AerisGo

A PROJECT REPORT

Submitted by

Vansh Lakhani 92300938064

Maan Trambadia 92300938072

Esha Bhadja 92300938084

Parth Joisar 92300938087

Aditya Paneri 92300938089

In partial fulfilment for the award of the degree of

DIPLOMA ENGINEERING

in

Computer Engineering

Faculty of Diploma Studies

Marwadi University, Rajkot

Marwadi University, Rajkot

Faculty of Diploma Studies

Computer Engineering Department

2025-26

CERTIFICATE

This is to certify that the project entitled **AerisGo** has been carried out by **Vansh Lakhani(92300938064)** under my guidance in partial fulfilment of the degree of Diploma Engineering in Computer Engineering (6th Semester) of Marwadi University, Rajkot during the academic year 2025-26.

Date: 24/01/2026

Internal Guide	**Head of the Department**
Prof. Rushi Raval	Prof. Smit Thacker
Assistant Professor	Computer Engineering

Marwadi University, Rajkot

Faculty of Diploma Studies

Computer Engineering Department

2025-26

CERTIFICATE

This is to certify that the project entitled **AerisGo** has been carried out by **Maan Trambadia(92300938072)** under my guidance in partial fulfilment of the degree of Diploma Engineering in Computer Engineering (6th Semester) of Marwadi University, Rajkot during the academic year 2025-26.

Date: 24/01/2026

Internal Guide	**Head of the Department**
Prof. Rushi Raval	Prof. Smit Thacker
Assistant Professor	Computer Engineering

Marwadi University, Rajkot

Faculty of Diploma Studies

Computer Engineering Department

2025-26

CERTIFICATE

This is to certify that the project entitled **AerisGo** has been carried out by **Esha Bhadja(92300938084)** under my guidance in partial fulfilment of the degree of Diploma Engineering in Computer Engineering (6th Semester) of Marwadi University, Rajkot during the academic year 2025-26.

Date: 24/01/2026

Internal Guide	**Head of the Department**
Prof. Rushi Raval	Prof. Smit Thacker
Assistant Professor	Computer Engineering

Marwadi University, Rajkot

Faculty of Diploma Studies

Computer Engineering Department

2025-26

CERTIFICATE

This is to certify that the project entitled **AerisGo** has been carried out by **Parth Joisar(92300938087)** under my guidance in partial fulfilment of the degree of Diploma Engineering in Computer Engineering (6th Semester) of Marwadi University, Rajkot during the academic year 2025-26.

Date: 24/01/2026

Internal Guide	**Head of the Department**
Prof. Rushi Raval	Prof. Smit Thacker
Assistant Professor	Computer Engineering

Marwadi University, Rajkot

Faculty of Diploma Studies

Computer Engineering Department

2025-26

CERTIFICATE

This is to certify that the project entitled **AerisGo** has been carried out by **Aditya Paneri(92300938089)** under my guidance in partial fulfilment of the degree of Diploma Engineering in Computer Engineering (6th Semester) of Marwadi University, Rajkot during the academic year 2025-26.

Date: 24/01/2026

Internal Guide	**Head of the Department**
Prof. Rushi Raval	Prof. Smit Thacker
Assistant Professor	Computer Engineering

Contents

Acknowledgement

We are sincerely grateful for the unwavering support and continuous guidance we received throughout the development journey of our project, **"AerisGo."** This project would not have reached its present form without the involvement and encouragement of several individuals who played a vital role at various stages of its execution.

We would like to extend our deepest and most heartfelt gratitude to our esteemed project guide, **Prof. Rushi Raval**, for being a constant source of inspiration and mentorship. From the very beginning of the project, Prof. Raval provided us with valuable suggestions, expert insights, and timely feedback that helped us shape our raw ideas into a structured and practical system. Their vast knowledge and experience in the domain of software development were instrumental in helping us navigate through complex challenges and technical obstacles.

Prof. Raval not only guided us technically but also motivated us to think critically and work collaboratively. Their feedback was always constructive and thoughtful, enabling us to refine our approach and improve our understanding of both theoretical and practical aspects of the subject matter. The time and effort they invested in reviewing our work, pointing out areas for improvement, and encouraging us to strive for excellence had a significant impact on the overall quality and success of this project.

We are truly thankful for their patience, availability, and dedication throughout this process. It is with their support and mentorship that we were able to complete this project with confidence and gain a deeper appreciation for real-world software development practices.

Abstract

Our project is titled "AerisGo – Airline Booking Platform with Passenger and Operations Interfaces". It is a complete airline management system that now includes three main parts: a mobile app for passengers, a web app for passengers, and a web dashboard for airline staff (operations team). The passenger mobile app is built with React Native and Expo and supports both Android and iOS devices. The passenger web app and the staff dashboard are built with React.js. All of these share the same backend, which is developed using Node.js, Express.js, and MongoDB. This shared backend makes the system easier to maintain and keep consistent.

On the passenger side, users can sign up and sign in, search for flights, view flight details, choose seats, book tickets, and view their bookings. They can also manage their profile, add travel documents, check in for flights, see their boarding pass, and view information about baggage and meals. A reward system lets passengers earn and use points when they travel. Both the mobile app and the web app are designed with a clean, modern interface so that the booking experience feels smooth and easy to use.

On the operations side, airline staff and admins use a web-based dashboard. From there they can manage flights, aircraft, crew, and seat maps, view and control bookings, update or cancel flights, and mark delays or gate changes in near real time. The dashboard also provides basic analytics such as flight performance, load factor, and route performance to support better decisions. Because all platforms use the same backend APIs, data stays in sync across the system, reduces duplication, and helps streamline airline workflows while giving passengers a reliable booking and check-in experience.

