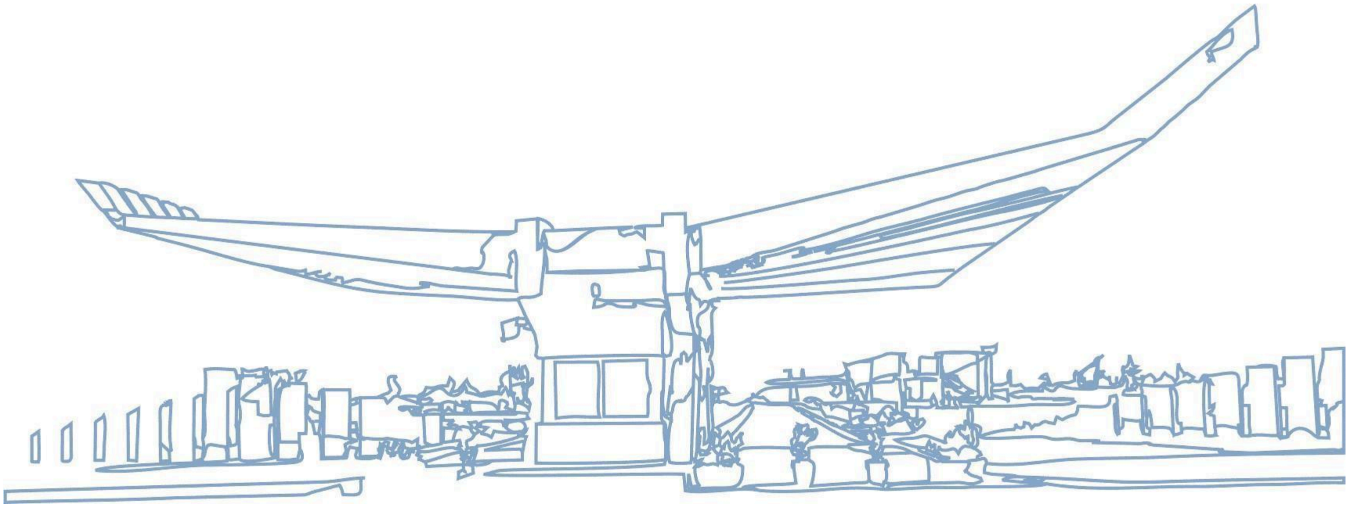
Software Project Management

[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.

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

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

Authentication & Account Management

- **Log in / Sign up:** Allows users to access or register their accounts.
- **Two-factor authentication (2FA) for enhanced security:** Adds an extra layer of protection for account access.

- **Profile management (edit details, change password, update payment methods):** Lets users update personal information and preferences.
- **Reset password via email or SMS :** Provides a secure method for recovering lost passwords.

Flight Booking & Management

- **Search for flights by date, destination, airline, and price:** Allows passengers to find flights based on their preferences and budget.
- **Book flights :** Enables passengers to select seats, customize meal options, and add extra baggage for an additional fee.
- **View ticket details and status (confirmed, pending, canceled):** Provides access to booking details and current ticket status.
- **Cancel or reschedule flights:** Allows passengers to modify or cancel bookings while viewing.
- **Receive real-time flight status notifications (delays, gate changes, etc.):** Notifies regarding flight changes, delays, and gate updates.
- **View baggage policies and restrictions:** Displays airline-specific baggage rules and limitations.

Payments & Transactions

- **Secure payment processing (possibly credit/debit cards, PayPal):** This feature ensures that users can securely make payments using credit/debit cards or PayPal, with encrypted transactions for safety.
- **Save payment methods for future use:** Users can securely save their payment methods for future transactions, streamlining checkout processes for convenience.
- **View and download invoices, receipts, and e-tickets:** Users can access and download invoices, e-tickets after being sent in their email, making it easy to keep track of their purchases.
- **Apply discount codes**

Flight & Travel History

- **View past flight history and upcoming flights:** Users can easily access their past flight details and upcoming flight schedules for better travel planning.
- **Download boarding passes:** Passengers can download and store their boarding passes directly from their email for quick access during travel.

Customer Service & Feedback

- **Submit feedback and rate flight experience:** Passengers can submit feedback and rate their flight experience to help airlines improve their services.
- **Request special assistance (wheelchair, medical, baby support):** Travelers can easily request special assistance, such as wheelchair support, medical aid, or assistance for babies, to ensure a smooth journey.
- **Extra features (meals, extra baggage, pets):** Travelers can request additional services such as meals, extra baggage, or accommodations for pets, ensuring a more comfortable journey.
- **Whatsapp redirect:** Users can be redirected to customer service via WhatsApp for quick and convenient support during their travel experience.

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.

