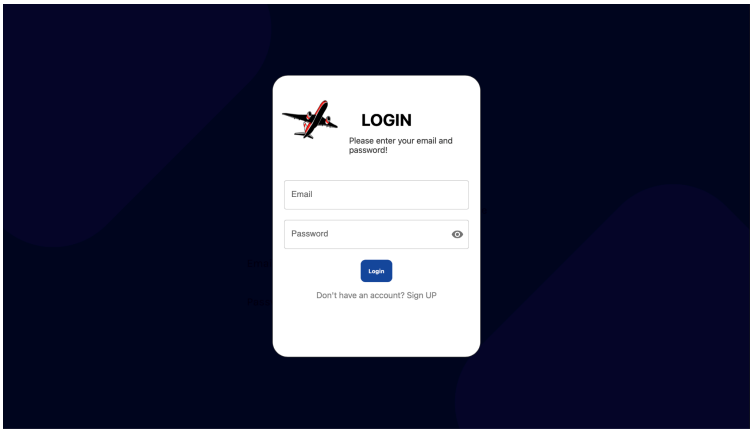
# 9. User Level Interfaces

## 9.1 LogIn & Registration Form

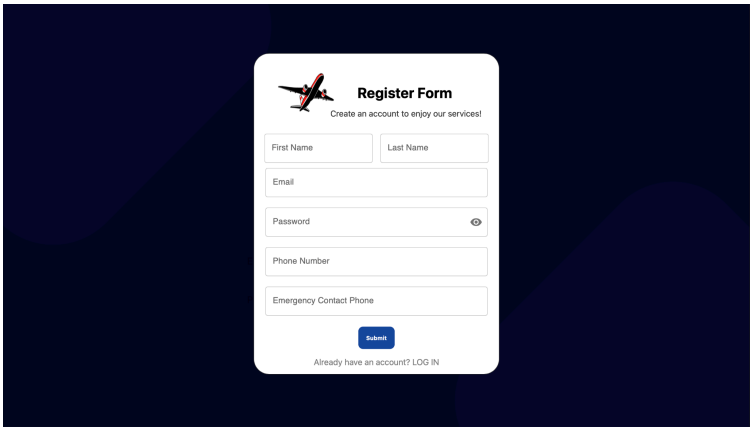
The LogIn and Registration Form is a fundamental feature, designed to provide secure access to the system for all users. The log-in form allows users to authenticate themselves by entering their email and password. It ensures that only authorized individuals can access the system, with role-based access controlling the features and data available to each user.

The registration form allows for the creation of new accounts, capturing essential information such as phone numbers, emails, and passwords. Both forms are built to be intuitive and user-friendly, guiding users through the authentication process with minimal effort.

**LogIn**




**Registration Form**



## *9.2 Landing Pages*

The Landing Page of the Shqipe Airline System serves as the user's first interaction with the platform, aiming to create a positive and lasting impression. It is designed to be visually appealing, intuitive to navigate, and feature-rich. The landing page provides quick access to essential features and information, ensuring a seamless and informative experience for users.

### *9.2.1 Database Tables and Queries*

The Landing Page relies on several core database tables to deliver dynamic and relevant content to users. For interactive elements like job applications, the **Jobs** and **Applications** tables track available positions and candidate submissions respectively, facilitating a smooth recruitment process. Additionally, the **Shop_Items** and **Orders** tables underpin the mini shop functionality, managing product listings and user purchases seamlessly.

Queries are optimized to retrieve real-time information such as featured flights, latest job openings, and current promotions for display on the landing page. Pagination and filtering are implemented in SQL queries to ensure quick response times, especially when loading large datasets like product catalogs or flight schedules.

Together, these tables and queries form the backbone of the landing page’s dynamic content, ensuring that users receive accurate, up-to-date, and personalized information immediately upon arrival.

### *9.2.2 Frontend Design and Features*

The frontend design of the Landing Pages in the Shqipe Airline System emphasizes visual appeal, intuitive navigation, and functionality to ensure a smooth user experience for all visitors. Developed using React, the landing page is structured with modular components that support quick access to various system features, such as flight booking, job applications, and browsing the mini shop.

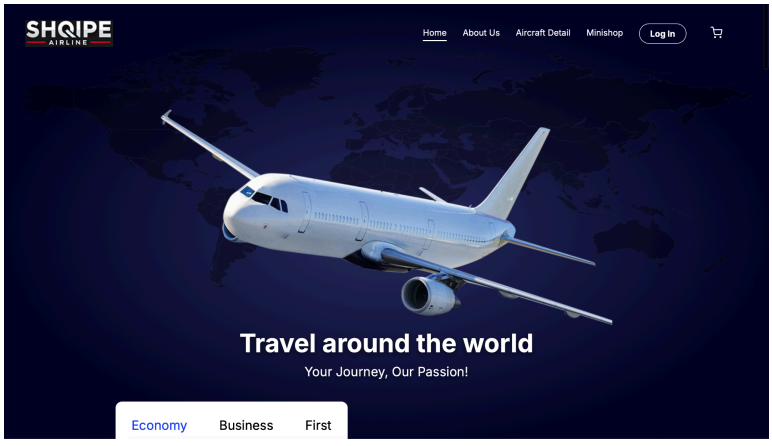
The layout is guided by a UI/UX prototype built in Figma, which served as the foundation for the design. This prototype was created with user-centric principles in mind, ensuring a clean interface with logical navigation and a clear flow of interaction.

Styling is managed using CSS and responsive design techniques, allowing the interface to seamlessly adapt to different screen sizes, from desktop monitors to mobile devices. The visual identity follows a minimalist design approach, using a well-balanced color palette and accessible typography to enhance readability and reduce distraction.