List of Table

List of Figures

1. Use case Diagram

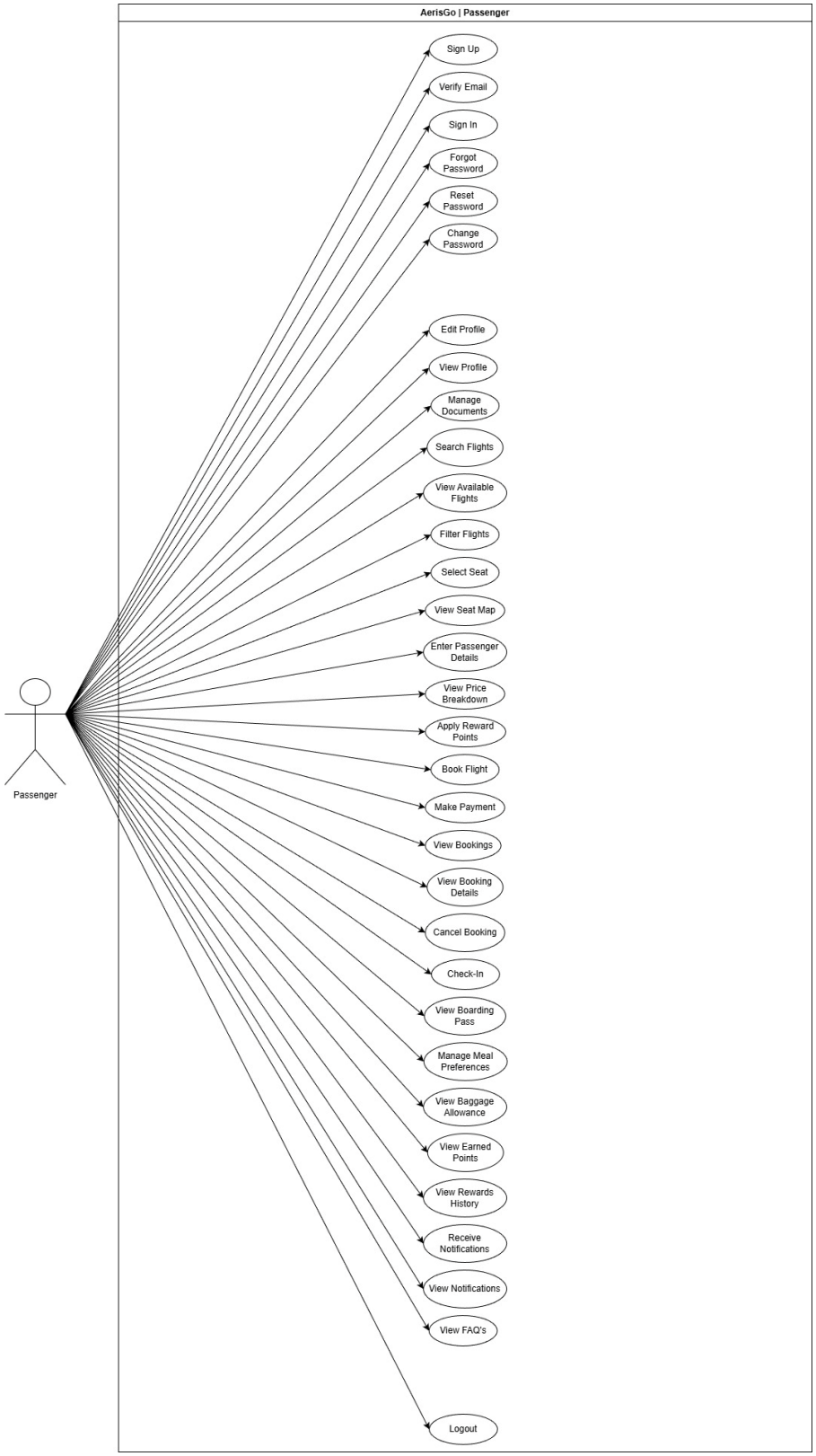


Figure 1.1.1 Passenger Use Case Diagram