Mini shop

- **In-flight shopping (pre-order, duty-free items):** Passengers can pre-order duty-free items for in-flight delivery, enhancing their shopping experience during the flight.
- **Browse and purchase items like snacks, souvenirs:** Travelers can browse and purchase a variety of in-flight items, such as snacks, and souvenirs, for added convenience.
- **Digital receipts and transaction tracking:** Users receive digital receipts.

Job application

- **Upload resume and necessary documents:** Candidates can apply for jobs by uploading their resume and necessary documents directly through the platform.

Admin - The admin oversees the entire airline management system.

User & Account Management

- **Manage user roles and permissions:** The admin can assign and modify user roles and permissions, controlling access to different parts of the system.

System Monitoring & Reporting

- **Access detailed reports:**
    - **Flight reports (delays, cancellations, trends)**
    - **Financial reports (revenue, losses, expenses)**

Job application

- **Publishes job application form and accesses applications:** The admin publishes job application forms and accesses submitted applications to manage recruitment processes.

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

Flight Management

- **Add, modify, and cancel flights:** The department has the authority to schedule new flights, adjust existing ones, or cancel flights when necessary.
- **Assign flight numbers, routes, and schedules:** The department assigns appropriate flight numbers, routes, and flight schedules to ensure optimal operations and efficiency.
- **Update flight capacity and available seats:** The department is responsible for managing the flight capacity and ensuring accurate availability of seats for passengers.

Data & Backup Management

- **Integrate flight plan data with air traffic control systems:** The department ensures that flight plans are seamlessly integrated with air traffic control systems to manage safe and efficient airspace operations.

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.

Mini shop

- **Handles mini shop:** The department handles financial management for the mini shop, including tracking sales, managing inventory, and ensuring profitability.

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.
- Two-factor authentication (2FA) should be mandatory for both passengers and admin users.
- 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.

Reliability & Availability

- Regular database backups (every 24 hours) to prevent data loss.