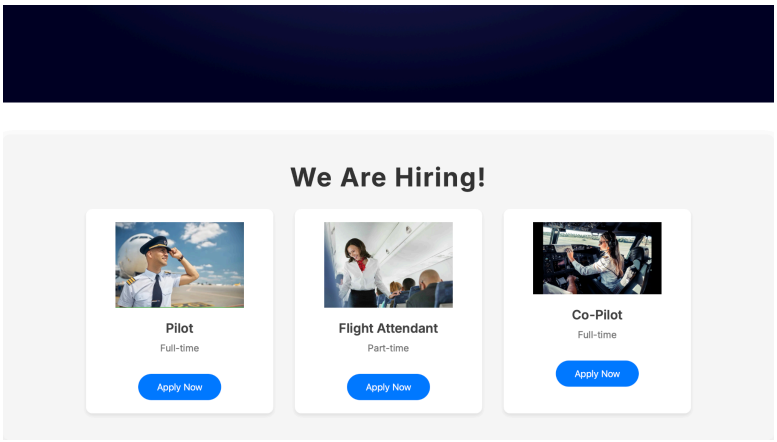
Vite is utilized for fast build times and efficient performance, ensuring that the landing page remains responsive and scalable as new features are introduced. The goal is to keep users engaged with quick load times and smooth transitions between sections.

The interface prioritizes ease of navigation and accessibility, allowing both new visitors and returning users to find key services effortlessly. Each section whether it's logging in, exploring aircraft details, or reaching out through the contact form is clearly marked and easy to access. The footer provides always-available links to contact details, physical address, and additional support, ensuring that vital information is never out of reach.