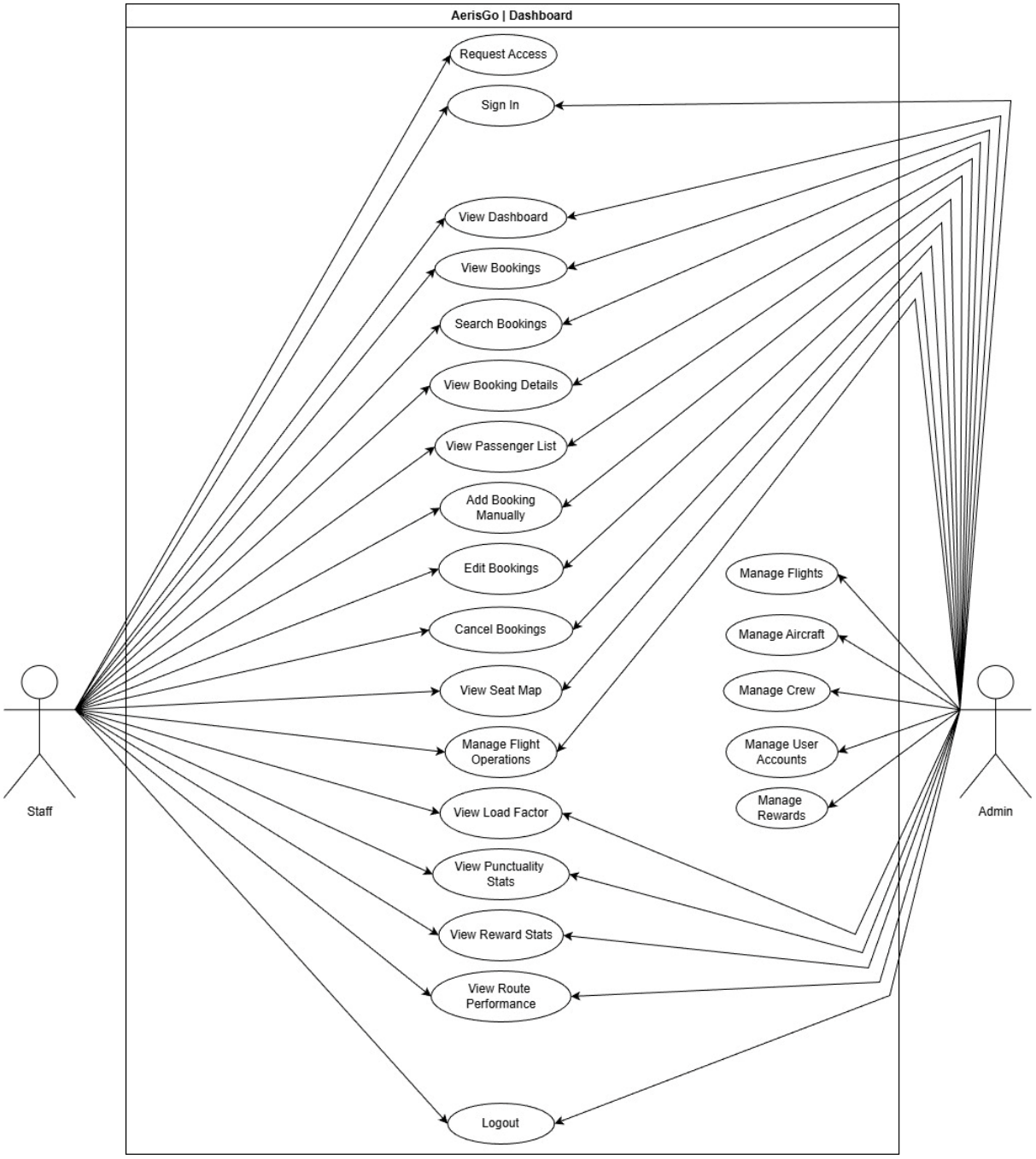


Figure 1.1.2 Dashboard Use Case Diagram

2. E.R Diagram

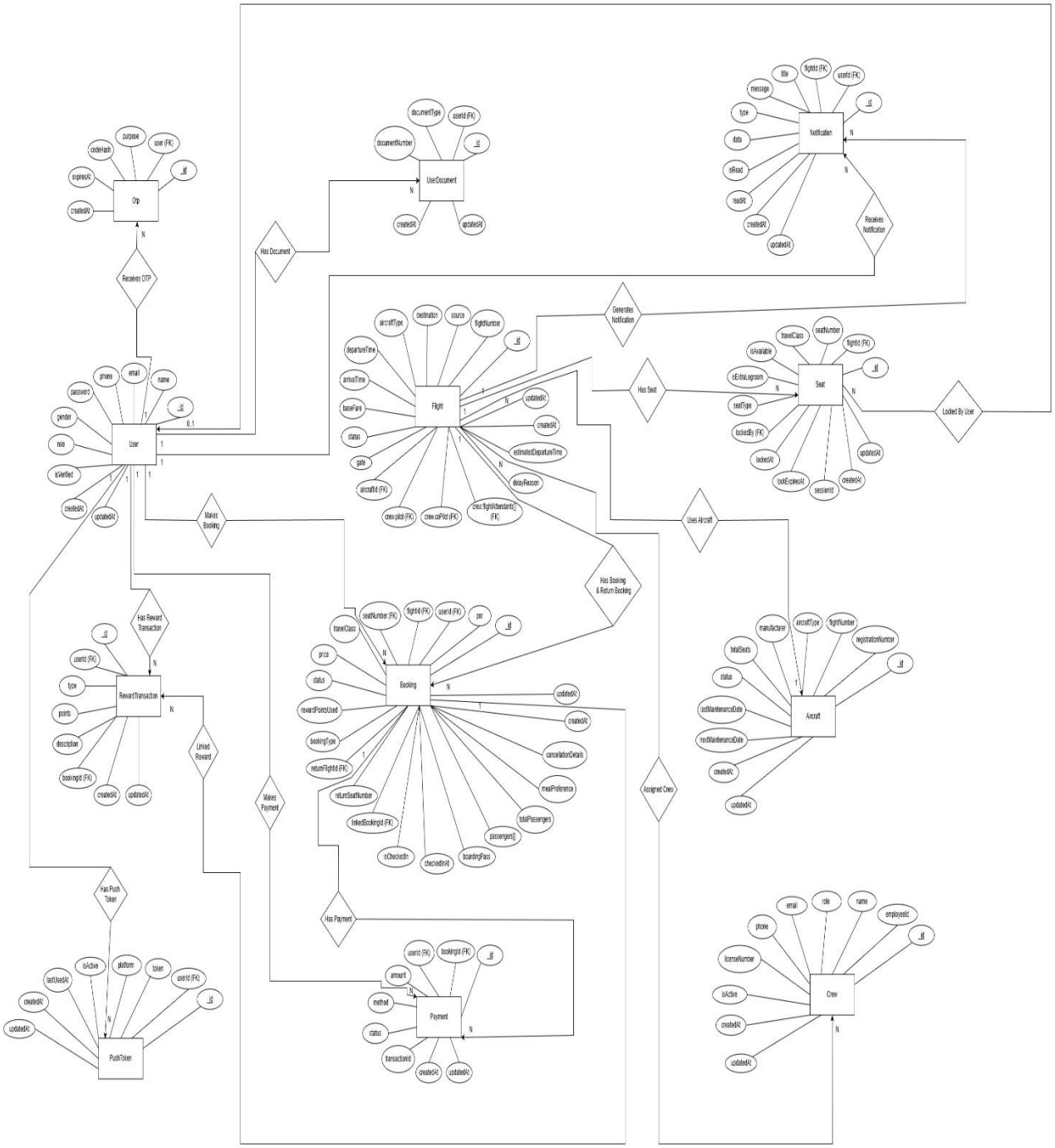