Maintainability & Support

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

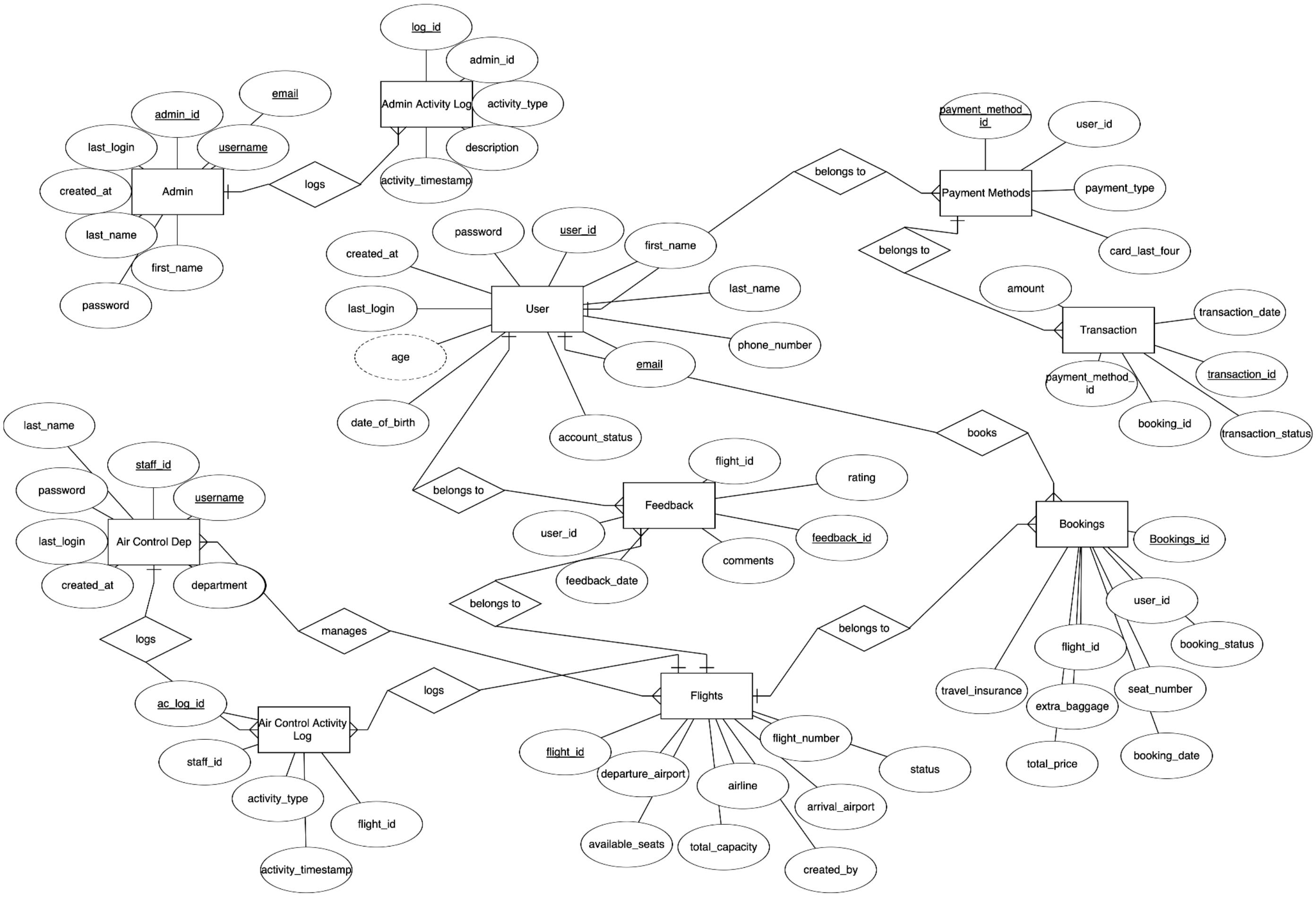
Interoperability

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

4. Diagram/Schemas

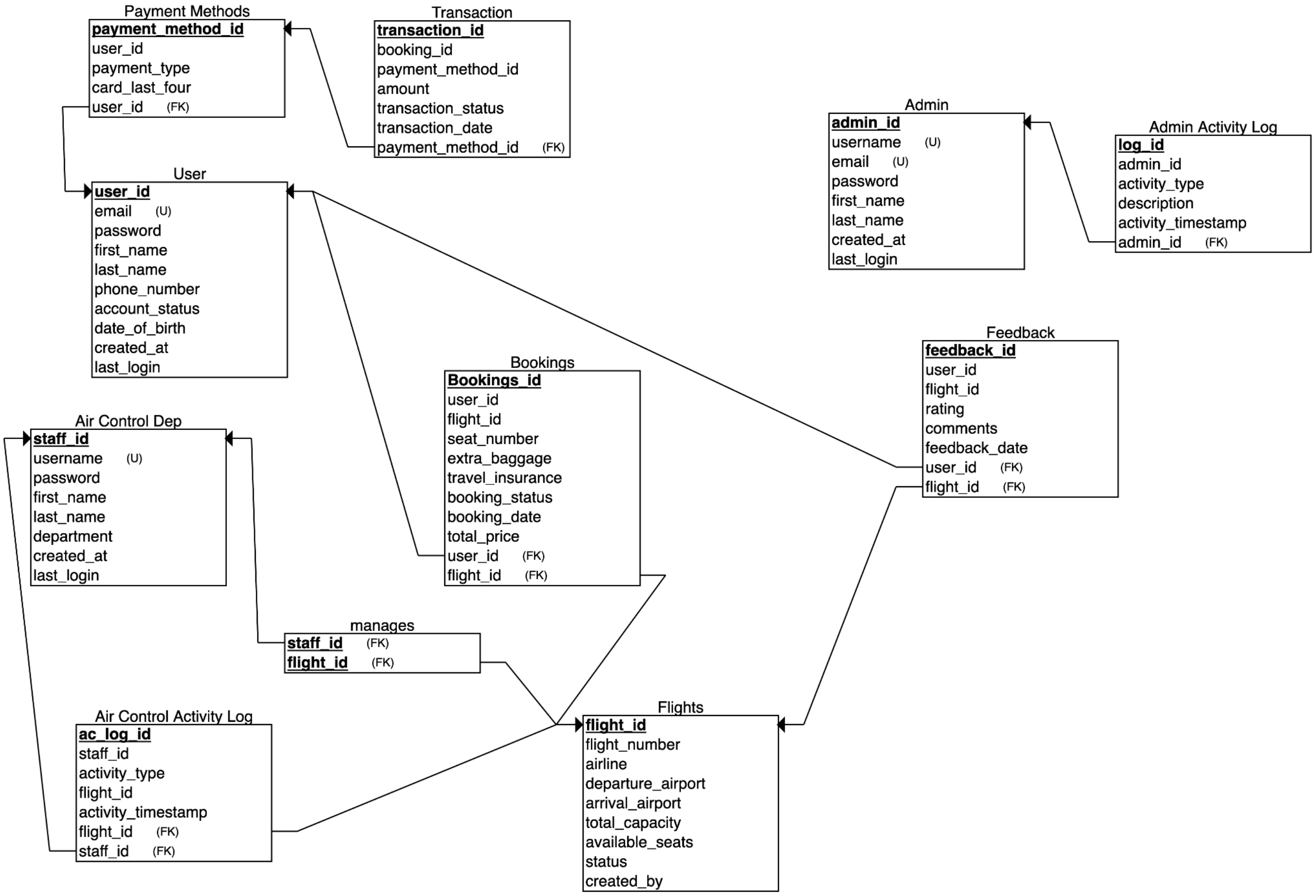
4.1 Entity Relationship Diagram (ERD)

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.



