**Software Requirements Specification**

**for**

**<**AIR WIZZ PROJECT**>**

**Version 1.0 approved**

**Prepared by Anas Ahmed**

**FAST NUCES**

**12-7-2024**

**Table of Contents**

**Table of Contents ii**

**Revision History ii**

**1. Introduction 1**

1.1 Purpose 1

1.2 Document Conventions 1

1.3 Intended Audience and Reading Suggestions 1

1.4 Product Scope 1

1.5 References 1

**2. Overall Description 2**

2.1 Product Perspective 2

2.2 Product Functions 2

2.3 User Classes and Characteristics 2

2.4 Operating Environment 2

2.5 Design and Implementation Constraints 2

2.6 User Documentation 2

2.7 Assumptions and Dependencies 3

**3. External Interface Requirements 3**

3.1 User Interfaces 3

3.2 Hardware Interfaces 3

3.3 Software Interfaces 3

3.4 Communications Interfaces 3

**4. System Features 4**

4.1 System Feature 1 4

4.2 System Feature 2 (and so on) 4

**5. Other Nonfunctional Requirements 4**

5.1 Performance Requirements 4

5.2 Safety Requirements 5

5.3 Security Requirements 5

5.4 Software Quality Attributes 5

5.5 Business Rules 5

**6. Other Requirements 5**

**Appendix A: Glossary 5**

**Appendix B: Analysis Models 5**

**Appendix C: To Be Determined List 6**

**Revision History**

| **Name** | **Date** | **Reason For Changes** | **Version** |
| --- | --- | --- | --- |
| Anas Ahmed | 7,12, 2024 | The initial draft of the SRS document for Air Wizz | 1.0 |
| Zain Abbas | 8,12,2024 | Updating doc with new data and progress | 2.0 |

### *1. Introduction*

#### 1.1 Purpose

*This Software Requirements Specification (SRS) document specifies the requirements for the Air Wizz flight ticket booking and management system. This document outlines the system's functionality, including the flight search, booking, user profile management, and admin features. The system is designed to allow users to book flights, manage their profiles, and track their booking history. Additionally, the administrative portal allows management of flights, users, and transactions. The current version of this document is Version 1.0, which provides the foundation for future iterations.*

#### 1.2 Document Conventions

*This document follows the standard conventions of technical specification writing, using the following formatting:*

* ***Bold*** *is used for key terms, system components, and important concepts.*
* *Italics is used for notes, examples, and additional clarifications.*
* ***Numbered lists*** *are used for listing requirements or features in a sequential order.*
* ***Priority Levels****: Requirements are categorized with priority levels of "High", "Medium", and "Low". Higher-level requirements are inherited by detailed requirements unless explicitly stated otherwise.*

*Whenever applicable, requirements will be marked with "Shall" to indicate a mandatory feature, "Should" for desired but not critical features, and "May" for optional features.*

#### 1.3 Intended Audience and Reading Suggestions

*This document is intended for a variety of audiences:*

* ***Developers****: They will focus on the detailed functional requirements to guide the implementation of the system.*
* ***Project Managers****: They should read the high-level functional requirements and timelines to understand the project scope and goals.*
* ***Marketing Staff****: They should review the product scope to understand the key features for user engagement and promotional strategies.*
* ***Users****: Although this document is not directly for end-users, they may refer to certain parts regarding user interaction flow.*
* ***Testers****: They should focus on the testable requirements and validation conditions outlined in the functional and non-functional sections.*

*The document is organized as follows:*

* ***Section 1****: Introduction, which provides an overview of the purpose, conventions, and intended audience.*
* ***Section 2****: Overall Description, explaining the general functionalities of the system.*
* ***Section 3****: System Features, detailing the functionalities and capabilities of the software.*
* ***Section 4****: External Interface Requirements, including user interface and software interfaces.*
* ***Section 5****: System Attributes, outlining performance, reliability, and security requirements.*
* ***Section 6****: Other Non-Functional Requirements, such as scalability and usability.*

*The suggested reading order is to begin with Sections 1 and 2 for an overview of the system, then move to Sections 3-5 for specific details related to development, testing, and deployment.*

#### 1.4 Product Scope

*The Air Wizz system is a web-based platform that simplifies the process of booking flights. It allows users to search for available flights, view flight details, make bookings, and manage their profiles. Admin users can control flight availability, manage bookings, and monitor user activity. The software is designed to be easy to use for the general public, while also providing the necessary tools for airline and customer management.*

*The main goals of Air Wizz include:*

* ***Streamlining flight bookings*** *through an intuitive, easy-to-navigate interface.*
* ***Improving customer satisfaction*** *by offering a simple, accessible platform for flight reservations.*
* ***Facilitating airline operations*** *with an admin portal to manage flight schedules and user information.*
* ***Ensuring security*** *in payment processing and personal data management.*

*This system is aligned with the goal of automating flight booking and management for airlines, providing a cost-effective solution to both users and airline staff.*

#### 1.5 References

* *Oracle SQL Database Documentation, Oracle Corporation, Version 12c, July 2014.*
* *".NET Core Framework Documentation," Microsoft, Version 5.0, June 2021.*
* *"Software Engineering: A Practitioner’s Approach" by Roger S. Pressman, 10th Edition, McGraw-Hill, 2014.*
* *User Interface Style Guide, Apple Inc., Version 3.0, December 2020.*
* *"Air Wizz: Vision and Scope Document," [Author], FAST NUCES, November 2024.*

### *2. Overall Description*

#### 2.1 Product Perspective

*The Air Wizz flight booking and management system is a new, self-contained product that does not replace any existing systems but aims to streamline the process of booking flights and managing reservations for both users and airlines. It will serve as a comprehensive solution for flight bookings, allowing users to search, book, and manage their flights seamlessly while providing administrators with tools to manage flight schedules and bookings.*

*This system is designed to integrate with external payment processing systems to facilitate secure online payments. It is also expected to interface with airline databases for real-time flight availability updates. The system will operate entirely over the web, and users will access it through a browser-based interface.*

*A high-level overview of the major system components includes:*

* ***User Interface (UI)****: The web interface for customers to search for flights, make bookings, and manage profiles.*
* ***Admin Portal****: A secure interface for airline staff to manage flights, bookings, and user information.*
* ***Payment Gateway Integration****: A component for handling secure payment transactions.*
* ***Database****: The back-end system to manage flight data, user data, and transaction history.*

#### 2.2 Product Functions

*The major functions that Air Wizz must perform include:*

* ***User Features****:*
  + *Search for available flights based on destinations, dates, and other preferences.*
  + *View flight details, including pricing, flight timings, and seat availability.*
  + *Book flights and complete transactions securely via an integrated payment gateway.*
  + *Manage user profile (view past bookings, update personal details).*
  + *View booking history and cancel bookings if applicable.*
* ***Admin Features****:*
  + *Add, update, or delete flight information (flight schedules, prices, etc.).*
  + *Manage customer bookings (view, modify, cancel bookings).*
  + *Access analytics for flight bookings, user activity, and system performance.*
  + *Administer payment transactions and monitor system health.*

#### 2.3 User Classes and Characteristics

*The Air Wizz system will be used by various user classes, each with different levels of access and functionality:*

* ***End Users****:*
  + ***Characteristics****: General public, tech-savvy individuals, and occasional travelers.*
  + ***Usage****: Frequent and occasional use, primarily for booking flights and managing profiles.*
  + ***Privileges****: Can search flights, make bookings, and view/edit profiles.*
* ***Admins****:*
  + ***Characteristics****: Airline staff with administrative access.*
  + ***Usage****: Frequent use for managing flight schedules, user accounts, and transactions.*
  + ***Privileges****: Full system access, including managing flights and bookings, and accessing transaction data.*
* ***System Administrators****:*
  + ***Characteristics****: IT staff responsible for maintaining the system.*
  + ***Usage****: Infrequent use but essential for system monitoring and troubleshooting.*
  + ***Privileges****: Access to server logs, backups, and system configuration settings.*

#### 2.4 Operating Environment

*The Air Wizz system will operate in the following environment:*

* ***Hardware Platform****: Web server and database server capable of supporting web applications and storing flight and user data. The hardware should be scalable to accommodate increasing traffic and data.*
* ***Operating System****: The system will run on Linux-based servers for the web application and database, with support for major browsers (Chrome, Firefox, Safari) for the client interface.*
* ***Database****: Oracle or MySQL, used for storing flight data, user information, and transactions.*
* ***Web Server****: Apache or Nginx, used for serving the web interface.*
* ***Other Software****: Payment gateway integration (e.g., PayPal, Stripe), and analytics tools.*

#### 2.5 Design and Implementation Constraints

*The development of Air Wizz is subject to the following constraints:*

* ***Payment Gateway Integration****: The system must comply with relevant security standards (e.g., PCI-DSS) for handling online payments.*
* ***Regulatory Compliance****: The system must adhere to data protection regulations such as GDPR for user data privacy.*
* ***Browser Compatibility****: The front-end UI must be compatible with modern web browsers (Chrome, Firefox, Safari) to ensure broad accessibility.*
* ***Technology Stack****: The system will use* ***.NET Core*** *for back-end development and* ***React.js*** *or* ***Angular*** *for the front-end.*
* ***Scalability****: The system must be designed to handle future growth in user traffic and data without significant performance degradation.*

#### 2.6 User Documentation

*The following user documentation will be provided with the Air Wizz system:*

* ***User Manual****: A comprehensive guide for end users, covering flight search, booking, and profile management.*
* ***Admin Manual****: A separate guide for administrative users, detailing how to manage flights, users, and bookings.*
* ***Online Help****: Contextual help integrated into the UI for quick user reference.*
* ***Tutorial Videos****: Step-by-step video tutorials for both end users and admins to assist with basic operations.*

#### 2.7 Assumptions and Dependencies

*The following assumptions and dependencies are relevant for the Air Wizz system:*

* ***Third-Party Payment Gateway****: The system assumes the availability of third-party payment providers (e.g., PayPal, Stripe) for processing transactions. If these providers change or are unavailable, the system may face delays.*
* ***Real-Time Flight Data****: The system assumes integration with airline databases or third-party services to fetch real-time flight availability. Delays or changes in API availability could impact system functionality.*
* ***Internet Connectivity****: The system depends on users having stable internet connections to interact with the platform and complete bookings.*
* ***Regulatory Changes****: Any changes in data protection laws (e.g., GDPR) may necessitate updates to the system to ensure compliance.*

## Process Model

The process model for this project follows a Modified Waterfall Model, emphasizing structured, sequential development while allowing feedback loops for flexibility. Key stages include Requirement Gathering, where user needs like flight search, booking, and admin management are defined; Design, focusing on system architecture (N-tier structure) and UI/UX planning; Development, with iterative coding for front-end, back-end, and database integration; Testing, covering unit, integration, and system testing for reliability; and Deployment and Maintenance, ensuring the live system is secure, scalable, and updated as needed.