Figure 1.2 ER Diagram

3. Activity Diagram

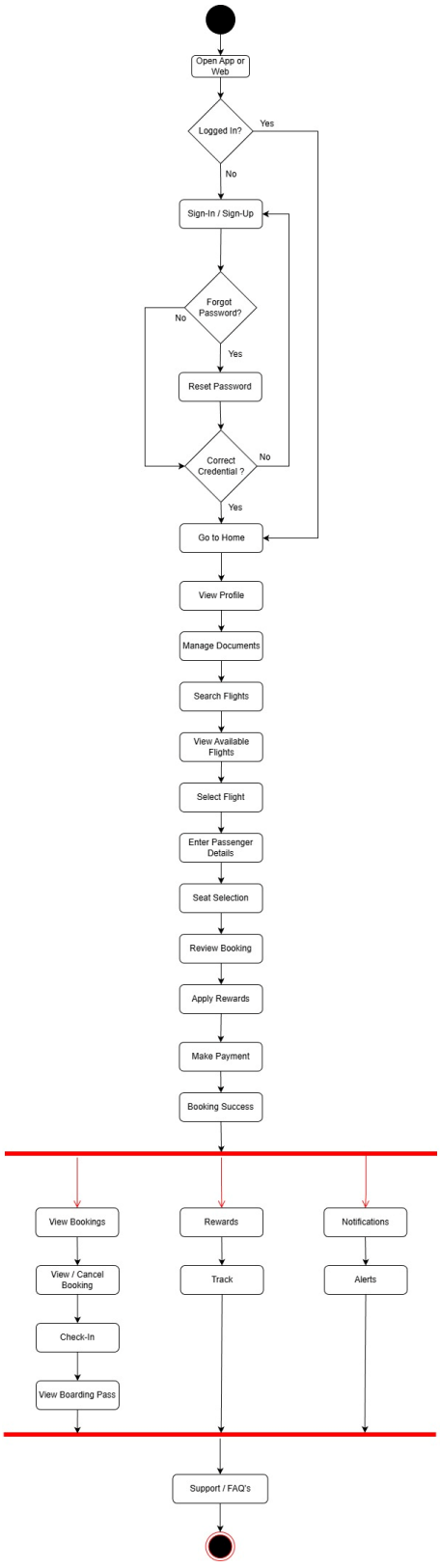


Figure 1.3.1 Passenger Activity Diagram