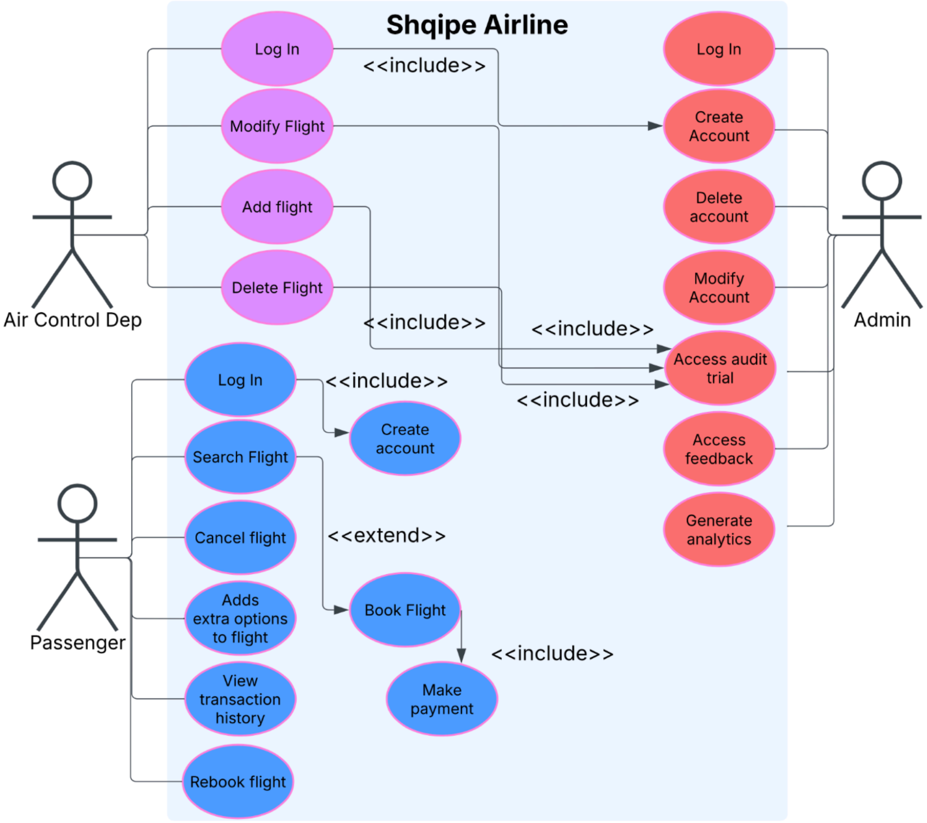
4.2 Relational Schema (RS)

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.



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.

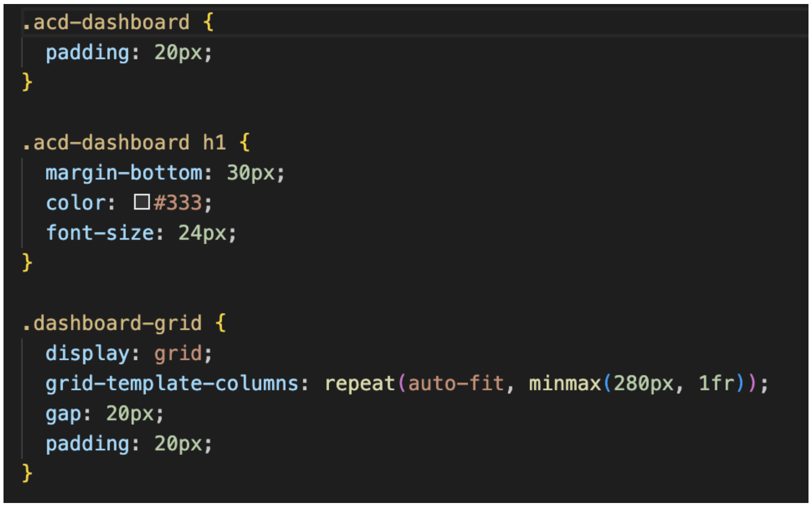
7.2 Tech Stack

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.