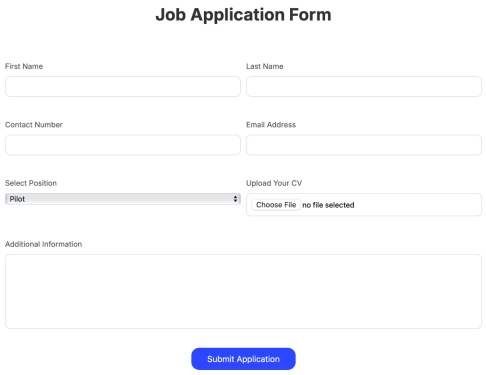
**Home**



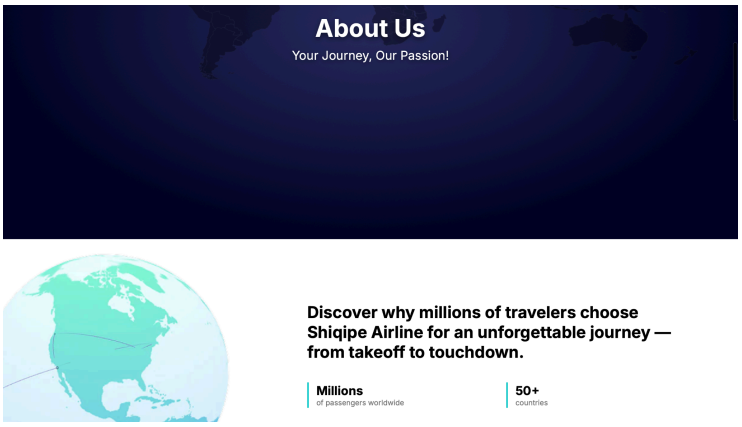

**Job Application**



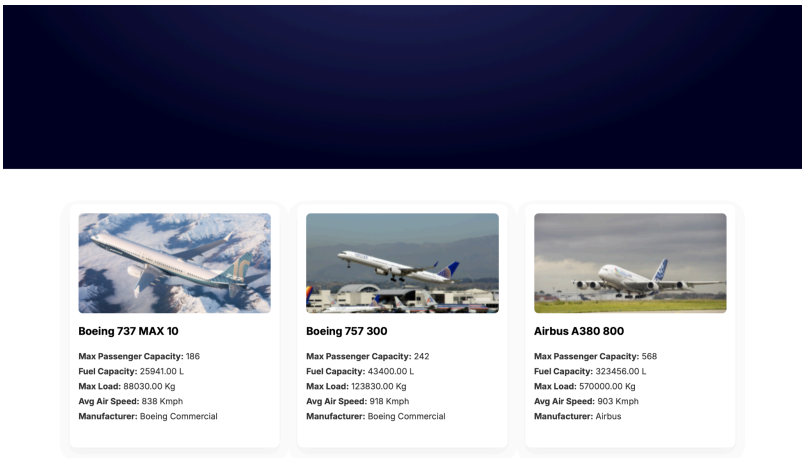

## Job Application Form



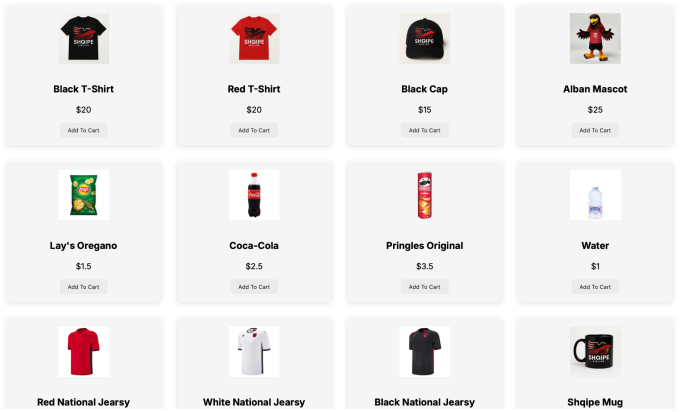

## About Us



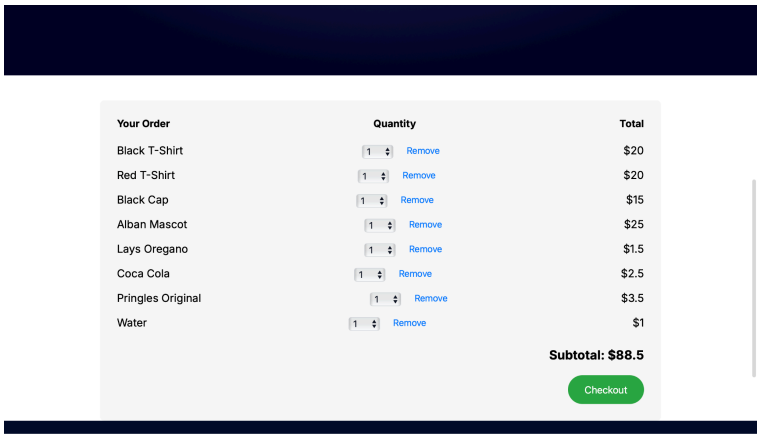
## Aircraft Detail



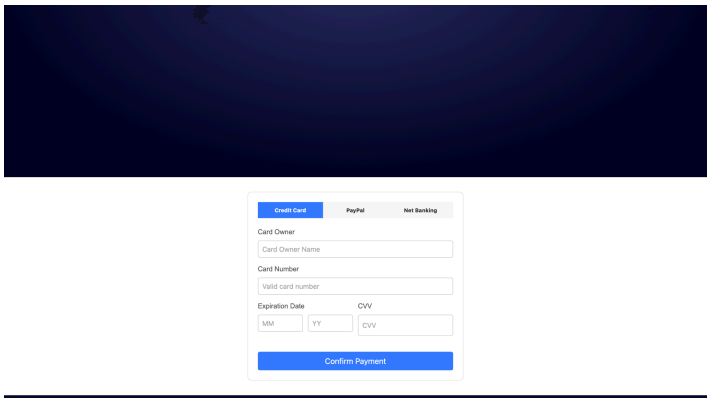

## Minishop




## Shopping Cart



**Payment**




### *9.2.3 Backend Logic and API Endpoints*

The landing page prominently features a section highlighting current job openings at Shqipe Airline, with backend APIs providing up-to-date job listings stored in dedicated database tables and validated through structured schemas. These APIs deliver essential details such as job titles, descriptions, requirements, and application deadlines, enabling dynamic content updates without the need to redeploy the frontend. Additionally, users exploring the airline's mini-shop benefit from backend endpoints that supply comprehensive product information, including prices and availability, managed through schemas similar to those used for bookings and payments. The shopping cart functionality is supported by APIs that allow users to add or remove items and calculate totals, ensuring a smooth and seamless checkout process. Furthermore, contact and support information is efficiently handled via backend services that provide static and real-time data—such as addresses, phone numbers, emails, and live chat availability—using simple schemas to populate the landing page's footer and support sections, guaranteeing users can easily access help whenever needed.

## *9.3 Air Control Department Interface*

The Air Control Department Interface serves as a dedicated platform tailored to the needs of personnel responsible for overseeing aviation operations. It provides a centralized space where essential information related to flight activities, schedules, and control operations can be accessed and managed efficiently. Designed with clarity and usability in mind, the interface supports the department's role in maintaining organized, timely, and safe airspace coordination. It reflects the critical nature of the department's

responsibilities while ensuring a user-friendly experience aligned with operational standards.

### *9.3.1 Database Tables and Queries*

The database structure supporting the Air Control Department Interface is designed to ensure efficient management of flight operations and staff activities. It includes three core tables:

- The *Flights* table stores comprehensive information about each scheduled flight, including flight number, airline, departure and arrival airports, seating capacity, availability, and status (e.g., on-time, delayed). Each flight is associated with the staff member who created it, identified through the created_by field.
- The *Air_Control_Dep* table holds records of all personnel in the air control department. This includes usernames, emails, encrypted passwords, names, contact details, and department names. It also logs timestamps for account creation and last login, enabling better user tracking and accountability.
- The *Air_Control_Activity_Log* table functions as an audit trail, recording every significant action performed by air control staff, such as monitoring or updating flight records. Each log entry links back to both the staff member and the specific flight, preserving data integrity through foreign key constraints.

Sample SQL queries have been implemented to insert staff records, flight data, and log activity, illustrating how these tables interact. This structure provides a solid foundation for role-based access, operational transparency, and future scalability.

### *9.3.2 Frontend Design and Features*

The frontend design of the Air Control Department Interface focuses on simplicity, clarity, and usability to ensure an efficient user experience for air control personnel. Built with React, the interface is organized into clear, manageable components that allow for a seamless flow between different sections of the application. The use of CSS ensures that the visual layout is clean and professional, with a responsive design that adapts to various screen sizes, providing consistency across desktop and mobile devices.

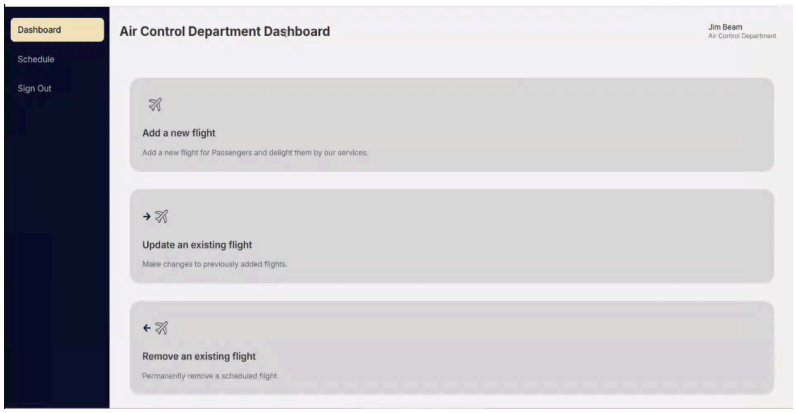
The overall structure of the interface is based on a UI/UX Figma model, which served as the blueprint for the design. This model was carefully crafted to ensure an intuitive user experience, with organized

components and logical flow that enhances usability.

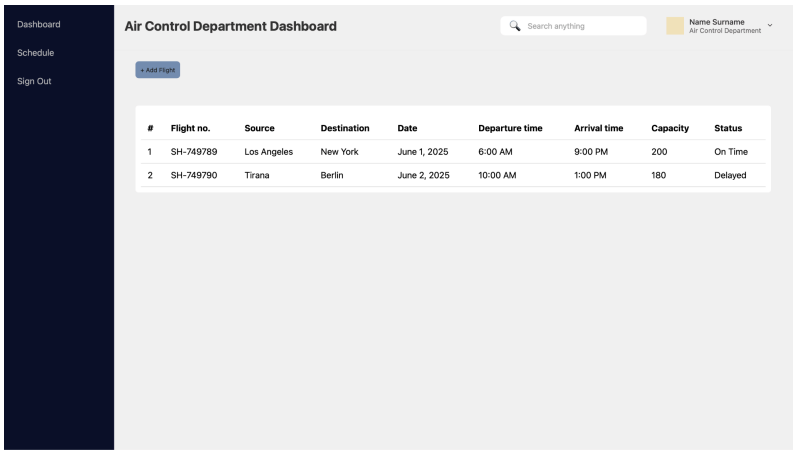
The overall aesthetic emphasizes a minimalistic approach, using a color palette and typography that supports clarity and reduces visual clutter. By leveraging Vite for fast, optimized builds, the development process ensures rapid updates and smooth performance, even as the system scales.