## Project Plan

Requirement Analysis to gather user and system needs, followed by Design and Development of the system's N-tier architecture, UI, and functionality. Testing and Deployment ensure the system meets quality standards, with ongoing Maintenance to address updates and scalability. Each phase has defined deliverables and timelines for efficient execution.

## Feasibility Report

The system's feasibility analysis ensures viability across key dimensions: Technical Feasibility, confirming compatibility with N-tier architecture and integration of APIs; Operational Feasibility, ensuring user-friendly interfaces for end-users and administrators; and Economic Feasibility, showing cost-effectiveness with available resources and tools. Legal Feasibility ensures compliance with data privacy and security standards. The project demonstrates strong scalability for future enhancements and manageable risks with planned mitigation strategies, making it feasible for development and deployment.

## Homogenization Process

ensures uniformity across system components by standardizing data formats, API communication protocols, and user interfaces. It aligns diverse modules like flight search, booking, and admin management into a cohesive architecture. This promotes seamless integration, consistent performance, and user-friendly interaction throughout the system.

## Use case descriptions

User Registration and Login: Allows users to create accounts, securely log in, and manage sessions. Handles incorrect login attempts with error feedback.

Flight Search and Details Viewing: Enables users to search for flights by criteria like destination and date, and view detailed flight information.

Basic Flight Booking System: Users can book flights, confirm bookings, and receive notifications while ensuring seat availability is updated.

Booking History Access: Displays a user’s past bookings with essential details like route, date, and payment status for easy reference.

Admin Flight Management: Admins can add, update, or delete flights, manage schedules, and view flight lists through a secure interface.

Flight Tracker: Fetches real-time flight data via API, displaying flight status and positions dynamically on an interactive map.

Currency Conversion: Converts flight prices into the user’s selected currency using real-time conversion rates fetched from an external API.

AI Help Chatbot Interaction: Provides instant assistance for user queries on flights, bookings, and policies, escalating complex issues when necessary.

Logout Functionality: Securely ends the user session, clears session data, and redirects the user to the login page to prevent unauthorized access.

## Video URL

<https://www.youtube.com/watch?v=DpEsAi5dlZk>

### *3. External Interface Requirements*

#### 3.1 User Interfaces

*The user interfaces (UI) for the Air Wizz system are designed to provide an intuitive, responsive, and user-friendly experience. Key features include:*

* ***Flight Search Interface****: A search bar where users can enter their departure and destination cities, dates, and the number of passengers. This interface will return available flights, along with relevant details such as price, time, and available seats.*
* ***Flight Booking Interface****: A clean layout displaying selected flight details, price breakdowns, and seat selection. The user can proceed with booking by entering personal information and making a payment.*
* ***Profile Management Interface****: A dashboard for users to view and edit their profiles, past bookings, and payment methods.*
* ***Admin Dashboard****: A centralized interface for administrators to manage flight data, bookings, and users, with a clean and organized layout.*
* ***Error Message Standards****: Error messages should be clear and concise, appearing in red text with a brief description of the issue. They should be followed by suggested actions (e.g., "Please check your payment details and try again").*

*All UIs must adhere to the following:*

* ***Style Guide****: The UI will follow modern web design principles, with responsive layouts that adapt to desktop and mobile views. The layout will use a simple, flat design with minimalistic elements.*
* ***Standard Buttons and Functions****: Each page will have a "Help" button for user assistance, a "Log Out" button in the top-right corner, and a "Back" button where appropriate.*
* ***Screen Layout****: Key elements such as flight search, booking details, and user profiles must be organized logically, with the most important functions displayed prominently. Buttons will be uniform across the platform, with distinct calls to action (e.g., "Book Now" or "View Details").*
* ***Keyboard Shortcuts****: Shortcuts for common actions (e.g., "Ctrl+S" for saving a profile, "Esc" to cancel actions) will be supported.*
* ***User Interface Design****: Detailed UI designs will be documented separately in a UI specification document.*

#### 3.2 Hardware Interfaces

*The Air Wizz system is primarily software-based, but it will interact with the following hardware components:*

* ***Server Hardware****: The system will run on web and database servers. The servers must support a minimum of* ***8 GB RAM****,* ***4-core CPU****, and* ***100 GB of storage****. Servers will handle both web and application layers.*
* ***User Devices****: Users will interact with the system through personal devices, including desktops, laptops, smartphones, and tablets. Supported devices should be able to run modern web browsers (Chrome, Firefox, Safari).*
* ***Point-of-Sale Devices (Admin)****: The system may interface with admin-level POS devices for managing in-person bookings or payments. These devices should communicate via standard network protocols to sync with the server.*
* ***Barcode Scanners****: If required, barcode scanners may be used to scan booking confirmation or flight tickets for offline verification.*

#### 3.3 Software Interfaces

*The Air Wizz system will integrate with various software components:*

* ***Web Server****: The front-end application will be hosted on an Apache or Nginx web server.*
* ***Database****: The system will interact with an* ***Oracle*** *or* ***MySQL*** *database to manage flight schedules, user profiles, booking history, and payment information.*
* ***Payment Gateway API****: The system will use external APIs for payment processing, such as* ***PayPal*** *or* ***Stripe****, for secure online transactions. These APIs will handle communication for payment authorizations, confirmations, and errors.*
* ***Email Service****: The system will integrate with an email service (e.g.,* ***SendGrid****) for sending booking confirmations, flight updates, and promotional offers.*
* ***Operating System****: The server-side components of Air Wizz will run on a* ***Linux-based operating system*** *(e.g., Ubuntu or CentOS).*
* ***Other Software****:*
  + ***Analytics Tools*** *(e.g.,* ***Google Analytics****): For monitoring user behavior and system performance.*
  + ***Web Browser****: The software must be compatible with common web browsers such as* ***Google Chrome****,* ***Mozilla Firefox****, and* ***Safari*** *for the end users.*

*The system will communicate with the above-mentioned software components via* ***RESTful APIs*** *and standard web communication protocols.*

#### 3.4 Communications Interfaces

*The Air Wizz system must support several communication interfaces:*

* ***Web Communication Protocol****: The system will use* ***HTTP/HTTPS*** *for communication between the client (user) and the web server, ensuring secure data transmission via SSL/TLS encryption.*
* ***Email Communication****: Confirmation emails, flight updates, and promotional messages will be sent via email using an external email service (e.g.,* ***SendGrid****). These messages will follow the standard email formatting.*
* ***Real-Time Flight Data Updates****: The system will use* ***API calls*** *to fetch real-time flight information from external flight data providers. This requires a stable internet connection and standard web communication protocols (RESTful APIs).*
* ***Payment Communication****: Payment transactions will occur over* ***HTTPS****, using secure encryption and communication protocols supported by payment gateways (e.g.,* ***Stripe*** *or* ***PayPal API****).*
* ***Data Transfer****: All data between the system and external services (e.g., payment gateways, flight data providers) will be transferred securely with* ***SSL encryption*** *to ensure confidentiality and integrity.*
* ***Security****: Sensitive data, such as payment details and user information, will be encrypted both at rest and in transit. The system will comply with* ***PCI-DSS*** *standards for secure payment processing.*

### 4. System Features

### 4.1 User Registration and Login

#### 4.1.1 Description and Priority

This feature allows users to create an account with basic details such as name, email, and password. The login process ensures secure authentication by hashing passwords. The system also includes session handling for users who are logged in, error messages for incorrect login attempts, and a "Forgot Password" option for resetting accounts.  
**Priority**: High  
**Benefit**: 9  
**Penalty**: 8  
**Cost**: 7  
**Risk**: 6

#### 4.1.2 Stimulus/Response Sequences

* **Stimulus 1**: A user navigates to the registration page.
  + **Response 1**: The system displays the registration form, prompting the user to enter details like email, name, and password.
* **Stimulus 2**: The user submits the registration form.
  + **Response 2**: The system stores the user details in the database and creates a new user account.
* **Stimulus 3**: The user attempts to log in.
  + **Response 3**: The system verifies the entered credentials, and if correct, grants access to the user dashboard.
* **Stimulus 4**: The user clicks the "Forgot Password" link.
  + **Response 4**: The system sends a password reset link to the user's email.

#### 4.1.3 Functional Requirements

* **REQ-1**: The system must allow users to create accounts by providing basic details.
* **REQ-2**: The system must hash passwords for security purposes.
* **REQ-3**: The system must handle user sessions securely.
* **REQ-4**: The system must provide error messages for incorrect login attempts.
* **REQ-5**: The system must allow users to reset their passwords via email (if implemented).

### 4.2 Flight Search

#### 4.2.1 Description and Priority

This feature enables users to search for available flights based on origin, destination, and travel date. It includes a class selection dropdown for different flight classes (e.g., Economy, Business) and updates the flight results as the user inputs data. The feature implements basic validation to ensure correct inputs.  
**Priority**: High  
**Benefit**: 9  
**Penalty**: 6  
**Cost**: 7  
**Risk**: 5

#### 4.2.2 Stimulus/Response Sequences

* **Stimulus 1**: The user enters the origin, destination, and date in the search form.
  + **Response 1**: The system displays a list of available flights matching the search criteria.
* **Stimulus 2**: The user selects a flight class from the dropdown menu.
  + **Response 2**: The system filters the available flights according to the selected class.
* **Stimulus 3**: The user adjusts search parameters (e.g., origin, destination).
  + **Response 3**: The system dynamically updates the flight list based on the new input.

#### 4.2.3 Functional Requirements

* **REQ-1**: The system must allow users to input origin, destination, and date for flight searches.
* **REQ-2**: The system must provide a dropdown for selecting flight classes.
* **REQ-3**: The system must validate the input (e.g., date format) before proceeding.
* **REQ-4**: The system must dynamically update the flight list based on user input.