Snippet from code:

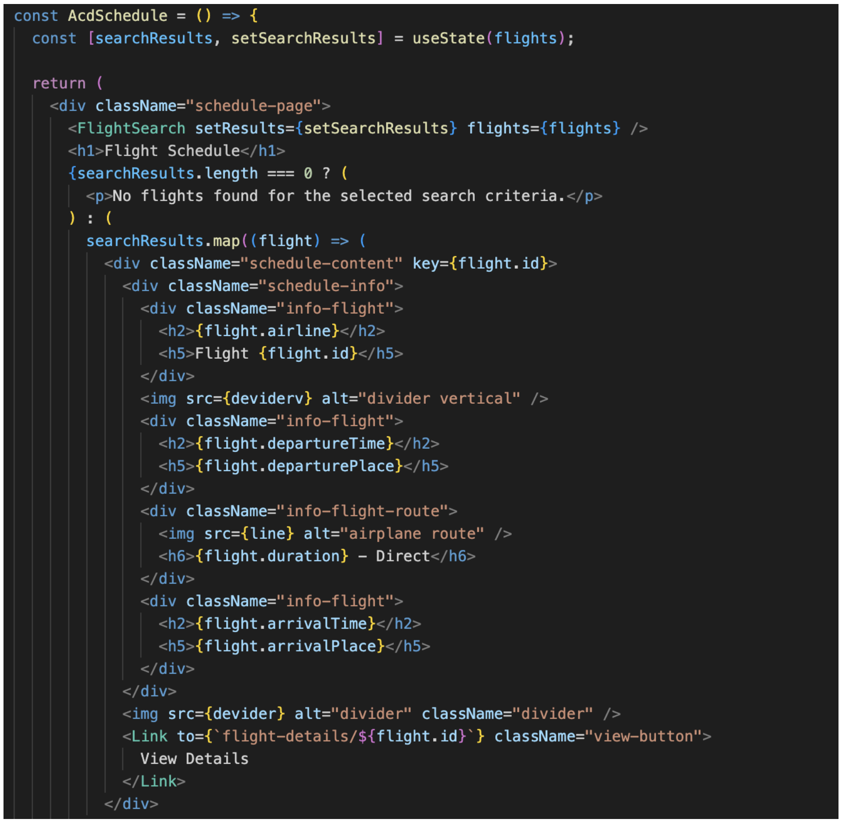


```
.acd-dashboard {
  padding: 20px;
}

.acd-dashboard h1 {
  margin-bottom: 30px;
  color: □#333;
  font-size: 24px;
}

.dashboard-grid {
  display: grid;
  grid-template-columns: repeat(auto-fit, minmax(280px, 1fr));
  gap: 20px;
  padding: 20px;
}
```

JavaScript (JS):

- **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.

Snippet from code:



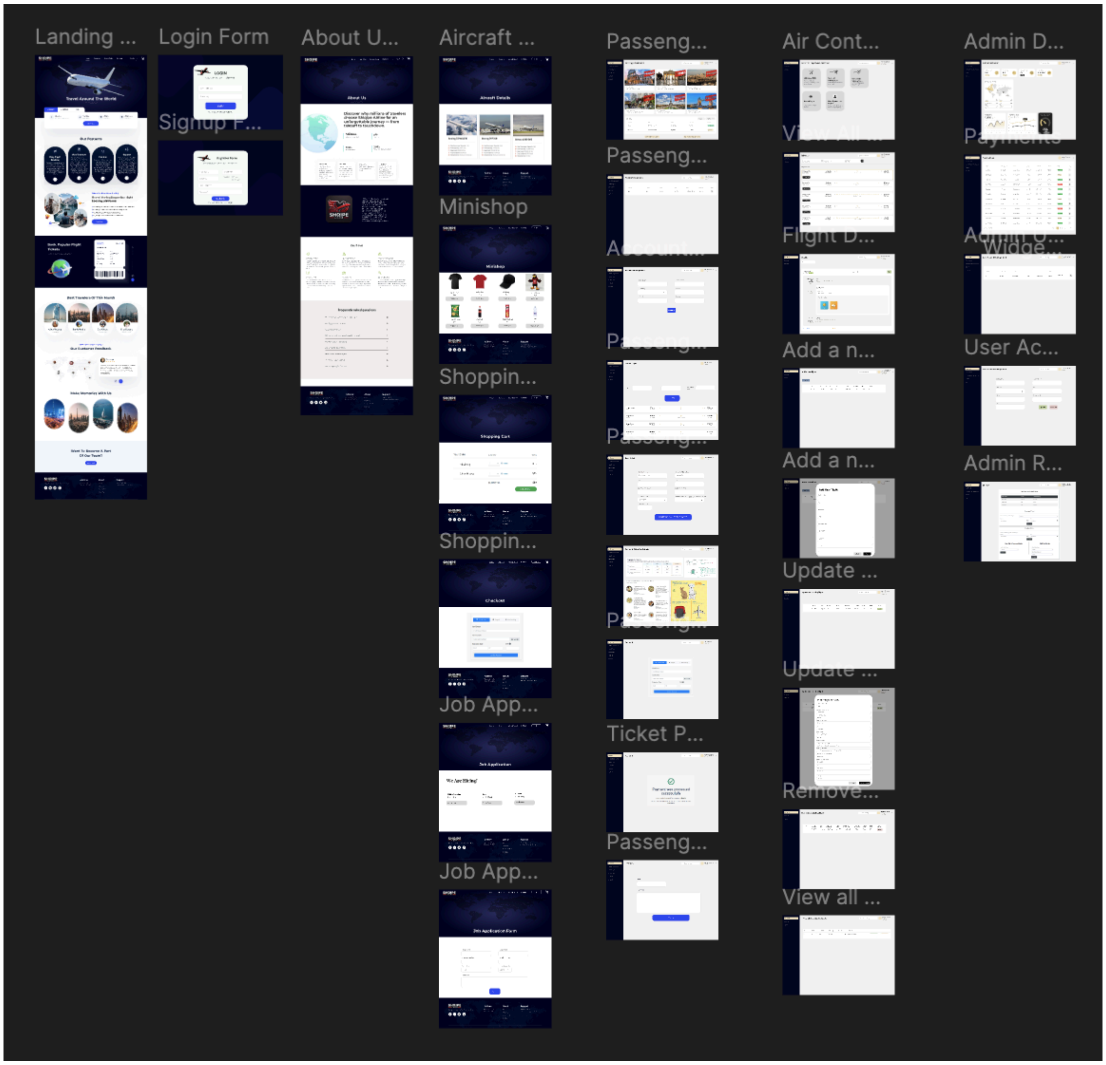
```
const AcdSchedule = () => {
  const [searchResults, setSearchResults] = useState(flights);

  return (
    <div className="schedule-page">
      <FlightSearch setResults={setSearchResults} flights={flights} />
      <h1>Flight Schedule</h1>
      {searchResults.length === 0 ? (
        <p>No flights found for the selected search criteria.</p>
      ) : (
        searchResults.map((flight) => (
          <div className="schedule-content" key={flight.id}>
            <div className="schedule-info">
              <div className="info-flight">
                <h2>{flight.airline}</h2>
                <h5>Flight {flight.id}</h5>
              </div>
              <img src={deviderv} alt="divider vertical" />
              <div className="info-flight">
                <h2>{flight.departureTime}</h2>
                <h5>{flight.departurePlace}</h5>
              </div>
              <div className="info-flight-route">
                <img src={line} alt="airplane route" />
                <h6>{flight.duration} - Direct</h6>
              </div>
              <div className="info-flight">
                <h2>{flight.arrivalTime}</h2>
                <h5>{flight.arrivalPlace}</h5>
              </div>
            </div>
            <img src={devider} alt="divider" className="divider" />
            <Link to={`flight-details/${flight.id}`} className="view-button">
              View Details
            </Link>
          </div>
```

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.

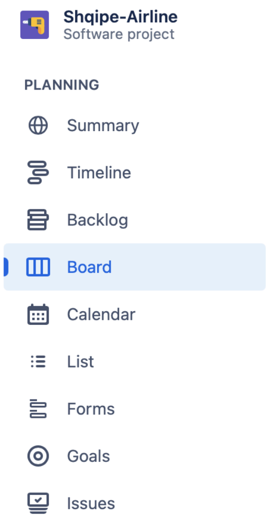
Figma:

- **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 Backtend

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.

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.

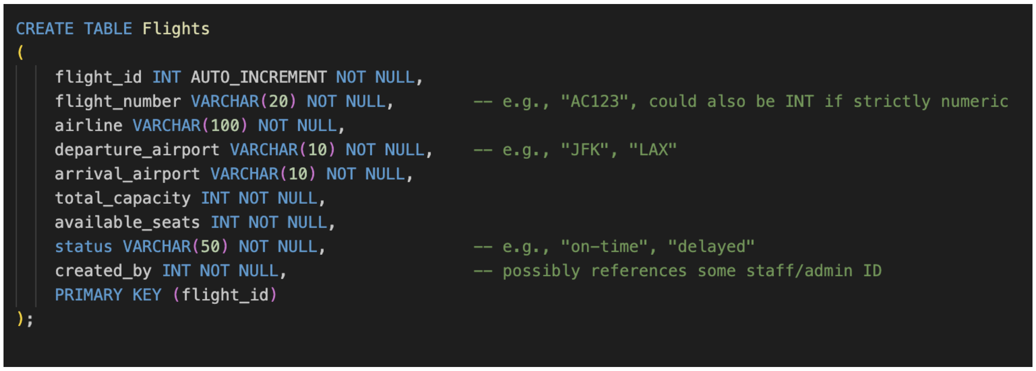
By using Python for the backend, the Shqipe Airline Management System ensures that the system is flexible, maintainable, and capable of handling complex tasks in a reliable manner.