The interface's design also incorporates user-centric principles, focusing on accessibility and ease of navigation. The design principles emphasize accessibility, ensuring that all users can easily interact with the interface, while the layout offers quick access to essential data and functions. Each section is carefully organized to minimize unnecessary steps, enhancing the overall workflow for air control staff.

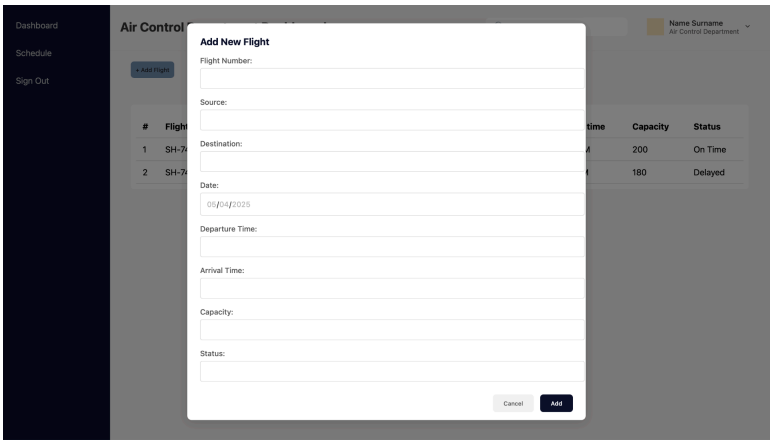
**Air Control Department Dashboard**



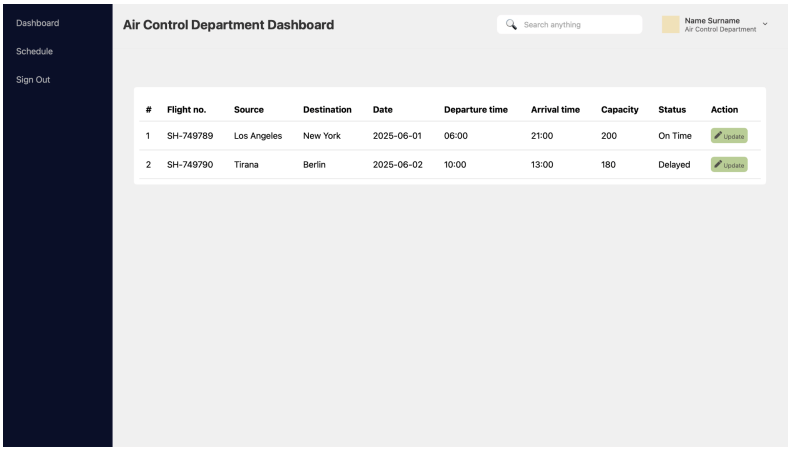

**Add a New Flight**



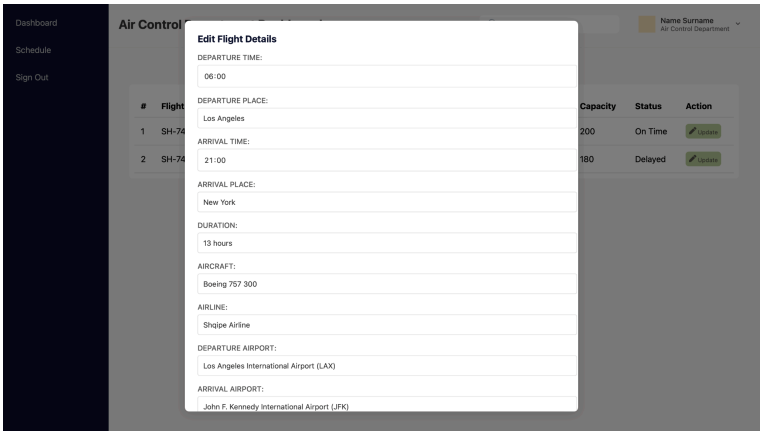
## Add a New Flight Form



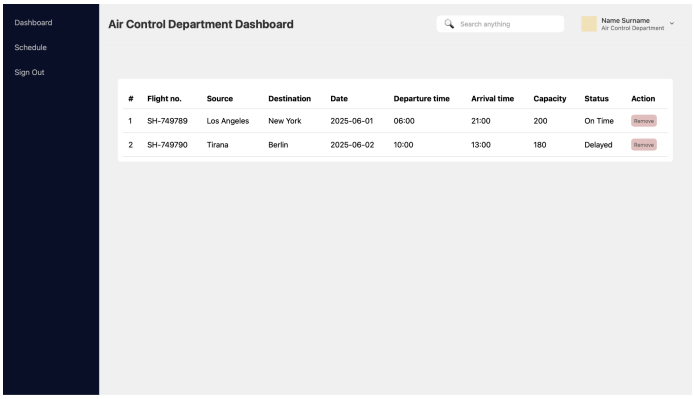

## Update an Existing Flight



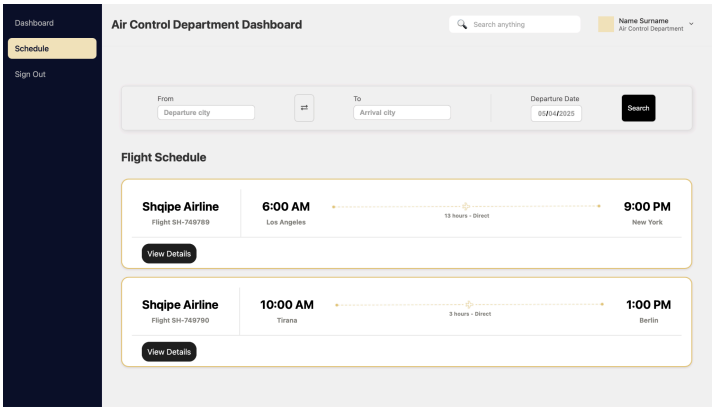

## Update an Existing Flight Form



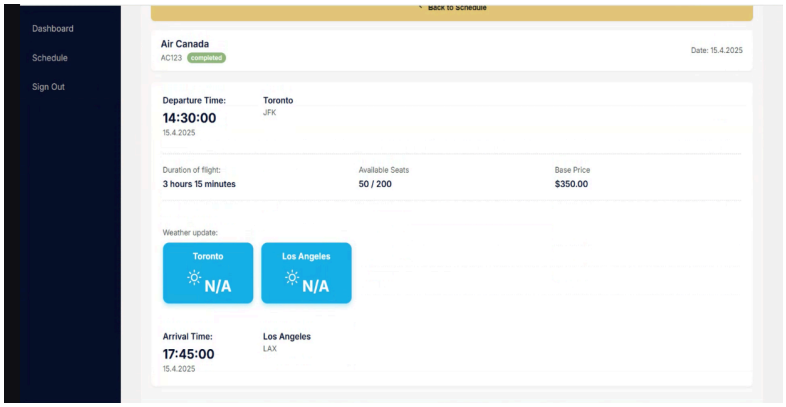
## Remove an Existing Flight



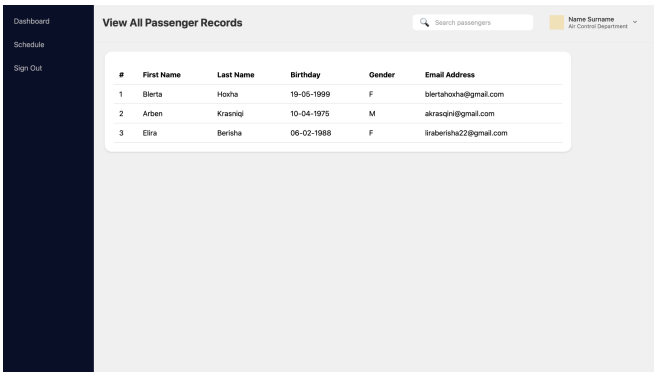

## View All Flights (Schedule)




## Flight Details



**View All Passenger Records**




### *9.3.3 Backend Logic and API Endpoints*

The backend logic of the Air Control Department Interface is built to provide a secure and efficient framework for managing core operations, including flight management, etc.

The backend logic of the Air Control Department Interface is specifically designed to manage operations related to flight scheduling, user authentication, and activity tracking for air control staff. Built with Flask, the application utilizes SQLAlchemy for database management, ensuring efficient handling of flight and staff data. Flask-JWT-Extended is implemented for token-based authentication, allowing only authorized air control personnel to access the system and perform specific actions based on their roles.

API endpoints are created to handle tasks such as flight updates, performed by air control staff. Each action is recorded in an activity log for transparency and accountability. Additionally, the backend ensures the secure handling of sensitive data with token validation, expiration handling, and environment variables for critical configurations.

With these features, the backend is designed to offer a reliable, secure, and scalable solution for the Air Control Department, ensuring smooth operations and a seamless user experience.

## *9.4 Admin Interface*

The Admin Interface provides a centralized control panel for managing all core operations of the Shqipe

Airline System. Designed with clarity and efficiency in mind, it allows administrators to oversee bookings, flights, transactions, users, and reports through an intuitive, real-time dashboard.