### 4.3 Basic Booking System

#### 4.3.1 Description and Priority

This feature enables users to select a flight from search results and proceed with booking. A confirmation page displays booking details before finalizing. The system stores booking data for future reference and includes a simple payment placeholder (e.g., Cash on Delivery or dummy payment).  
**Priority**: High  
**Benefit**: 9  
**Penalty**: 6  
**Cost**: 8  
**Risk**: 6

#### 4.3.2 Stimulus/Response Sequences

* **Stimulus 1**: The user selects a flight from the search results.
  + **Response 1**: The system shows a booking confirmation page with details like flight number, date, and price.
* **Stimulus 2**: The user submits the booking form.
  + **Response 2**: The system stores the booking details in the database and displays a confirmation message.
* **Stimulus 3**: The user proceeds to payment.
  + **Response 3**: The system displays a placeholder for payment (e.g., Cash on Delivery).

#### 4.3.3 Functional Requirements

* **REQ-1**: The system must allow users to select a flight from the search results.
* **REQ-2**: The system must display a booking confirmation page.
* **REQ-3**: The system must store booking data for future reference.
* **REQ-4**: The system must provide a placeholder for payment options.
* **REQ-5**: The system must display a booking confirmation message.

### 4.4 Admin Flight Management

#### 4.4.1 Description and Priority

Admins can manage flight records by adding new flights, editing existing ones, and removing those no longer needed. This feature helps keep the system up-to-date and allows admins to monitor and manage bookings efficiently.  
**Priority**: High  
**Benefit**: 9  
**Penalty**: 7  
**Cost**: 6  
**Risk**: 7

#### 4.4.2 Stimulus/Response Sequences

* **Stimulus 1**: The admin logs in to the admin portal.
  + **Response 1**: The system displays the admin dashboard with options to manage flights, bookings, and users.
* **Stimulus 2**: The admin clicks on "Manage Flights" to add a new flight.
  + **Response 2**: The system prompts the admin to input flight details (e.g., airline, departure time, price, etc.).
* **Stimulus 3**: The admin submits the flight details.
  + **Response 3**: The system adds the new flight to the database and displays a confirmation message.
* **Stimulus 4**: The admin selects an existing flight to edit or remove.
  + **Response 4**: The system allows the admin to update flight details or remove the flight from the system.

#### 4.4.3 Functional Requirements

* **REQ-1**: The system must allow admins to log in to the admin portal with proper authentication.
* **REQ-2**: The system must provide an interface for adding, editing, and removing flights.
* **REQ-3**: The system must validate flight data before saving (e.g., correct dates, pricing).
* **REQ-4**: The system must display a confirmation message after adding or editing a flight.
* **REQ-5**: The system must log administrative actions for auditing purposes.

### 4.5 Flight Database Integration

#### 4.5.1 Description and Priority

This feature integrates and manages the flight database, including users, bookings, and administrative logs. It ensures efficient data storage and querying through proper normalization and relationships between entities.  
**Priority**: High  
**Benefit**: 9  
**Penalty**: 7  
**Cost**: 8  
**Risk**: 6

#### 4.5.2 Stimulus/Response Sequences

* **Stimulus 1**: The admin adds a new flight.
  + **Response 1**: The system stores the flight data in the database.
* **Stimulus 2**: The user makes a booking.
  + **Response 2**: The system updates the user’s booking history and flight availability in the database.
* **Stimulus 3**: The admin updates flight prices.
  + **Response 3**: The system updates the flight price in the database and reflects it in user bookings.

#### 4.5.3 Functional Requirements

* **REQ-1**: The system must store flight details, user profiles, booking records, and administrative actions.
* **REQ-2**: The system must maintain normalized database tables for efficiency.
* **REQ-3**: The system must ensure consistency between flight availability and booking data.
* **REQ-4**: The system must use primary and foreign keys to ensure proper relationships and data integrity.

### 4.6 Booking History

#### 4.6.1 Description and Priority

This feature allows users to view their booking history, including past flights, booking dates, and status (confirmed, canceled, etc.). It helps users keep track of their travel experiences and manage any upcoming trips.  
**Priority**: Medium  
**Benefit**: 7  
**Penalty**: 6  
**Cost**: 5  
**Risk**: 4

#### 4.6.2 Stimulus/Response Sequences

* **Stimulus 1**: The user logs into their account.
  + **Response 1**: The system displays the user dashboard with an option to view booking history.
* **Stimulus 2**: The user selects "Booking History."
  + **Response 2**: The system shows a list of past bookings with details such as flight number, date, and status.
* **Stimulus 3**: The user selects a booking to view details.
  + **Response 3**: The system displays detailed information about the selected booking, including booking date, flight details, and current status.

#### 4.6.3 Functional Requirements

* **REQ-1**: The system must allow users to view their booking history.
* **REQ-2**: The system must display booking details such as flight number, date, and status.
* **REQ-3**: The system must allow users to click on any past booking to view detailed information.
* **REQ-4**: The system must ensure that booking history is displayed in chronological order.

### 4.7 Admin User Management

#### 4.7.1 Description and Priority

Admins can manage user accounts, including viewing user profiles, deactivating accounts, or resetting passwords. This feature helps maintain the security of the platform by allowing admins to monitor user activities and control access.  
**Priority**: High  
**Benefit**: 8  
**Penalty**: 6  
**Cost**: 7  
**Risk**: 5

#### 4.7.2 Stimulus/Response Sequences

* **Stimulus 1**: The admin logs into the admin portal.
  + **Response 1**: The system displays the admin dashboard with options to manage users.
* **Stimulus 2**: The admin clicks on "User Management."
  + **Response 2**: The system displays a list of all registered users.
* **Stimulus 3**: The admin selects a user from the list.
  + **Response 3**: The system displays the selected user’s profile with options to deactivate the account, reset password, or view booking history.
* **Stimulus 4**: The admin selects "Deactivate" or "Reset Password."
  + **Response 4**: The system performs the action, and a confirmation message is displayed.

#### 4.7.3 Functional Requirements

* **REQ-1**: The system must allow admins to view a list of all registered users.
* **REQ-2**: The system must allow admins to view detailed user profiles.
* **REQ-3**: The system must allow admins to deactivate or reset passwords for user accounts.
* **REQ-4**: The system must display confirmation messages after performing actions on user accounts.
* **REQ-5**: The system must ensure that only authorized admins can access the User Management page.

### 4.8 Flight Tracker

#### 4.8.1 Description and Priority

The Flight Tracker fetches real-time data for the 100 latest flights using an API. It displays essential flight details, including flight name, number, status, and current position. The system maps the flight positions dynamically, providing a visual representation on a map for users. It features a simple, user-friendly interface and ensures data accuracy by periodically refreshing flight details.  
**Priority**: High  
**Benefit**: 9  
**Penalty**: 7  
**Cost**: 8  
**Risk**: 6

#### 4.8.2 Stimulus/Response Sequences

* **Stimulus 1**: The user accesses the Flight Tracker section of the website.
  + **Response 1**: The system fetches real-time data for the 100 latest flights from the API and displays them in a list with flight details (name, number, status).
* **Stimulus 2**: The user clicks on a specific flight from the list.
  + **Response 2**: The system dynamically maps the flight’s current position on a visual map for easy tracking.
* **Stimulus 3**: The system detects that flight data has become outdated.
  + **Response 3**: The system automatically refreshes the flight details to ensure accuracy.

#### 4.8.3 Functional Requirements

* **REQ-1**: The system must fetch real-time flight data from an API for the 100 latest flights.
* **REQ-2**: The system must display flight details including flight name, number, status, and current position.
* **REQ-3**: The system must dynamically map flight positions on a visual map.
* **REQ-4**: The system must periodically refresh flight data to ensure accuracy.
* **REQ-5**: The system must provide a user-friendly interface for easy browsing of flight information.

### 4.9 Flight Seat Availability

#### 4.9.1 Description and Priority

This feature allows users to view available seats for flights and select a seat when booking. It also includes functionality for admins to update seat availability after bookings are made.  
**Priority**: High  
**Benefit**: 8  
**Penalty**: 7  
**Cost**: 6  
**Risk**: 6

#### 4.9.2 Stimulus/Response Sequences

* **Stimulus 1**: The user selects a flight and proceeds to booking.
  + **Response 1**: The system displays available seats for the selected flight.
* **Stimulus 2**: The user selects a seat from the available options.
  + **Response 2**: The system reserves the seat for the user and updates seat availability.
* **Stimulus 3**: The user finalizes the booking.
  + **Response 3**: The system stores the booking details, including the selected seat, and updates the flight’s available seat count.
* **Stimulus 4**: The admin updates seat availability.
  + **Response 4**: The system reflects the updated seat availability in real-time for users.

#### 4.9.3 Functional Requirements

* **REQ-1**: The system must display available seats for selected flights.
* **REQ-2**: The system must allow users to select a seat from available options.
* **REQ-3**: The system must update seat availability upon booking.
* **REQ-4**: The system must allow admins to modify seat availability and update the database accordingly.
* **REQ-5**: The system must prevent users from booking already reserved seats.

### 4.10 User Profile Management

#### 4.10.1 Description and Priority

This feature allows users to update and manage their personal information, including name, email, and contact details. Users can also update their password or delete their account if needed.  
**Priority**: Medium  
**Benefit**: 7  
**Penalty**: 6  
**Cost**: 5  
**Risk**: 4

#### 4.10.2 Stimulus/Response Sequences

* **Stimulus 1**: The user logs into their account.
  + **Response 1**: The system displays the user dashboard with an option to view or edit profile.
* **Stimulus 2**: The user selects "Edit Profile."
  + **Response 2**: The system displays the user's current profile information and allows changes.
* **Stimulus 3**: The user updates their information and submits the form.
  + **Response 3**: The system saves the updated details and displays a confirmation message.
* **Stimulus 4**: The user selects "Delete Account."
  + **Response 4**: The system asks for confirmation and deletes the account if confirmed.

#### 4.10.3 Functional Requirements

* **REQ-1**: The system must allow users to view and edit their profile information.
* **REQ-2**: The system must validate the new information before saving changes.
* **REQ-3**: The system must allow users to change their password.
* **REQ-4**: The system must ask for confirmation before deleting an account.
* **REQ-5**: The system must display a success message after updating profile information.