8.2 Tech Stack

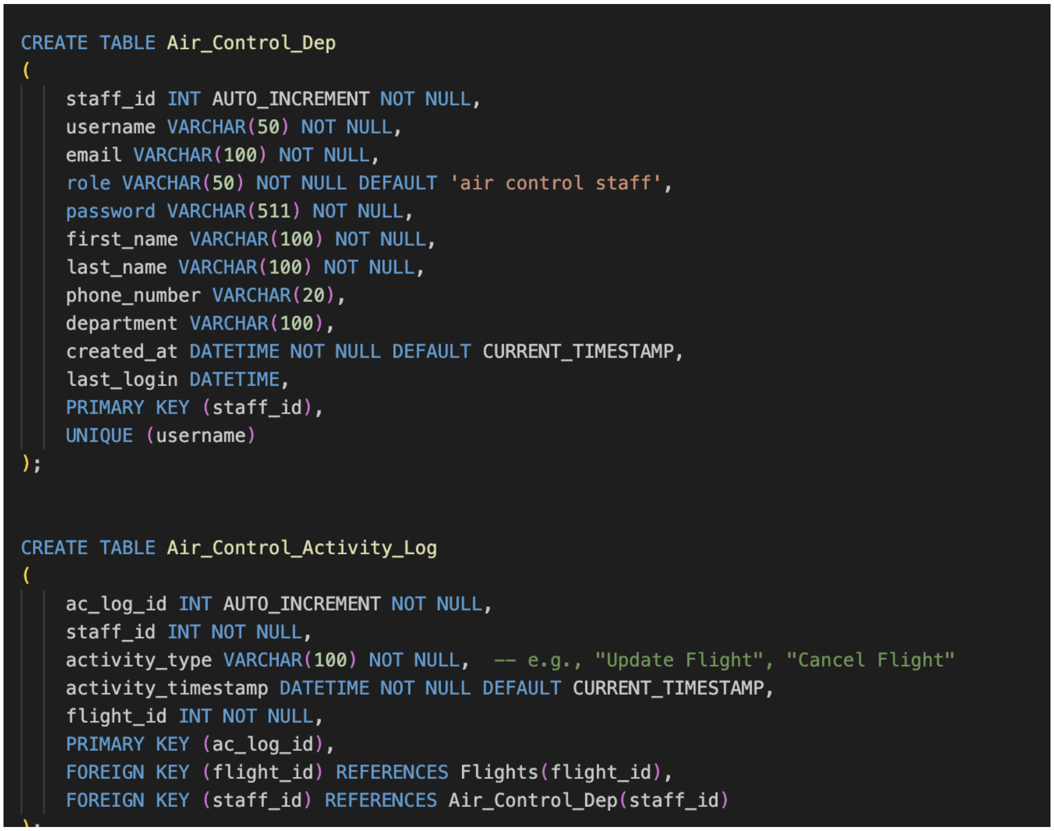
MySQL:

- 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.

Snippets of the code:



```sql
CREATE TABLE Flights
(
    flight_id INT AUTO_INCREMENT NOT NULL,
    flight_number VARCHAR(20) NOT NULL,        -- e.g., "AC123", could also be INT if strictly numeric
    airline VARCHAR(100) NOT NULL,
    departure_airport VARCHAR(10) NOT NULL,    -- e.g., "JFK", "LAX"
    arrival_airport VARCHAR(10) NOT NULL,
    total_capacity INT NOT NULL,
    available_seats INT NOT NULL,
    status VARCHAR(50) NOT NULL,               -- e.g., "on-time", "delayed"
    created_by INT NOT NULL,                   -- possibly references some staff/admin ID
    PRIMARY KEY (flight_id)
);
```