### *9.2.1 Database Tables and Queries*

The database behind the Admin Interface is designed to organize and store key information related to the airline's operations. It includes tables that represent important areas such as user accounts, flight schedules, bookings, transactions, and administrative actions. Each table holds specific types of information that help the system track what's happening, whether it's a new flight being added, a booking being made, or a refund being issued.

Queries are used to retrieve and manage this data. For example, admins can view the most recent bookings, check the status of flights, review financial transactions, or look up user information when needed. These queries are essential for keeping the system running smoothly and making sure the right data is shown in the Admin Dashboard.

The structure and relationships between these tables are designed to support both daily operations and broader reporting tasks, giving the admin team the tools they need to monitor activity and make informed decisions.

### *9.2.2 Frontend Design and Features*

The Admin Dashboard is designed to provide a comprehensive and centralized interface for the highest-level user within the Shqipe Airline System. Built using React and guided by a structured Figma UI/UX model, the dashboard focuses on clarity, control, and efficiency, enabling administrators to monitor and manage all critical aspects of the airline system.

The frontend layout is composed of modular, interactive components that present real-time data in an organized and actionable format. Key areas include:

- Booking Overview: Displays current booking statistics and trends.
- Flight Management: Provides access to upcoming, ongoing, and past flights.
- Transactions: Shows financial records including ticket sales and refunds.
- User Account Management: Enables the creation, updating, and deactivation of user accounts

(passengers, air control staff, etc.).

- Reports: Offers access to downloadable system-wide reports and analytics.

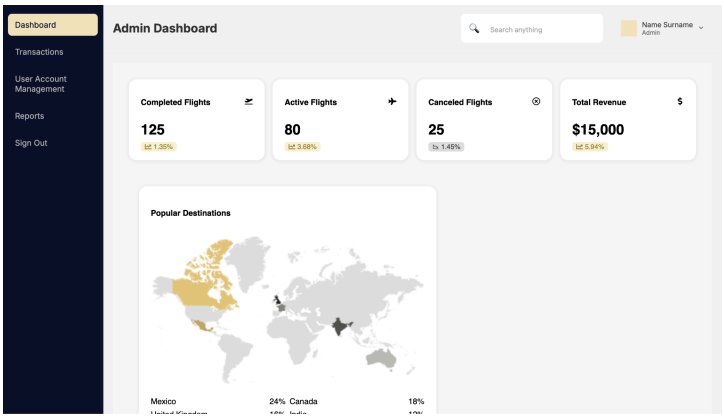
A minimalist design approach is adopted, prioritizing readability and reducing cognitive load. Icons, charts, and dynamic cards help the admin quickly interpret data without navigating through multiple layers.

To support rapid development and smooth performance, Vite is used for fast builds and real-time feedback during updates. The component-based architecture allows for scalability as new admin functionalities are introduced.