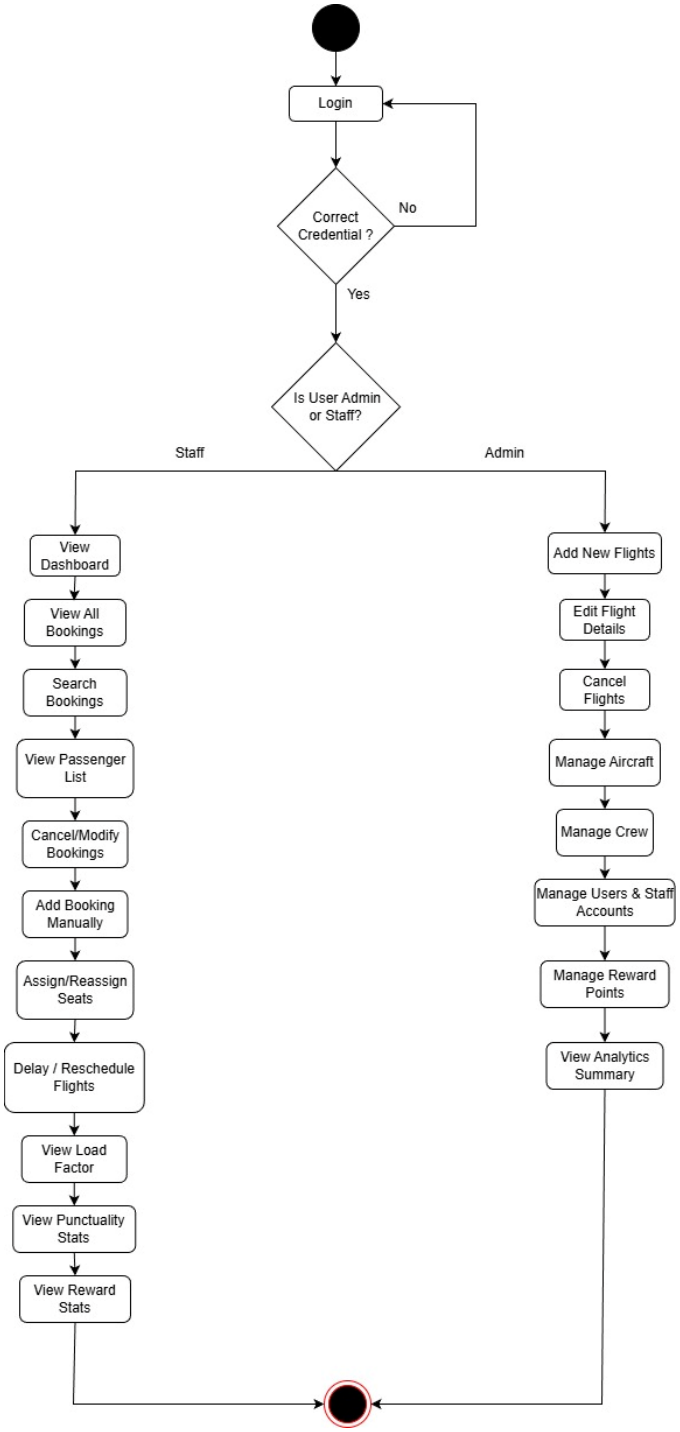
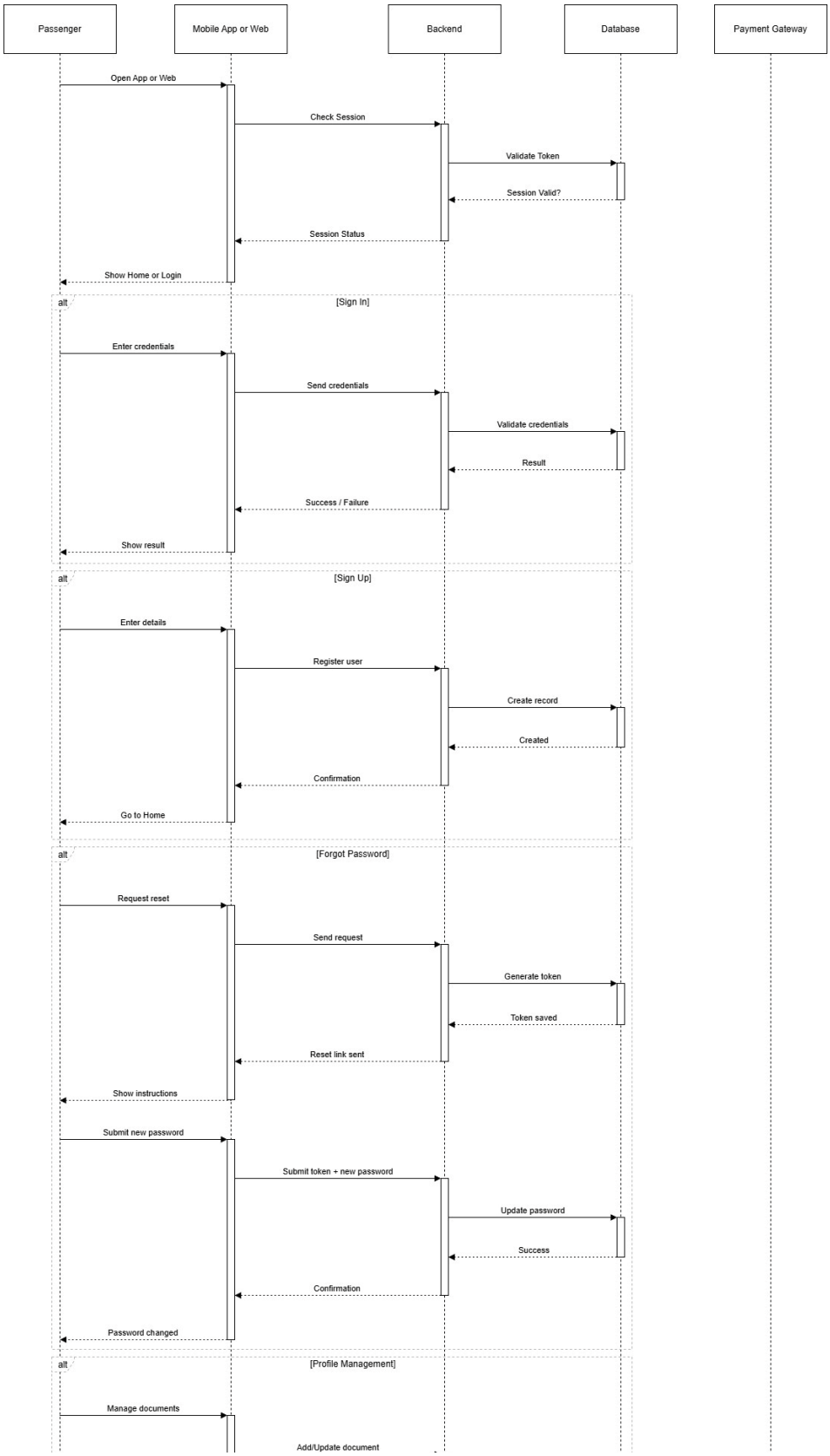
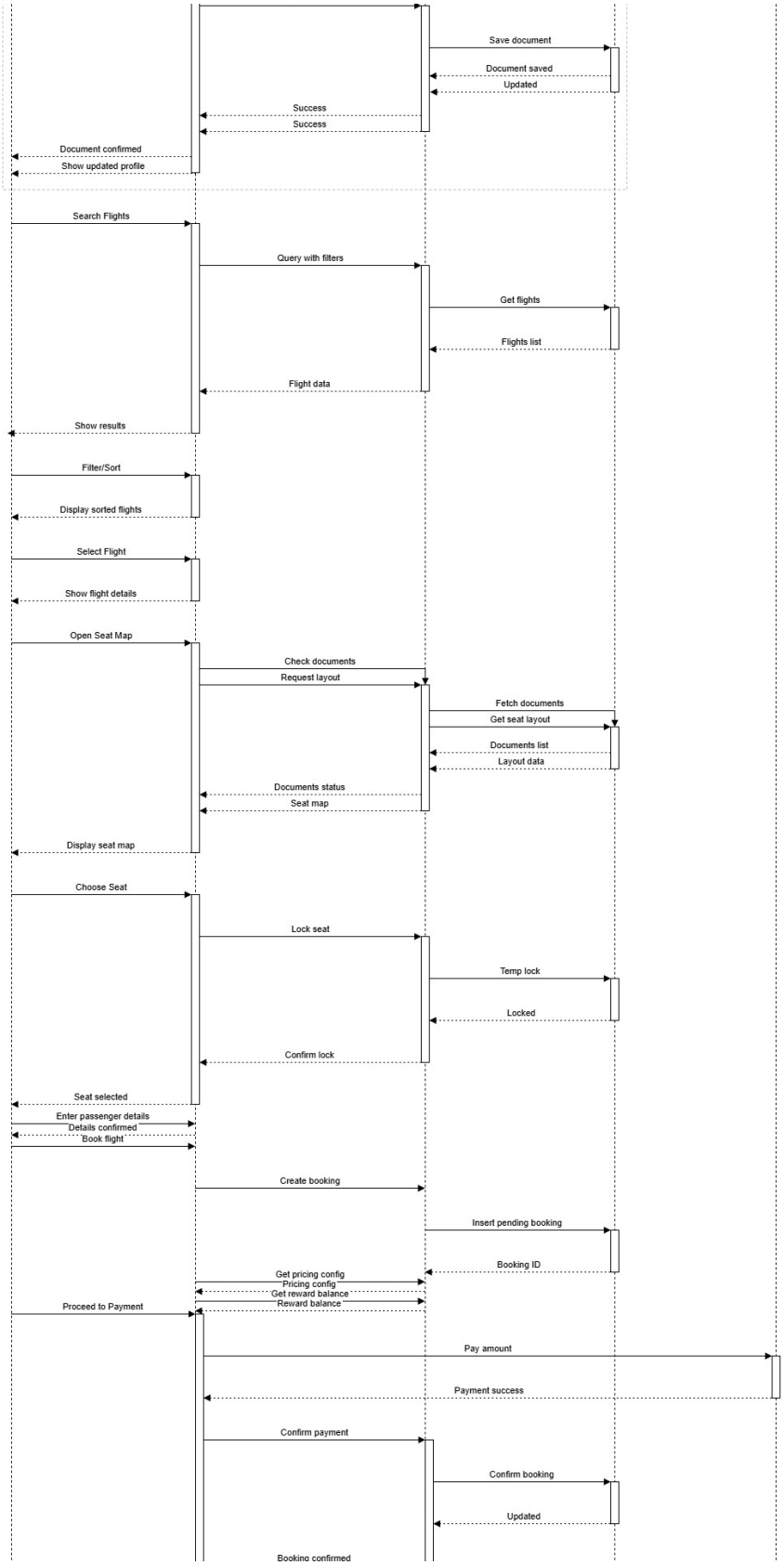