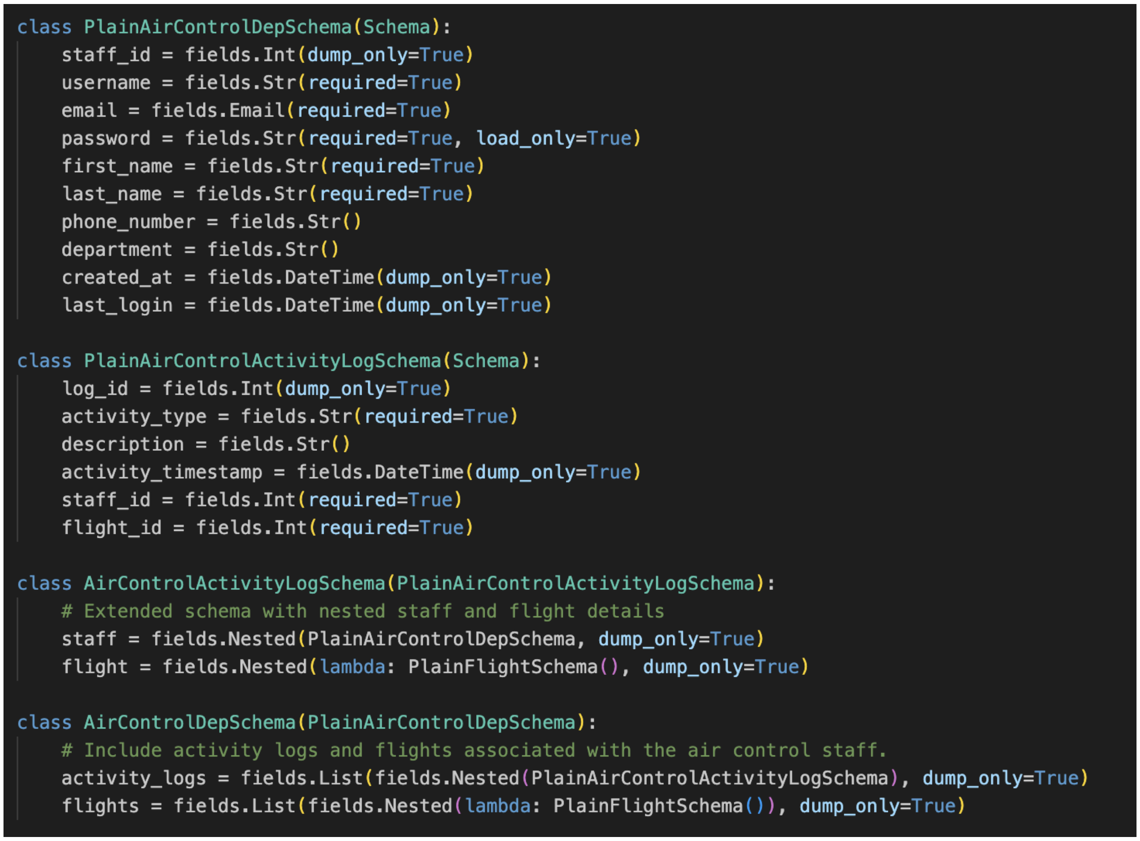


```sql
CREATE TABLE Air_Control_Dep
(
    staff_id INT AUTO_INCREMENT NOT NULL,
    username VARCHAR(50) NOT NULL,
    email VARCHAR(100) NOT NULL,
    role VARCHAR(50) NOT NULL DEFAULT 'air control staff',
    password VARCHAR(511) NOT NULL,
    first_name VARCHAR(100) NOT NULL,
    last_name VARCHAR(100) NOT NULL,
    phone_number VARCHAR(20),
    department VARCHAR(100),
    created_at DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
    last_login DATETIME,
    PRIMARY KEY (staff_id),
    UNIQUE (username)
);


CREATE TABLE Air_Control_Activity_Log
(
    ac_log_id INT AUTO_INCREMENT NOT NULL,
    staff_id INT NOT NULL,
    activity_type VARCHAR(100) NOT NULL,  -- e.g., "Update Flight", "Cancel Flight"
    activity_timestamp DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
    flight_id INT NOT NULL,
    PRIMARY KEY (ac_log_id),
    FOREIGN KEY (flight_id) REFERENCES Flights(flight_id),
    FOREIGN KEY (staff_id) REFERENCES Air_Control_Dep(staff_id)
);
```

Python:

- **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.

Snippet from the code:



```python
class PlainAirControlDepSchema(Schema):
    staff_id = fields.Int(dump_only=True)
    username = fields.Str(required=True)
    email = fields.Email(required=True)
    password = fields.Str(required=True, load_only=True)
    first_name = fields.Str(required=True)
    last_name = fields.Str(required=True)
    phone_number = fields.Str()
    department = fields.Str()
    created_at = fields.DateTime(dump_only=True)
    last_login = fields.DateTime(dump_only=True)

class PlainAirControlActivityLogSchema(Schema):
    log_id = fields.Int(dump_only=True)
    activity_type = fields.Str(required=True)
    description = fields.Str()
    activity_timestamp = fields.DateTime(dump_only=True)
    staff_id = fields.Int(required=True)
    flight_id = fields.Int(required=True)

class AirControlActivityLogSchema(PlainAirControlActivityLogSchema):
    # Extended schema with nested staff and flight details
    staff = fields.Nested(PlainAirControlDepSchema, dump_only=True)
    flight = fields.Nested(lambda: PlainFlightSchema(), dump_only=True)

class AirControlDepSchema(PlainAirControlDepSchema):
    # Include activity logs and flights associated with the air control staff.
    activity_logs = fields.List(fields.Nested(PlainAirControlActivityLogSchema), dump_only=True)
    flights = fields.List(fields.Nested(lambda: PlainFlightSchema()), dump_only=True)
```