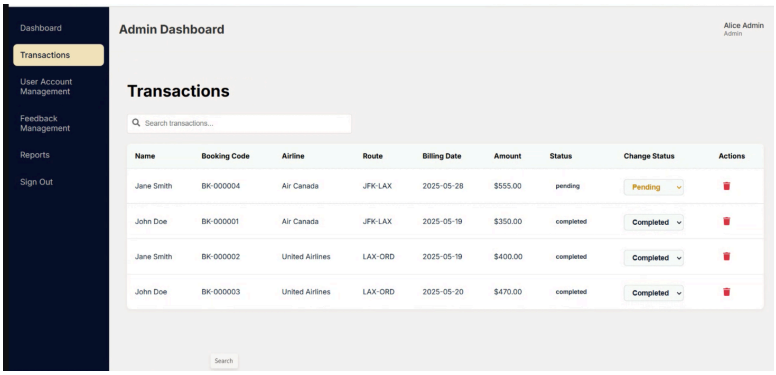
Accessibility remains a core priority, with keyboard navigation, screen reader support, and consistent layouts throughout. The design encourages task efficiency by minimizing unnecessary steps and offering quick access to core functions through a clean sidebar or top navigation menu.

Overall, the Admin Dashboard's frontend design ensures that airline administrators have full control over operational data, user roles, and system health all in one cohesive, user-friendly interface.

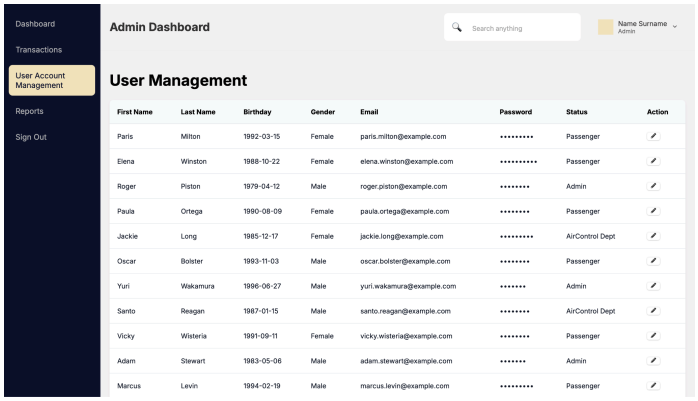
**Admin Dashboard**



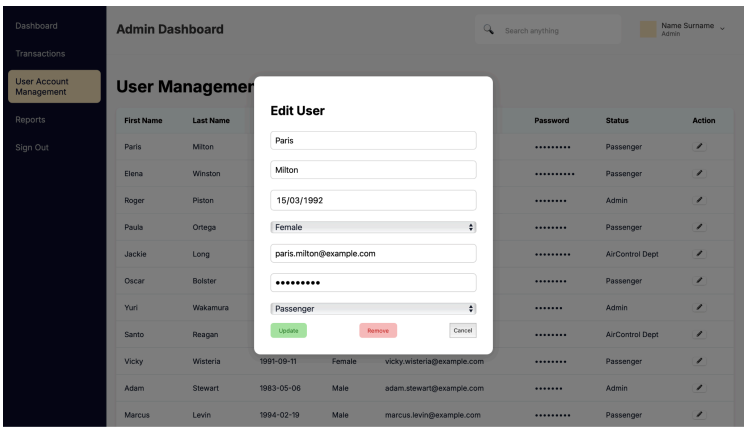
## Transactions



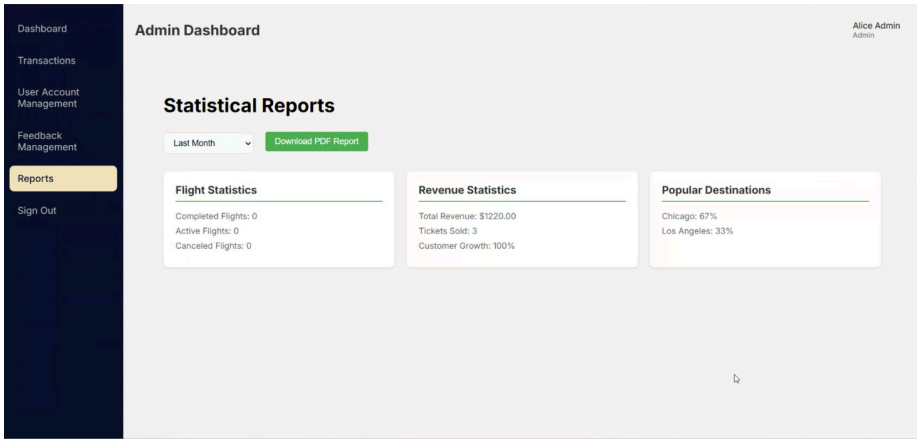

## User Account Management



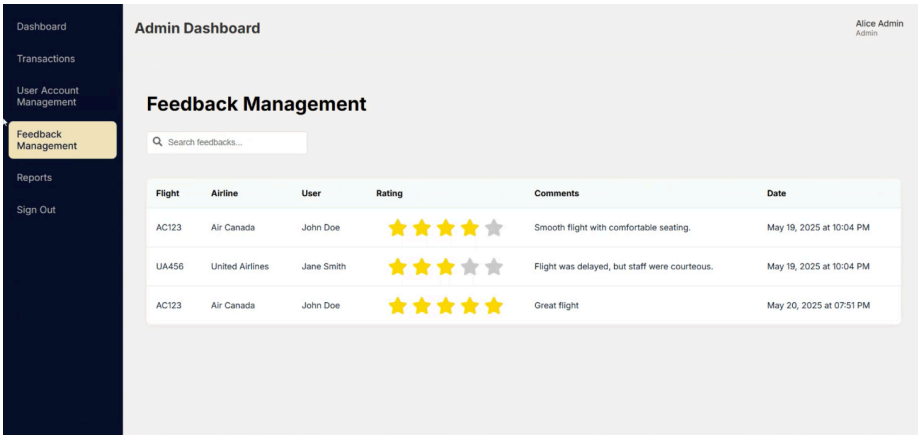

## User Account Management Form



**Reports**




**Feedback Management**




### *9.2.3 Backend Logic and API Endpoints*

The backend of the Admin Dashboard forms the core of the system's operational engine. It is responsible for processing all administrative actions and ensuring secure, efficient communication between the interface and the database. Every action initiated by the administrator—whether viewing flight statistics, managing users, or generating reports—is powered by carefully structured backend logic.

At the heart of this system are API endpoints, which act as bridges between the admin interface and the underlying data. These endpoints handle requests such as retrieving booking records, updating flight

statuses, processing transactions, and managing user roles. Each endpoint is crafted to respond with accurate and timely information, supporting the real-time needs of the dashboard.