Figure 1.3.2 Dashboard Activity Diagram

4. Sequence Diagram




Save document
Document saved
Updated
Success
Success
Document confirmed
Show updated profile
Search Flights
Query with filters
Get flights
Flights list
Flight data
Show results
Filter/Sort
Display sorted flights
Select Flight
Show flight details
Open Seat Map
Check documents
Request layout
Fetch documents
Get seat layout
Documents list
Layout data
Documents status
Seat map
Display seat map
Choose Seat
Lock seat
Temp lock
Locked
Confirm lock
Seat selected
Enter passenger details
Details confirmed
Book flight
Create booking
Insert pending booking
Booking ID
Get pricing config
Pricing config
Get reward balance
Reward balance
Proceed to Payment
Pay amount
Payment success
Confirm payment
Confirm booking
Updated
Booking confirmed

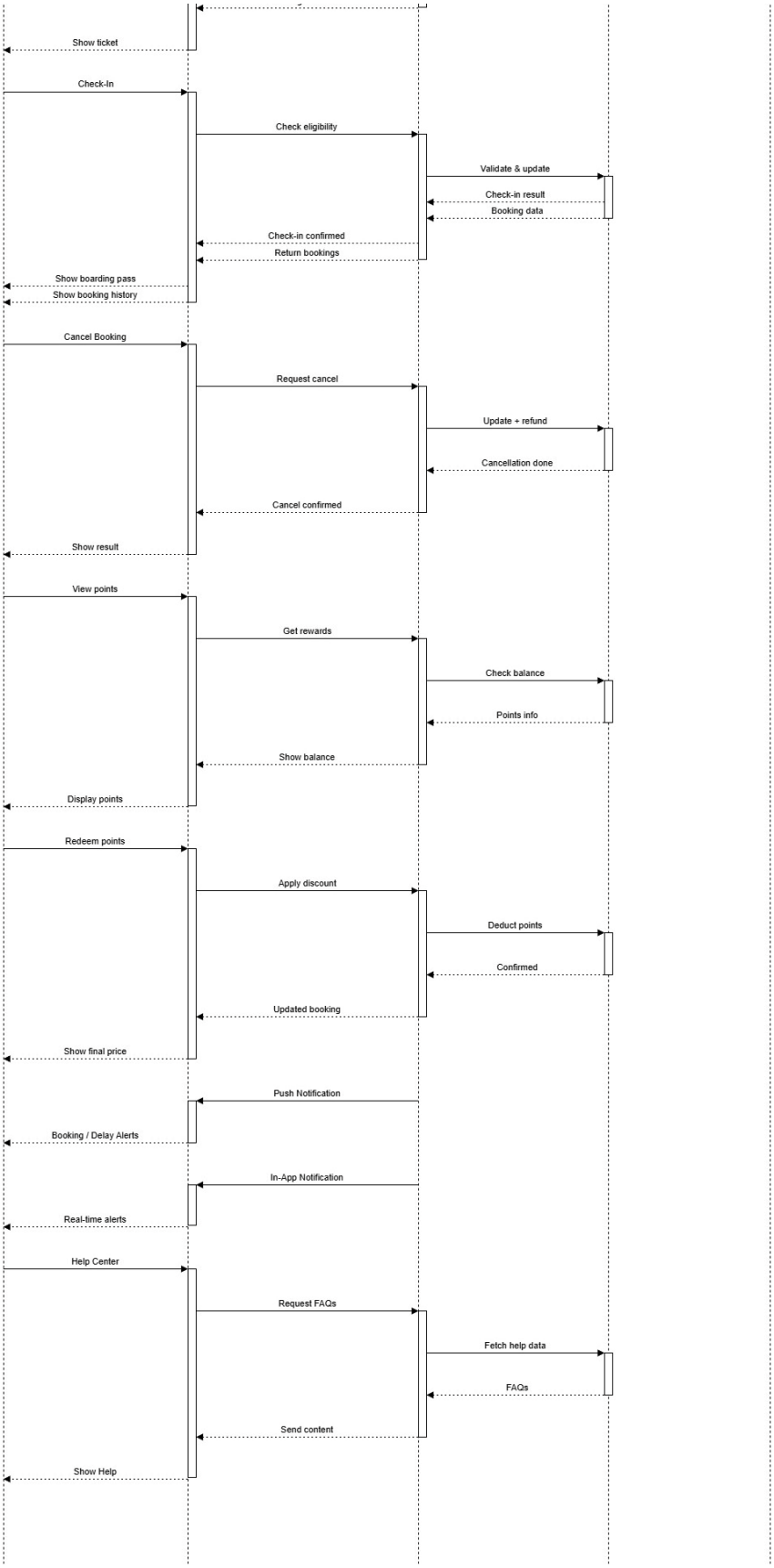
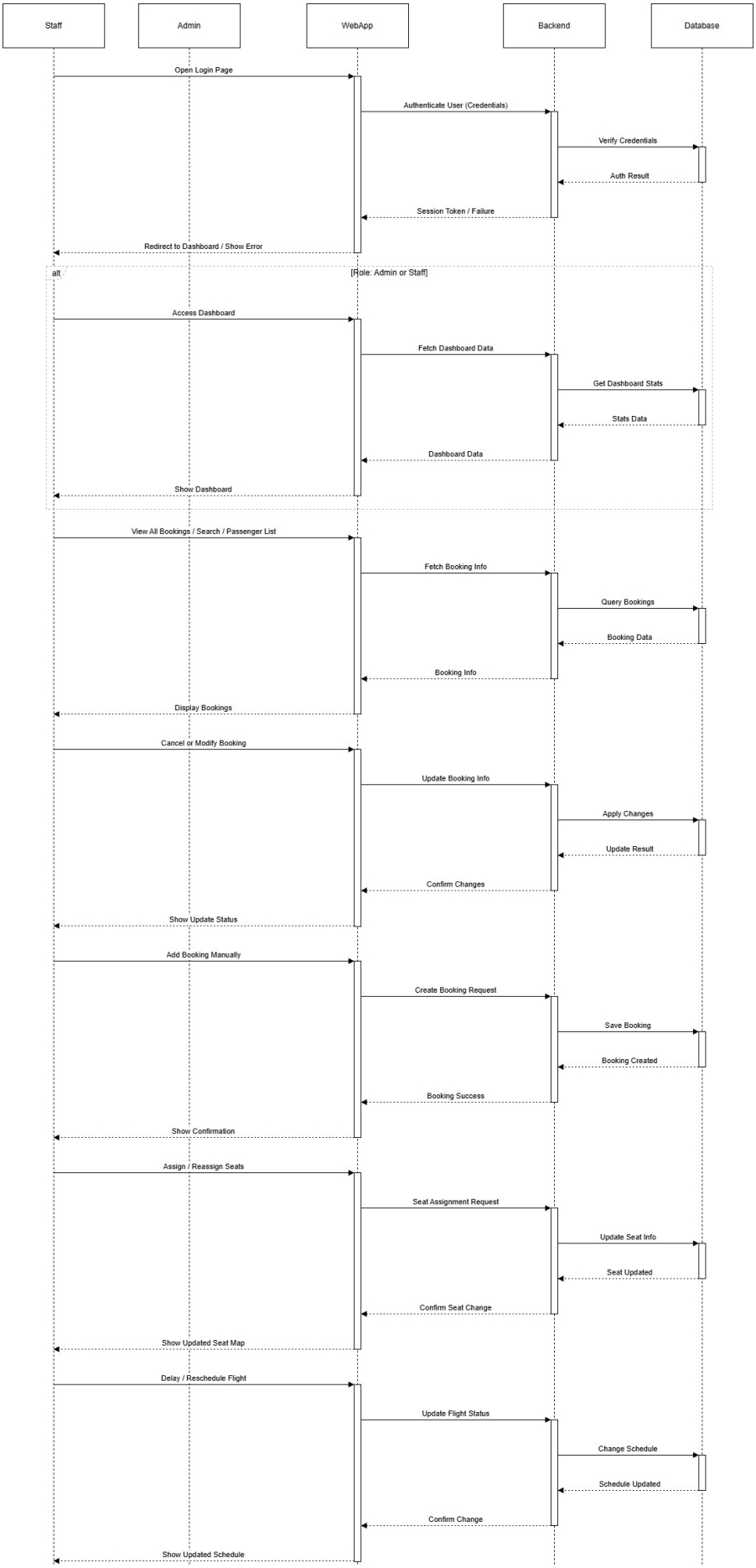