### 4.11 AI Help Chatbot

#### 4.11.1 Description and Priority

The AI Help Chatbot integrates Dialogflow to provide users with instant assistance. It answers queries related to flights, bookings, and policies, offering step-by-step guidance for common tasks like booking. The chatbot features natural language understanding for seamless interaction and escalates more complex issues to human support.  
**Priority**: High  
**Benefit**: 9  
**Penalty**: 7  
**Cost**: 8  
**Risk**: 5

#### 4.11.2 Stimulus/Response Sequences

* **Stimulus 1**: The user clicks on the chatbot icon on the website.
  + **Response 1**: The system opens the chatbot interface and prompts the user to ask a question.
* **Stimulus 2**: The user asks about flight booking.
  + **Response 2**: The chatbot provides step-by-step guidance for booking a flight.
* **Stimulus 3**: The user asks a complex question.
  + **Response 3**: The system escalates the issue to human support for further assistance.

#### 4.11.3 Functional Requirements

* **REQ-1**: The system must integrate with Dialogflow for natural language processing.
* **REQ-2**: The chatbot must provide answers to common user queries like booking and flight policies.
* **REQ-3**: The system must escalate unresolved issues to human support.
* **REQ-4**: The chatbot must display a user-friendly interface with clear responses.

### 4.12 Data Validation

#### 4.12.1 Description and Priority

Data validation ensures that user inputs in forms such as registration, login, and booking are accurate and secure. It validates email addresses and phone numbers, prevents SQL injection through prepared statements, handles errors gracefully, and enhances overall data integrity.  
**Priority**: High  
**Benefit**: 8  
**Penalty**: 6  
**Cost**: 7  
**Risk**: 6

#### 4.12.2 Stimulus/Response Sequences

* **Stimulus 1**: The user submits the registration form.
  + **Response 1**: The system validates the email and phone number inputs, ensuring they are valid.
* **Stimulus 2**: The user enters a potentially malicious SQL command in a booking form.
  + **Response 2**: The system uses prepared statements to prevent SQL injection.
* **Stimulus 3**: The system detects an invalid input.
  + **Response 3**: The system displays an error message, guiding the user to correct the issue.

#### 4.12.3 Functional Requirements

* **REQ-1**: The system must validate email and phone number formats to ensure they are correct.
* **REQ-2**: The system must prevent SQL injection using prepared statements for form submissions.
* **REQ-3**: The system must display user-friendly error messages if invalid data is entered.
* **REQ-4**: The system must ensure overall data integrity by validating inputs across the platform.

### 4.13 Logout Functionality

#### 4.13.1 Description and Priority

This feature allows users to securely log out of their account, clearing user data from the session and ensuring that sessions cannot be reused post-logout. After logout, users are redirected to the login page and unauthorized access to the dashboard is prevented.  
**Priority**: High  
**Benefit**: 8  
**Penalty**: 6  
**Cost**: 5  
**Risk**: 4

#### 4.13.2 Stimulus/Response Sequences

* **Stimulus 1**: The user clicks the logout button.
  + **Response 1**: The system securely clears the user's session data.
* **Stimulus 2**: The user attempts to access the dashboard after logging out.
  + **Response 2**: The system redirects the user to the login page.
* **Stimulus 3**: The user is logged out successfully.
  + **Response 3**: The system displays a confirmation message or redirects to the login page.

#### 4.13.3 Functional Requirements

* **REQ-1**: The system must allow users to log out securely.
* **REQ-2**: The system must clear all user session data upon logout.
* **REQ-3**: The system must prevent unauthorized access to the dashboard after logout.
* **REQ-4**: The system must redirect users to the login page post-logout.

### 4.14 Flight Availability Check

#### 4.14.1 Description and Priority

This feature ensures that seats are available for a flight before confirming a booking. It blocks duplicate bookings by the same user and updates seat availability in the database after each booking. The system notifies users when flights are sold out and ensures data consistency between the UI and backend.  
**Priority**: High  
**Benefit**: 9  
**Penalty**: 7  
**Cost**: 6  
**Risk**: 5

#### 4.14.2 Stimulus/Response Sequences

* **Stimulus 1**: The user attempts to book a flight.
  + **Response 1**: The system checks seat availability and prevents booking if seats are unavailable.
* **Stimulus 2**: The user selects a flight that has already been booked.
  + **Response 2**: The system prevents the duplicate booking and displays a relevant message.
* **Stimulus 3**: The user completes a booking.
  + **Response 3**: The system updates seat availability and confirms the booking.

#### 4.14.3 Functional Requirements

* **REQ-1**: The system must check seat availability before confirming a booking.
* **REQ-2**: The system must block duplicate bookings for the same flight by the same user.
* **REQ-3**: The system must update seat availability in the database after each booking.
* **REQ-4**: The system must notify users when a flight is sold out.

### 4.15 Basic Error Handling

#### 4.15.1 Description and Priority

This feature implements structured exception handling to manage errors efficiently. It ensures that critical errors are addressed appropriately, non-critical errors are handled gracefully with user-friendly messages, and no sensitive information is exposed. The system logs errors for debugging to maintain a seamless user experience.  
**Priority**: Medium  
**Benefit**: 7  
**Penalty**: 6  
**Cost**: 6  
**Risk**: 5

#### 4.15.2 Stimulus/Response Sequences

* **Stimulus 1**: The system encounters an error during a flight booking process.
  + **Response 1**: The system throws an exception and logs the error.
* **Stimulus 2**: The user sees an error message on the UI.
  + **Response 2**: The system displays a user-friendly error message, excluding sensitive information.
* **Stimulus 3**: The system detects a non-critical error (e.g., minor validation issue).
  + **Response 3**: The system prompts the user to correct the issue without interrupting the flow.

#### 4.15.3 Functional Requirements

* **REQ-1**: The system must use structured exception handling to address errors.
* **REQ-2**: The system must log all errors for debugging purposes.
* **REQ-3**: The system must display clear and user-friendly error messages.
* **REQ-4**: The system must avoid exposing sensitive information in error messages.

### *5 Other Nonfunctional Requirements*

#### 5.1 Performance Requirements

* ***REQ-1****: The system should load the flight search results within 3 seconds of user input, ensuring a responsive and smooth user experience.*
* ***REQ-2****: The Flight Tracker must update flight data every 30 seconds to ensure real-time accuracy without overloading the system.*
* ***REQ-3****: The system must handle at least 500 concurrent users without significant performance degradation.*
* ***REQ-4****: Flight booking confirmations should be processed and displayed within 5 seconds of completing payment or selection to ensure users receive timely feedback.*

***Rationale****: These performance requirements ensure that the system delivers a fast, responsive user experience while managing server load and processing data efficiently, especially when dealing with real-time updates and multiple concurrent users.*

#### 5.2 Safety Requirements

* ***REQ-1****: The system must encrypt sensitive data, such as user passwords, payment information, and personal details, using industry-standard encryption methods (e.g., AES-256) to protect against data breaches.*
* ***REQ-2****: Users must be alerted in case of any system errors or failures that could potentially lead to the loss of booking data or payment information, with immediate steps to rectify the issue.*
* ***REQ-3****: The system must include an automatic backup mechanism to secure booking and user data every 24 hours, ensuring recovery in case of system failure.*

***Rationale****: These safety requirements protect both user and system data from being compromised or lost due to unforeseen issues, ensuring that the product remains reliable and trustworthy.*

#### 5.3 Security Requirements

* ***REQ-1****: The system must require multi-factor authentication (MFA) for users accessing sensitive information, such as booking history, payment details, and administrative functions.*
* ***REQ-2****: All user communications with the system (including login, payment, and booking processes) must be transmitted over HTTPS to ensure secure, encrypted data transmission.*
* ***REQ-3****: The system must implement role-based access control (RBAC) to ensure that users can only access features and data appropriate to their roles (e.g., admins, users, guest users).*
* ***REQ-4****: The system must comply with the General Data Protection Regulation (GDPR) or equivalent local data protection laws to ensure privacy and proper handling of user data.*

***Rationale****: These security requirements ensure that the product complies with best practices for protecting user data, ensuring secure access to user accounts and administrative functions while complying with legal data privacy standards.*

#### 5.4 Software Quality Attributes

* ***REQ-1****:* ***Usability****: The system must be easy to use, with an intuitive interface that allows users to book flights and manage their accounts with minimal training or guidance. A usability score of 80% or higher, as measured by user satisfaction surveys, should be achieved.*
* ***REQ-2****:* ***Reliability****: The system must have an uptime of at least 99.5% over the course of a year, ensuring that users can consistently access the platform.*
* ***REQ-3****:* ***Maintainability****: The codebase must be modular and well-documented, enabling future updates and bug fixes to be implemented within 24 hours.*
* ***REQ-4****:* ***Scalability****: The system must be able to handle a 50% increase in traffic during peak seasons (e.g., holidays) without performance degradation.*
* ***REQ-5****:* ***Testability****: Automated unit tests must cover at least 80% of the system's core functionality, ensuring that new updates do not break existing features.*

***Rationale****: These software quality attributes ensure that the system is user-friendly, reliable, and adaptable to changes. They focus on providing long-term value, stability, and the ability to scale with user demand.*

#### 5.5 Business Rules

* ***BR-1****: Only users who are logged in can make a booking. Guest users can search for flights but must create an account to complete a booking.*
* ***BR-2****: Admins are the only users who can add, edit, or remove flight records. Regular users can only view flights and make bookings.*
* ***BR-3****: A user can only have one active booking for a flight at a time. If they attempt to book the same flight again, the system will prevent duplicate bookings.*
* ***BR-4****: Payment information must be provided for booking confirmation. Users can choose from available payment options like credit card, debit card, or cash on delivery.*
* ***BR-5****: Admins can set the flight availability based on seat counts and adjust pricing. Once a booking is made, the seat count for that flight is automatically reduced.*

***Rationale****: These business rules define the operational constraints and expectations for users and admins, ensuring proper workflow and preventing misuse of the system (e.g., duplicate bookings or unauthorized data modification).*

**Appendix A: Glossary**

*Base fare*

*The price of a ticket that only includes transportation, excluding taxes and other charges*

*Direct flight*

*A flight between two places that doesn't require a change of aircraft, even if it has an intermediate stop*

*Check-in*

*The process of arriving and registering at an airport or hotel*

*IATA*

*The International Air Transport Association, a trade association for the world's airlines*

*GDS*

*A Global Distribution System, a computerized network that allows travel agencies and airlines to access real-time flight information*

*Central reservation system*

*A database that stores flight schedules, available seats, fares, rules, and passenger profiles*