Built using a modular and scalable approach, the backend ensures that new functionalities can be added smoothly as the airline system grows. It also incorporates layers of validation and access control to maintain data integrity and security, ensuring that only authorized personnel can perform sensitive operations.

Logging mechanisms are in place to monitor activities and support auditing, while performance optimizations allow for quick response times even under high demand. Overall, the backend architecture empowers the admin interface with the reliability, control, and responsiveness necessary for managing the airline's operations seamlessly.

## *9.5 Passenger Interface*

The Passenger Interface is designed to offer travelers a seamless and intuitive experience while interacting with the Shqipe Airline System. Built with usability and convenience at its core, the interface enables passengers to search for flights, make bookings, manage their reservations, complete payments, and provide feedback.

With a clean and responsive design, passengers can access real-time flight information, choose seats, add extras like baggage or travel insurance, and securely complete transactions. The interface also provides access to booking history, e-tickets, and profile settings, ensuring users have full control over their travel journey from a single, user-friendly platform.

### *9.2.1 Database Tables and Queries*

The passenger interface relies on a structured set of relational database tables to store and manage key information related to users, bookings, flights, payments, and feedback. Core tables include users, bookings, flights, payment_methods, transactions, and feedback, each linked via primary and foreign key relationships to ensure referential integrity.

Efficient queries are designed to retrieve user-specific data such as active and past bookings, seat availability, transaction history, and feedback submissions. These queries enable fast, secure, and accurate access to essential passenger information, forming the foundation for personalized and responsive user

experiences throughout the system.

### *9.2.2 Frontend Design and Features*

The Passenger Dashboard in the Shqipe Airline System is designed with a strong focus on personalization, convenience, and self-service. Built using React and based on a detailed Figma UI/UX design, this dashboard offers an interactive and user-friendly environment where passengers can manage their travel plans with ease.

The interface is composed of modular components, enabling quick access to all essential features. Key elements include:

- Promotions Panel: Highlights current promotional offers, discounts, and personalized travel deals.
- Flight Status Section: Displays a categorized list of flights:
  - Confirmed (paid and booked),
  - Pending (awaiting payment or confirmation),
  - Cancelled or Rescheduled (if applicable).
- Travel Services: Quick links to recommended hotels and vehicle rental options based on travel destination or booking history.
- Account Management: Allows passengers to update their profile, manage personal details, and change passwords securely.
- Flight Booking Tools: Integrated flight search, booking functionality, and in-dashboard payment options.
- Feedback Submission: A simple and accessible form for passengers to leave feedback or rate their travel experience.

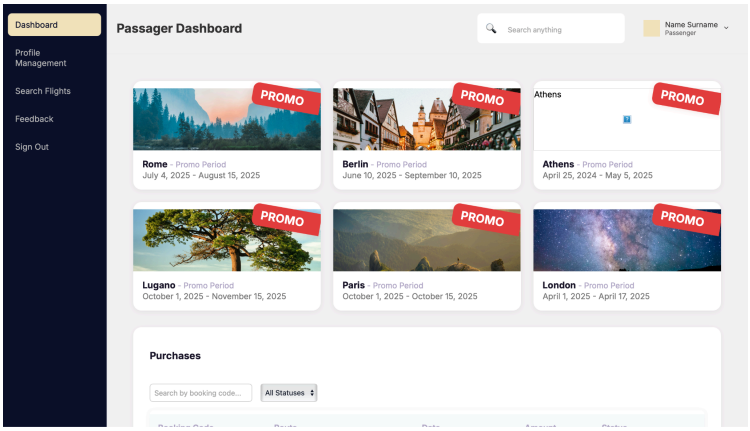
Responsive design and CSS styling ensure that the dashboard works smoothly on both mobile and desktop devices, adapting fluidly to various screen sizes. A modern, minimal UI supports clear navigation, while consistent color schemes and icons promote ease of use and visual appeal.

The use of Vite during development allows for rapid build times and enhanced performance, ensuring the passenger experience remains fast, seamless, and responsive. Real-time updates ensure that flight information, payment statuses, and offers remain current.

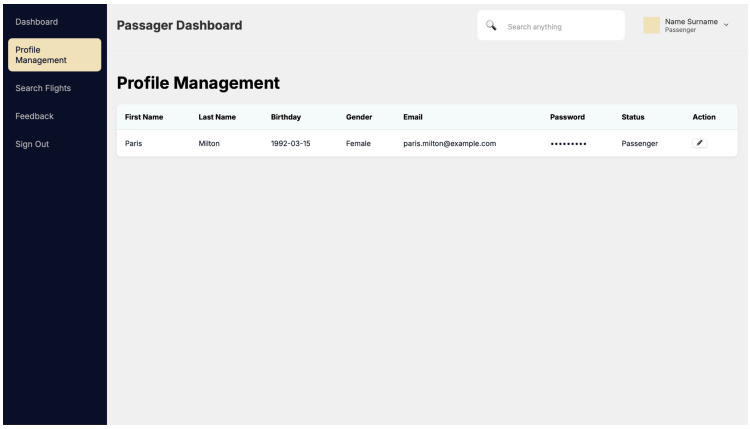
Accessibility features such as labeled form elements, logical tab navigation, and keyboard shortcuts enhance usability for all passengers, including those with assistive needs. The interface emphasizes clarity and independence, giving users full control over their travel activities with minimal friction.

The Passenger Dashboard empowers users with all the tools they need to manage their journey end-to-end searching, booking, paying, accessing related services, and sharing feedback within a single, intuitive interface.

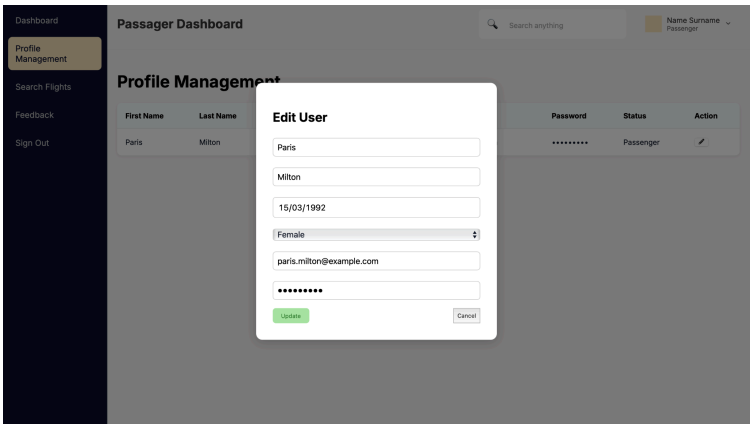
**Passenger Dashboard**



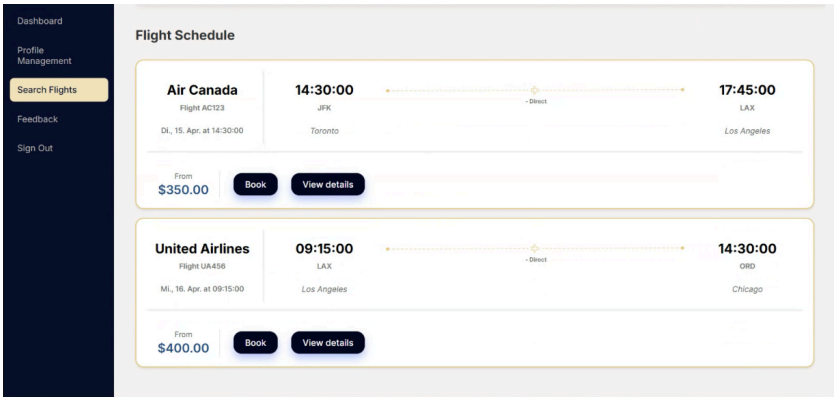

**Profile Management**



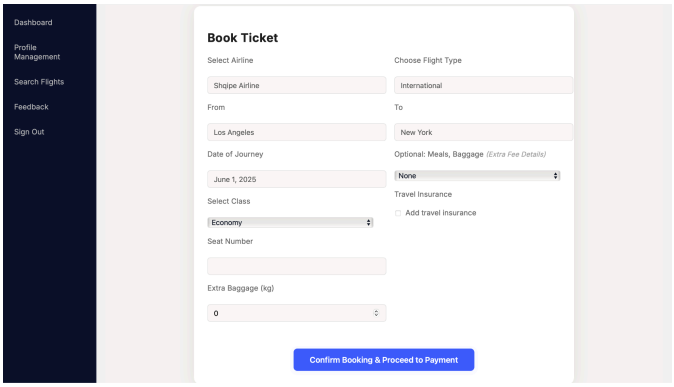

**Edit User Form**



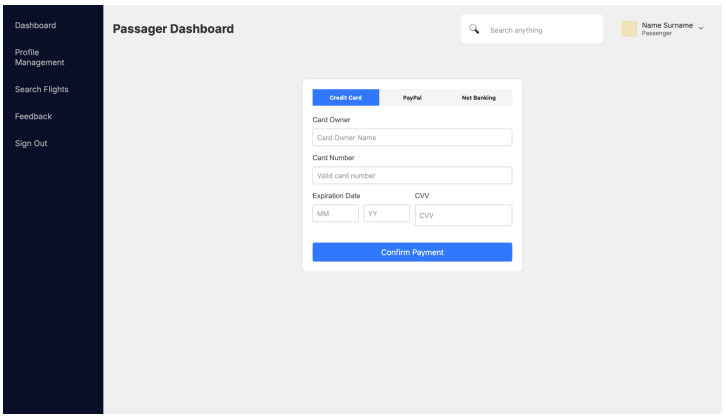

## Search Flight



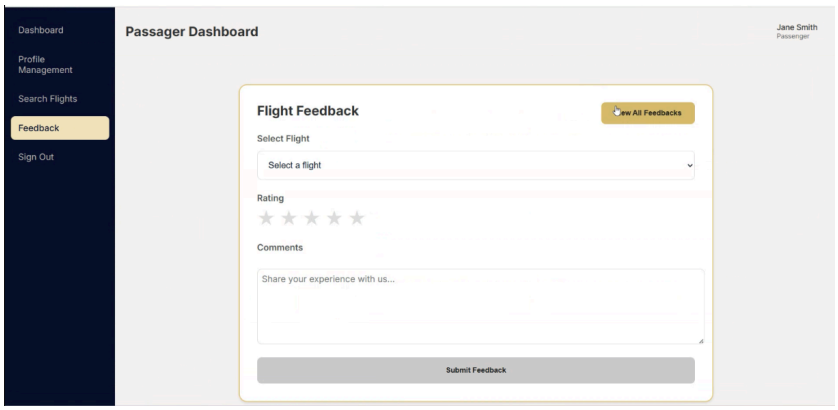

## Book Ticket

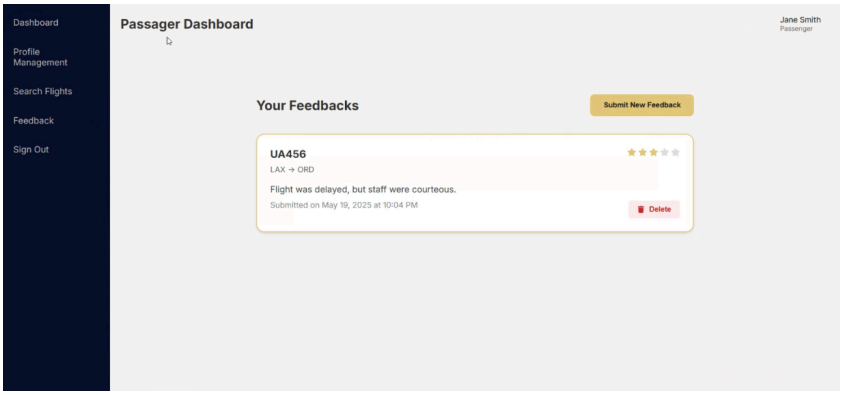


## Payment




## Feedback



### *9.2.3 Backend Logic and API Endpoints*

The backend logic for the Passenger Interface is responsible for processing user requests, handling business logic, and ensuring secure data transactions. Built with a RESTful architecture, the backend exposes API endpoints for functionalities such as user registration and login, flight search, booking creation and management, payment processing, and feedback submission.

Each endpoint interacts with the corresponding database tables and applies validations, role-based access control, and error handling to maintain data integrity and user security. Modular service layers support scalability, while JSON-formatted responses ensure smooth communication between the frontend and backend systems.