*First class*

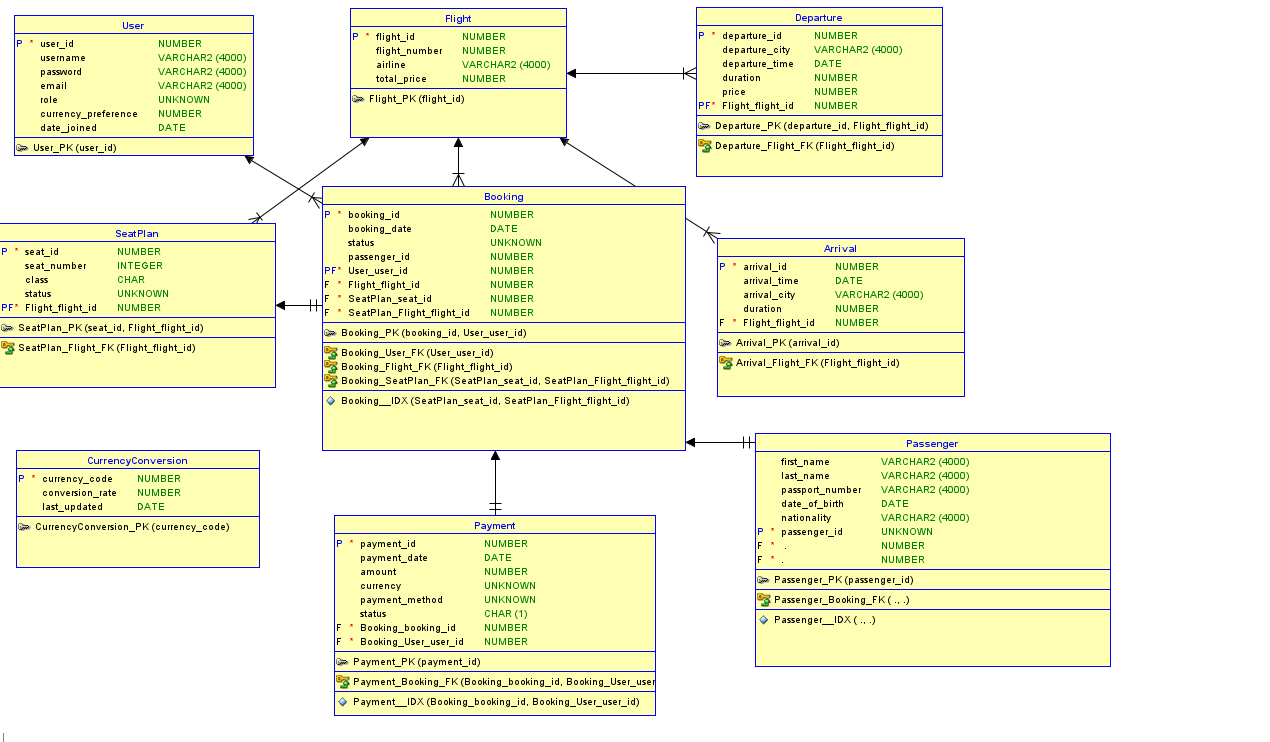
*The most luxurious cabin class on an airline, with superior amenities, more spacious seating, and enhanced services*

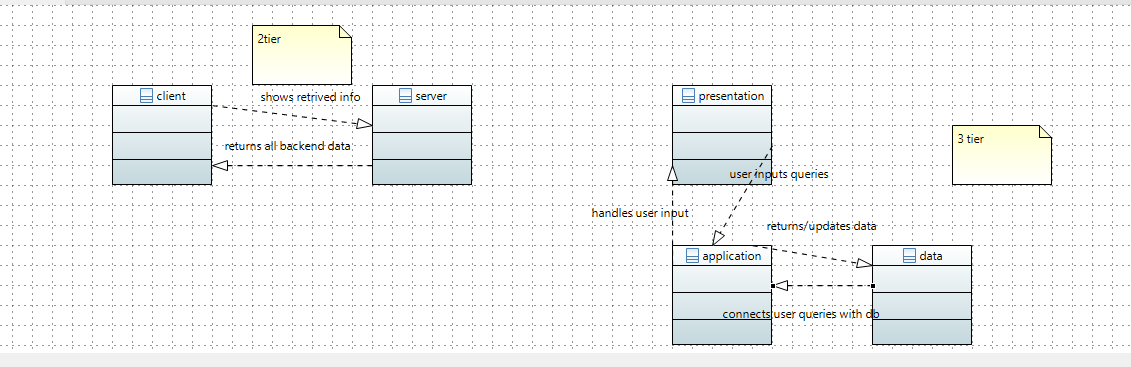
*Advance purchase*

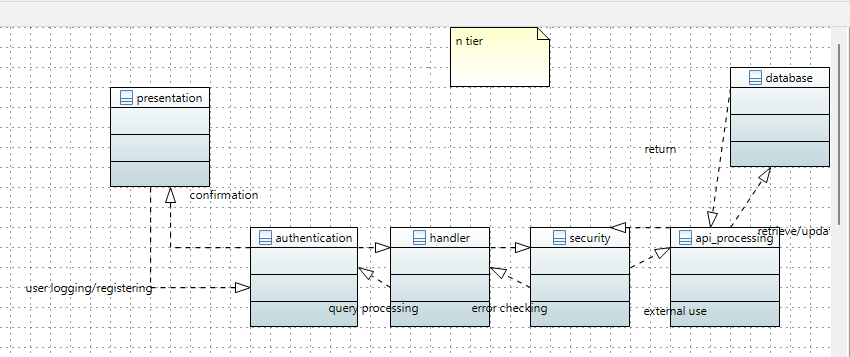
*An airfare that requires the purchase of a ticket a minimum number of days before departure*

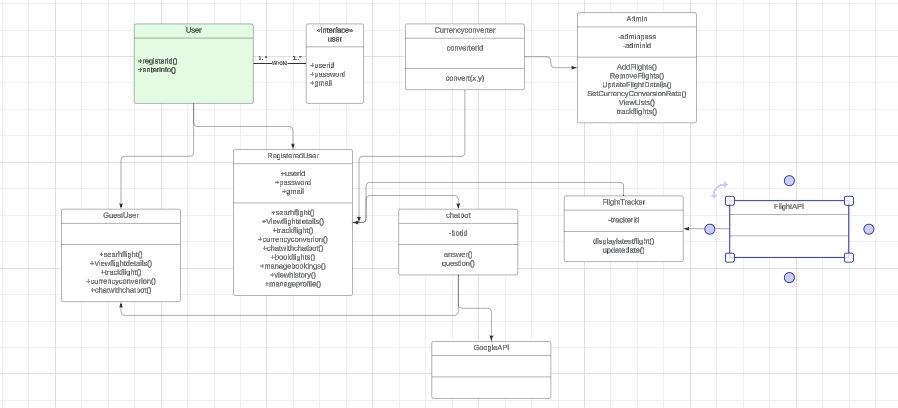
**Appendix B: Analysis Models**

ERD:

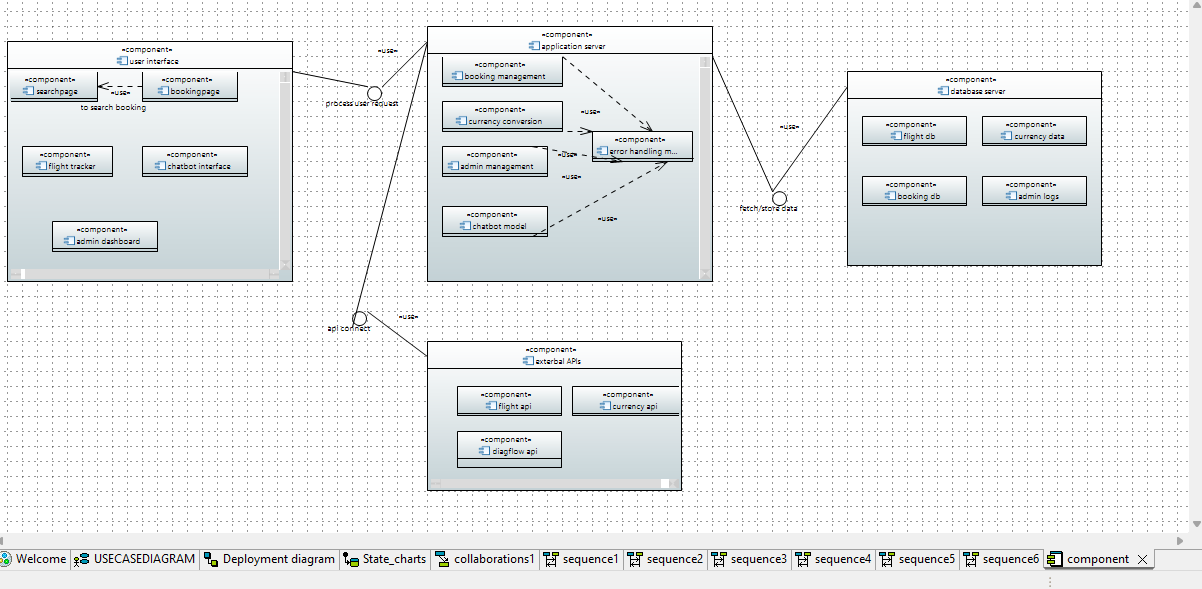
  
Architectures:



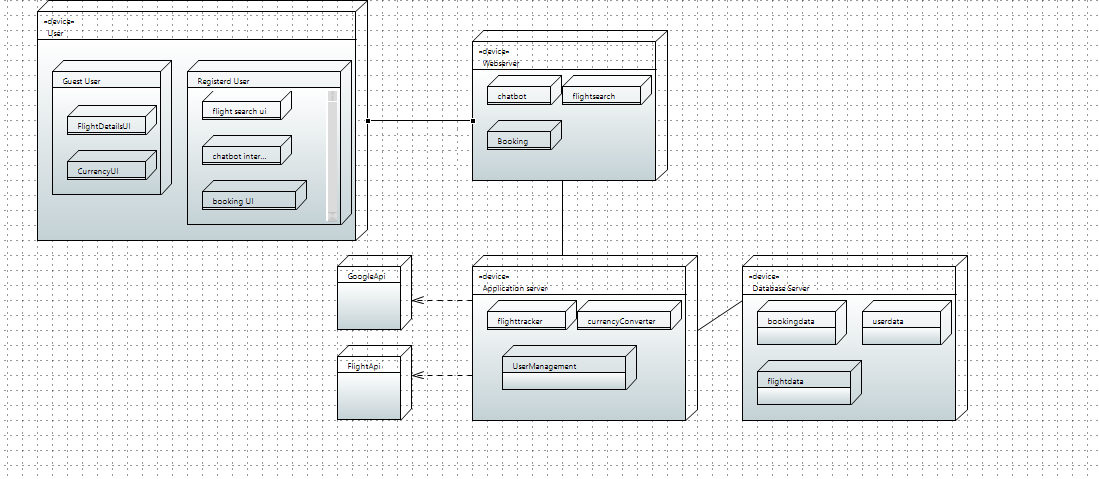


class  


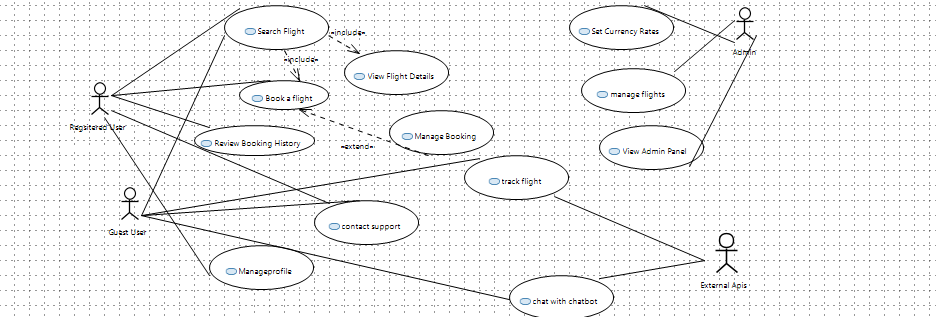
component



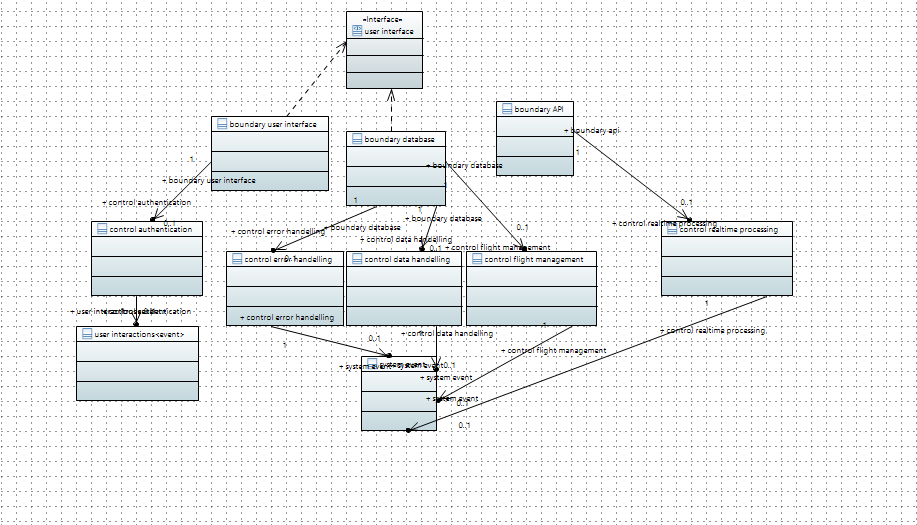
deployment



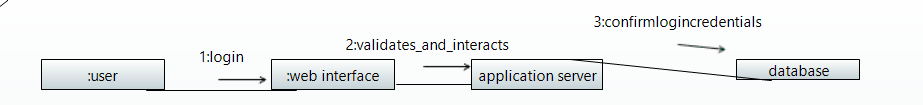
usecase

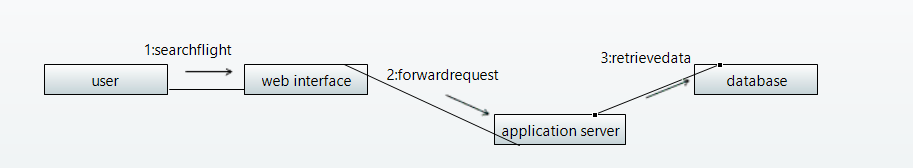


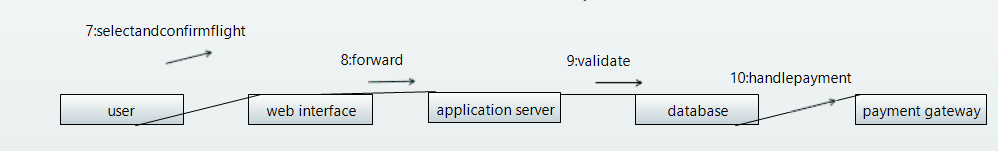
ecb:

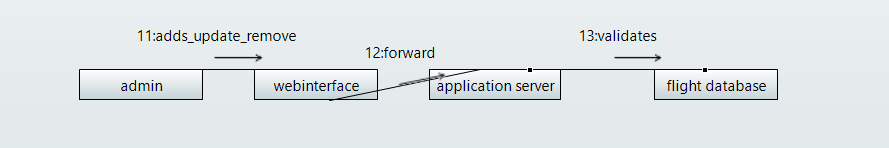
****

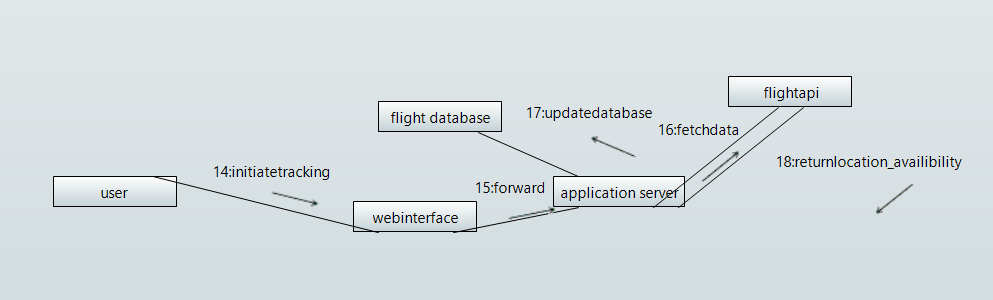
COLLABORATIONS:

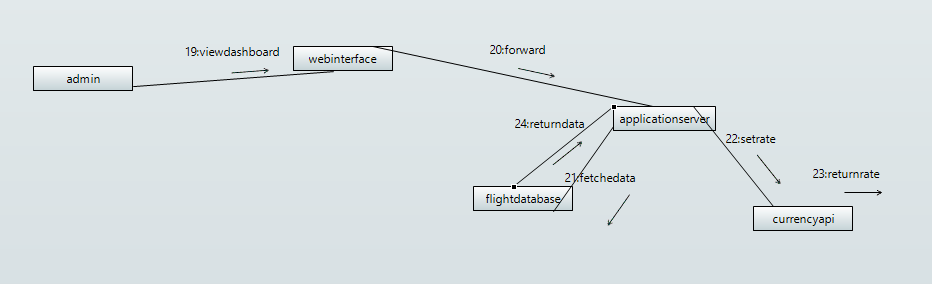


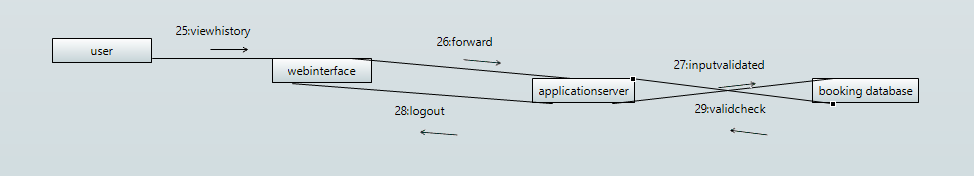


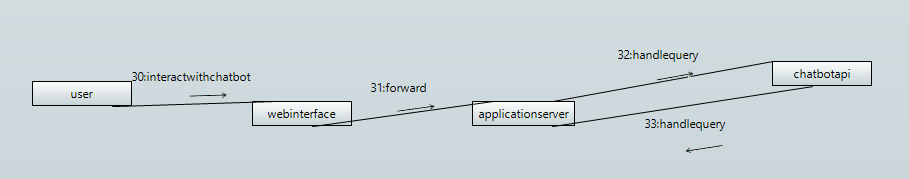


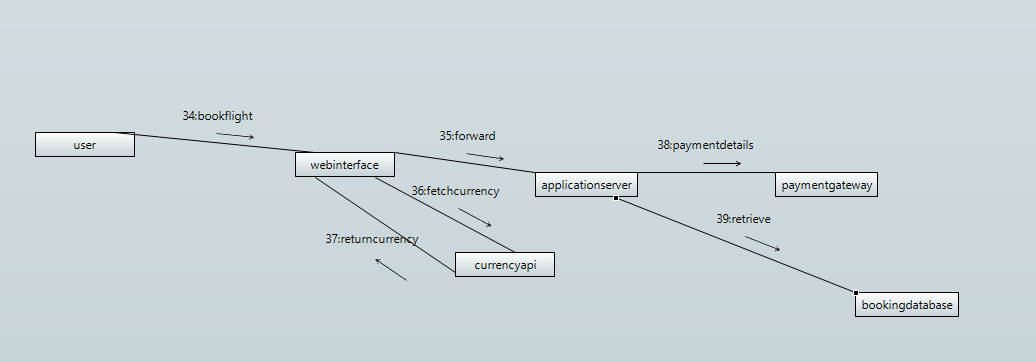


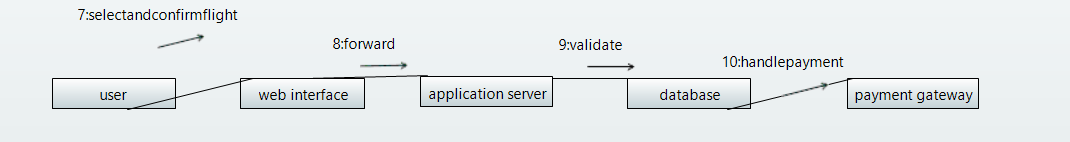




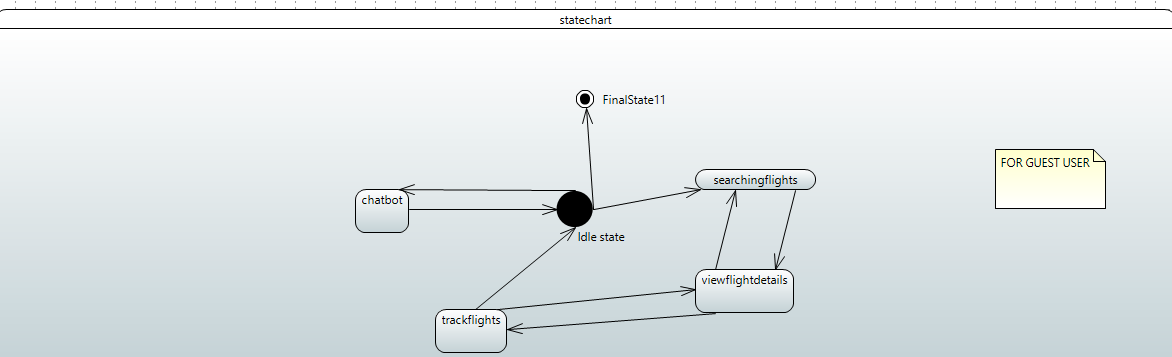


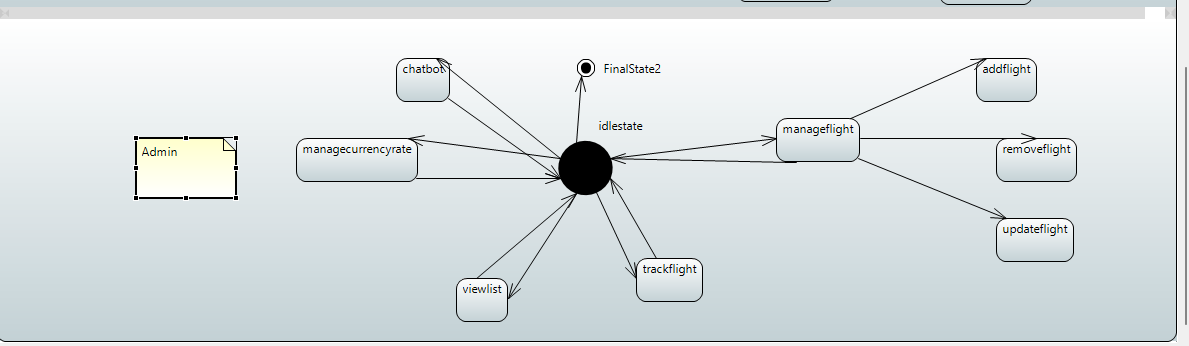


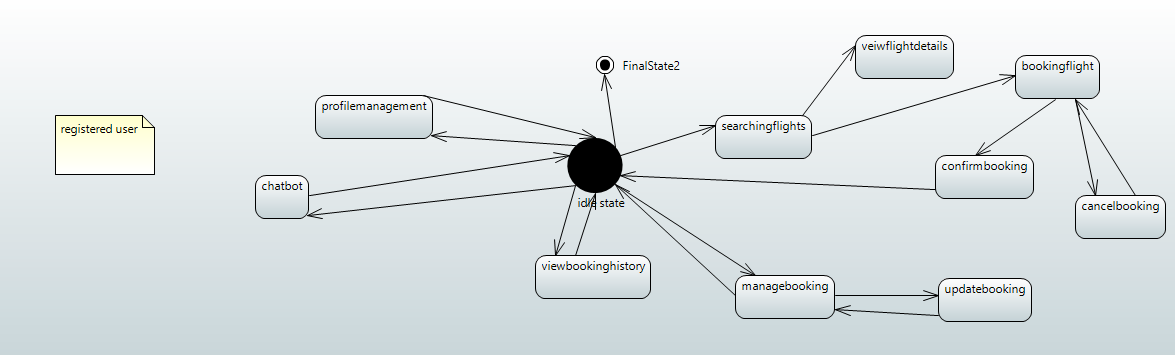




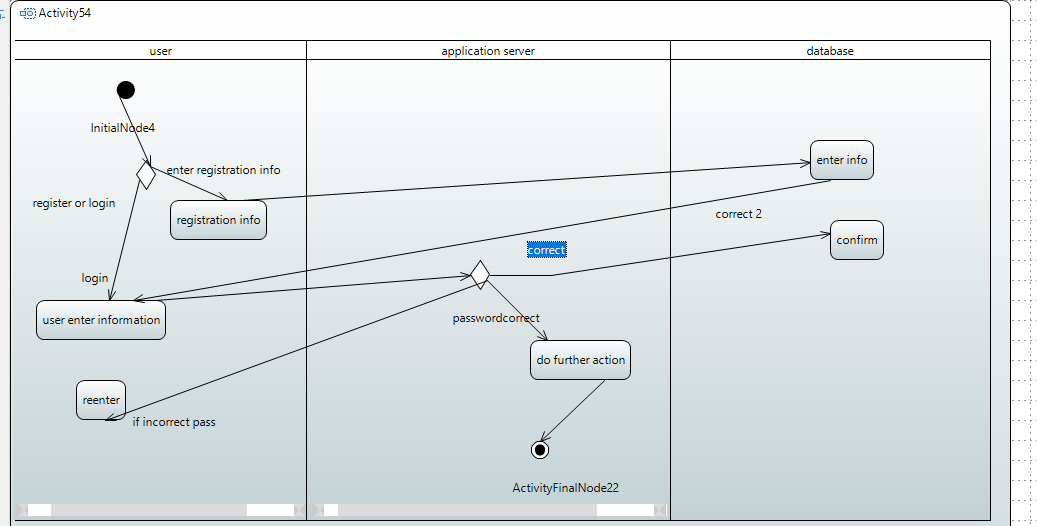
statecharts:

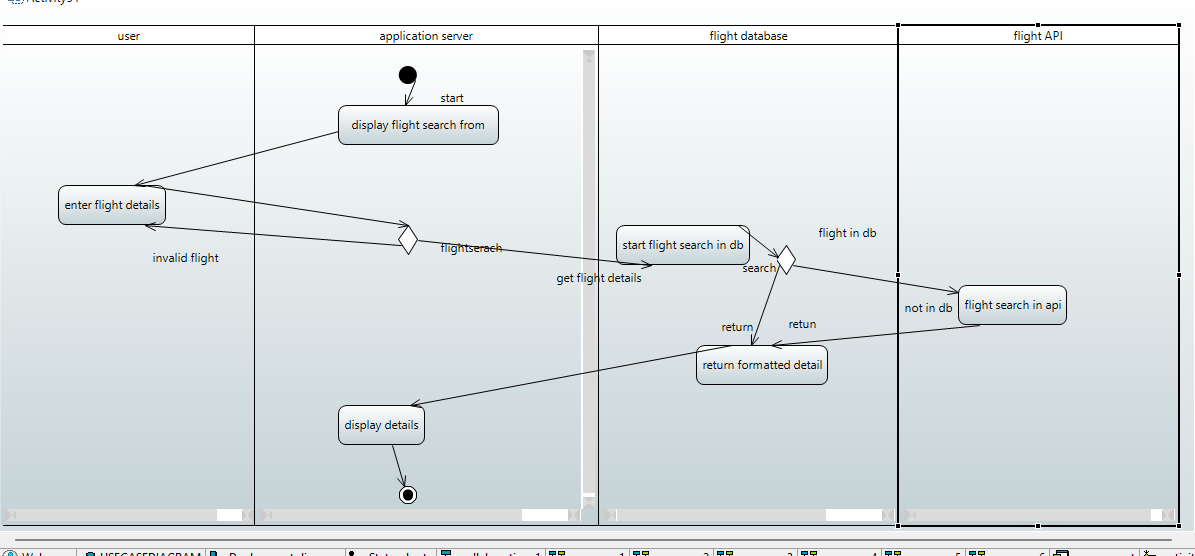


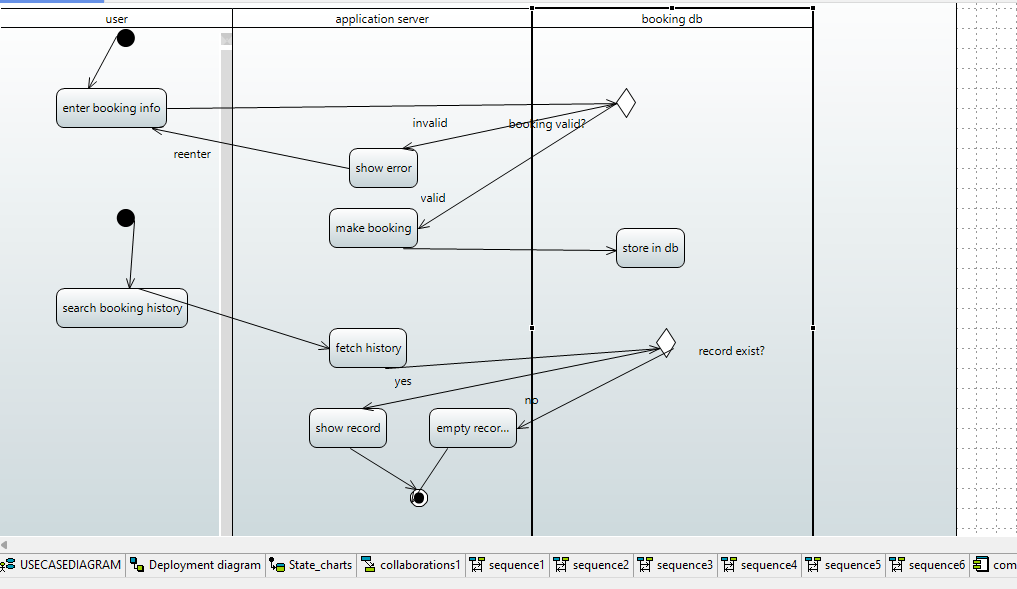


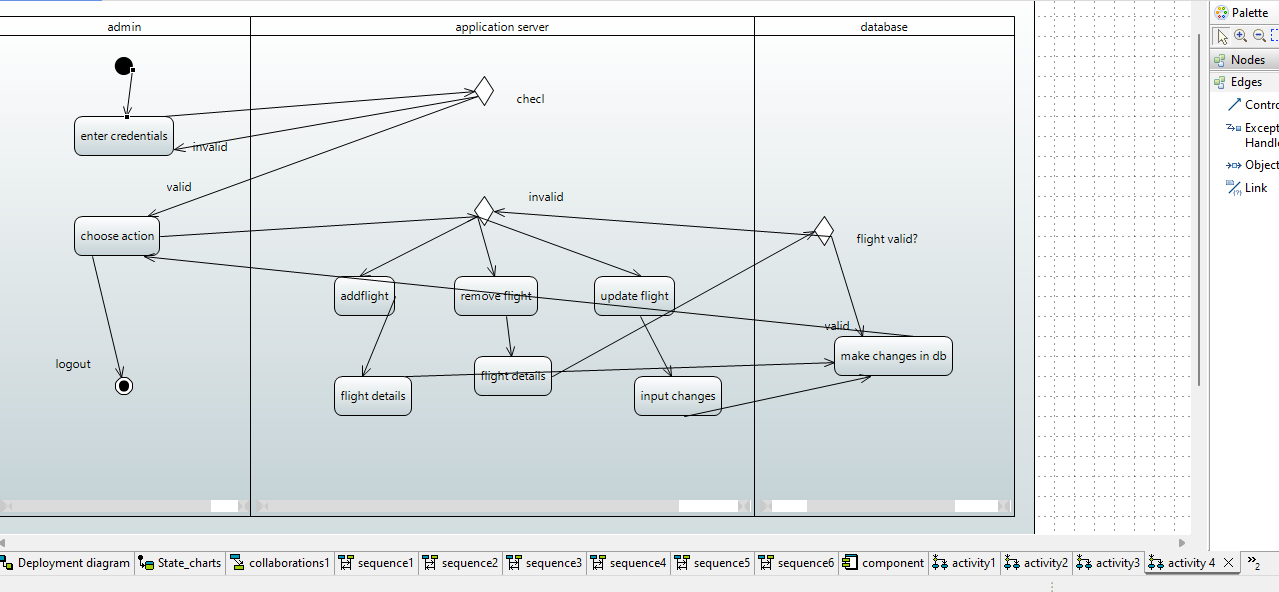


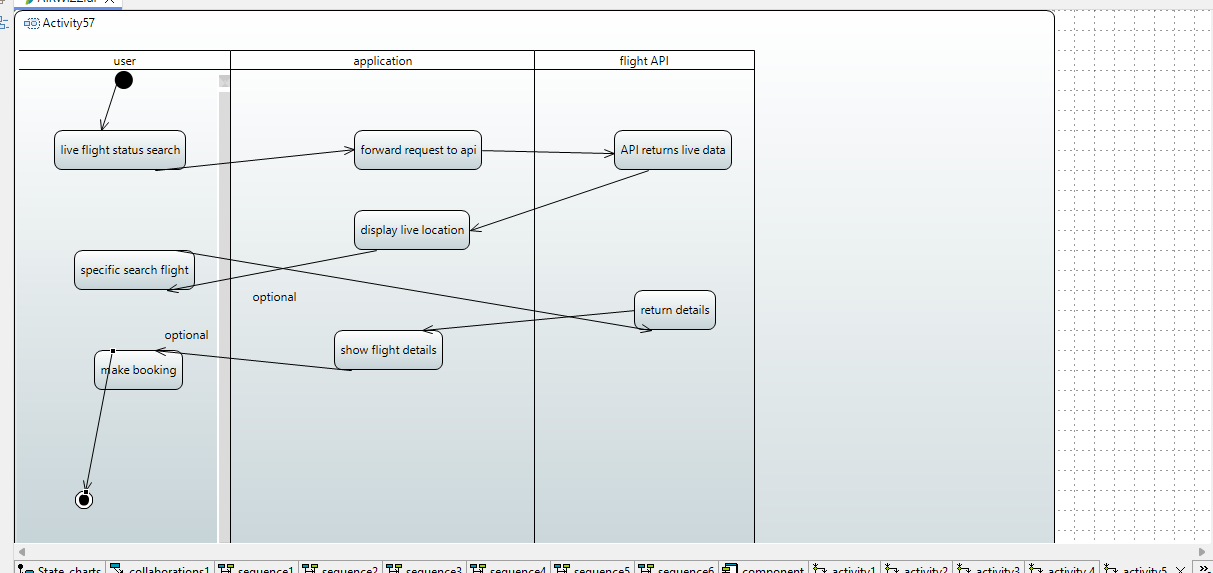
Activity

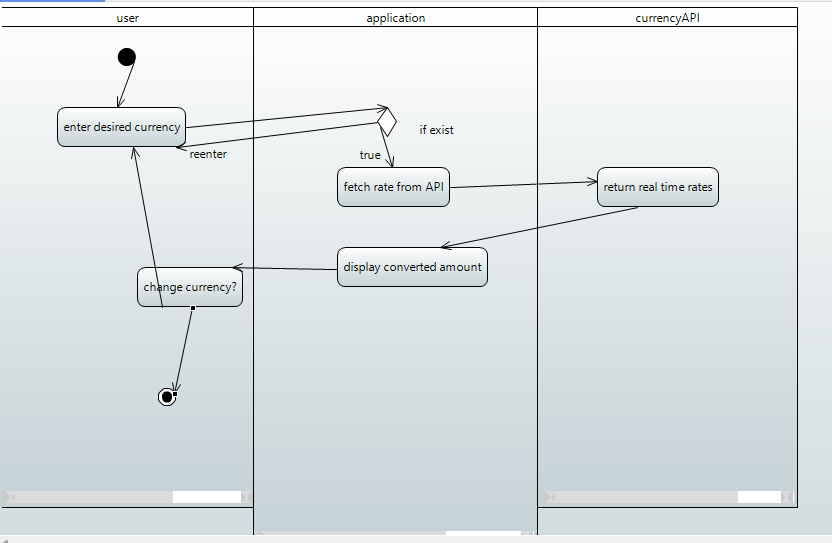


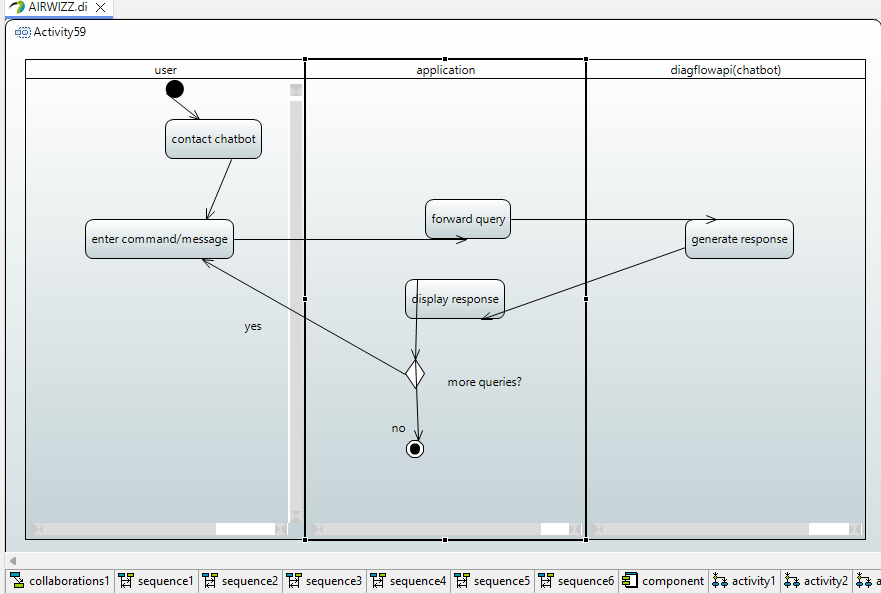


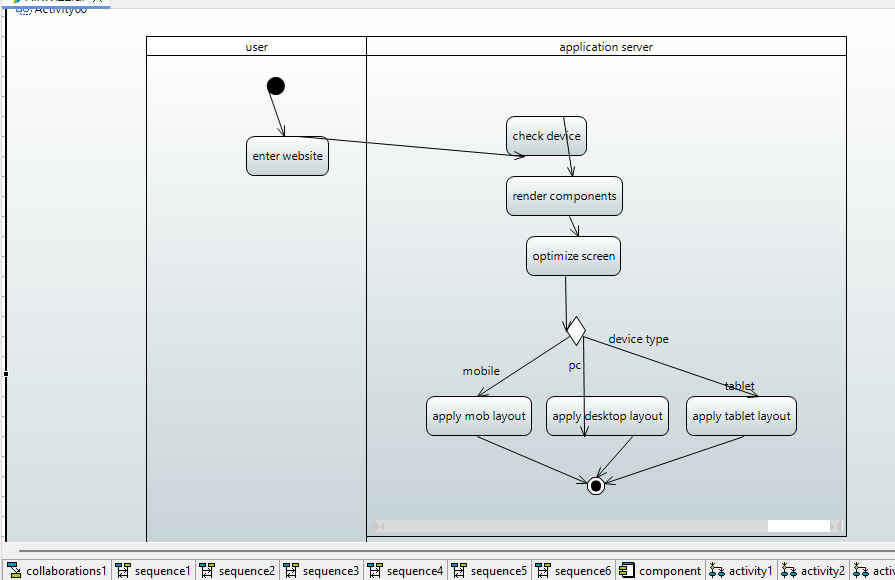


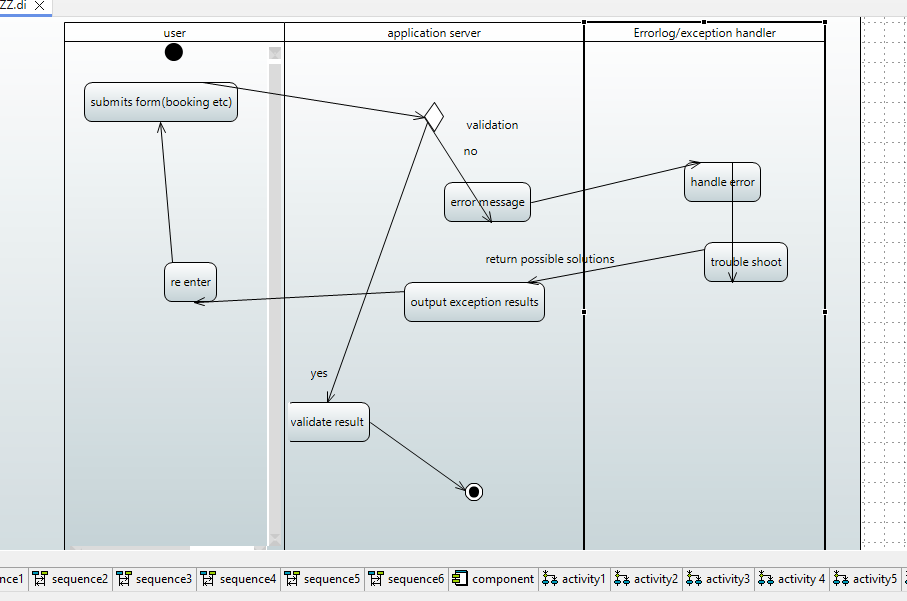












**Appendix C: To Be Determined List**

1. *further improving off flight tracking system*
2. *chatbot options*
3. *emails api*