Figure 1.4.1 Passenger Sequence Diagram


Staff
Admin
WebApp
Backend
Database
Open Login Page
Authenticate User (Credentials)
Verify Credentials
Auth Result
Session Token / Failure
Redirect to Dashboard / Show Error
alt
[Role: Admin or Staff]
Access Dashboard
Fetch Dashboard Data
Get Dashboard Stats
Stats Data
Dashboard Data
Show Dashboard
View All Bookings / Search / Passenger List
Fetch Booking Info
Query Bookings
Booking Data
Booking Info
Display Bookings
Cancel or Modify Booking
Update Booking Info
Apply Changes
Update Result
Confirm Changes
Show Update Status
Add Booking Manually
Create Booking Request
Save Booking
Booking Created
Booking Success
Show Confirmation
Assign / Reassign Seats
Seat Assignment Request
Update Seat Info
Seat Updated
Confirm Seat Change
Show Updated Seat Map
Delay / Reschedule Flight
Update Flight Status
Change Schedule
Schedule Updated
Confirm Change
Show Updated Schedule

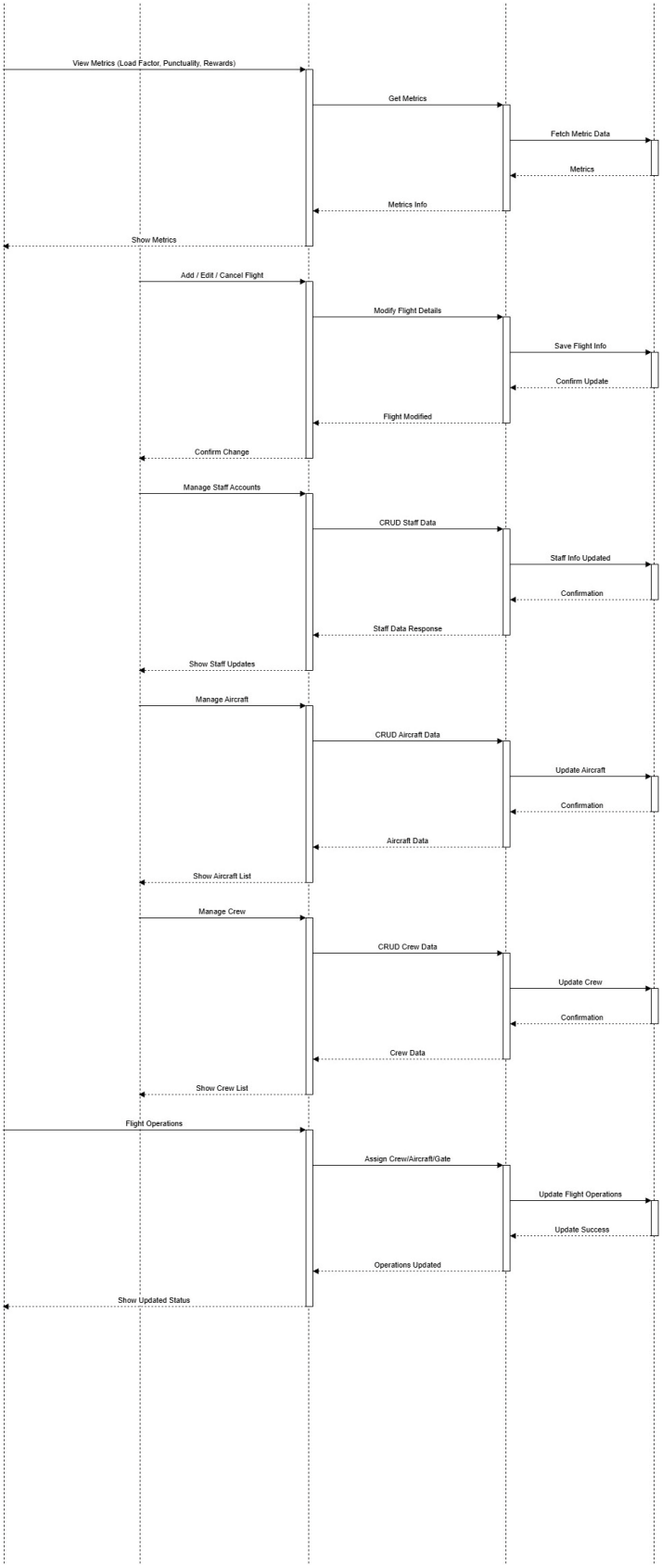


Figure 1.4.2 Dashboard Sequence Diagram

5. DFD Diagram(Level 0)

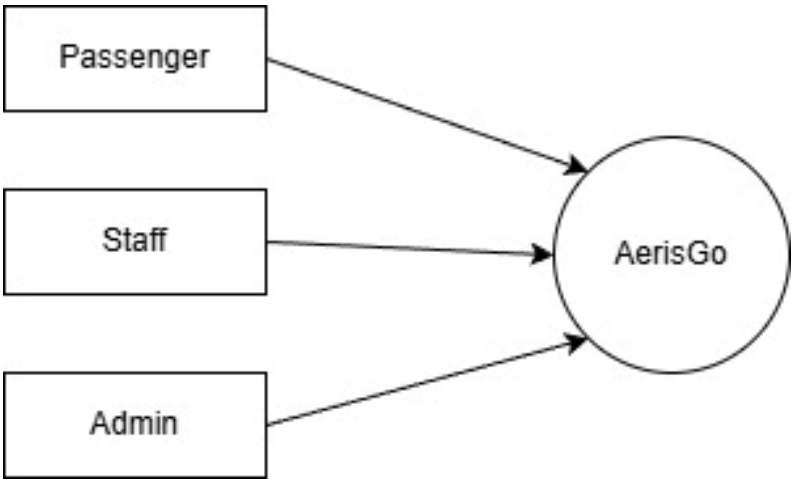


Figure 1.5 DFD Diagram(Level 0)

6. DFD Diagram(Level 1)

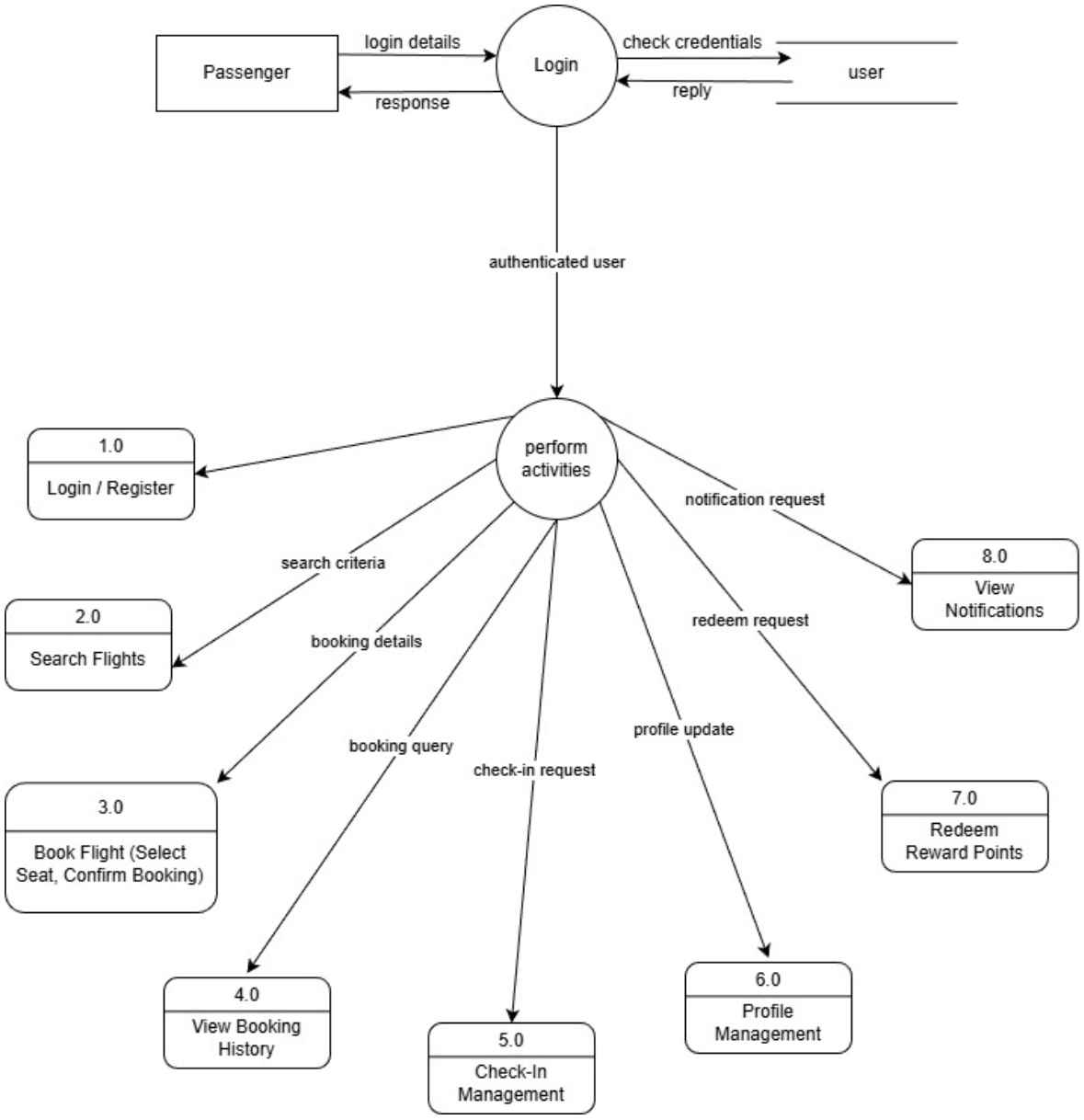


Figure 1.6.1 Passenger DFD Diagram(Level 1)

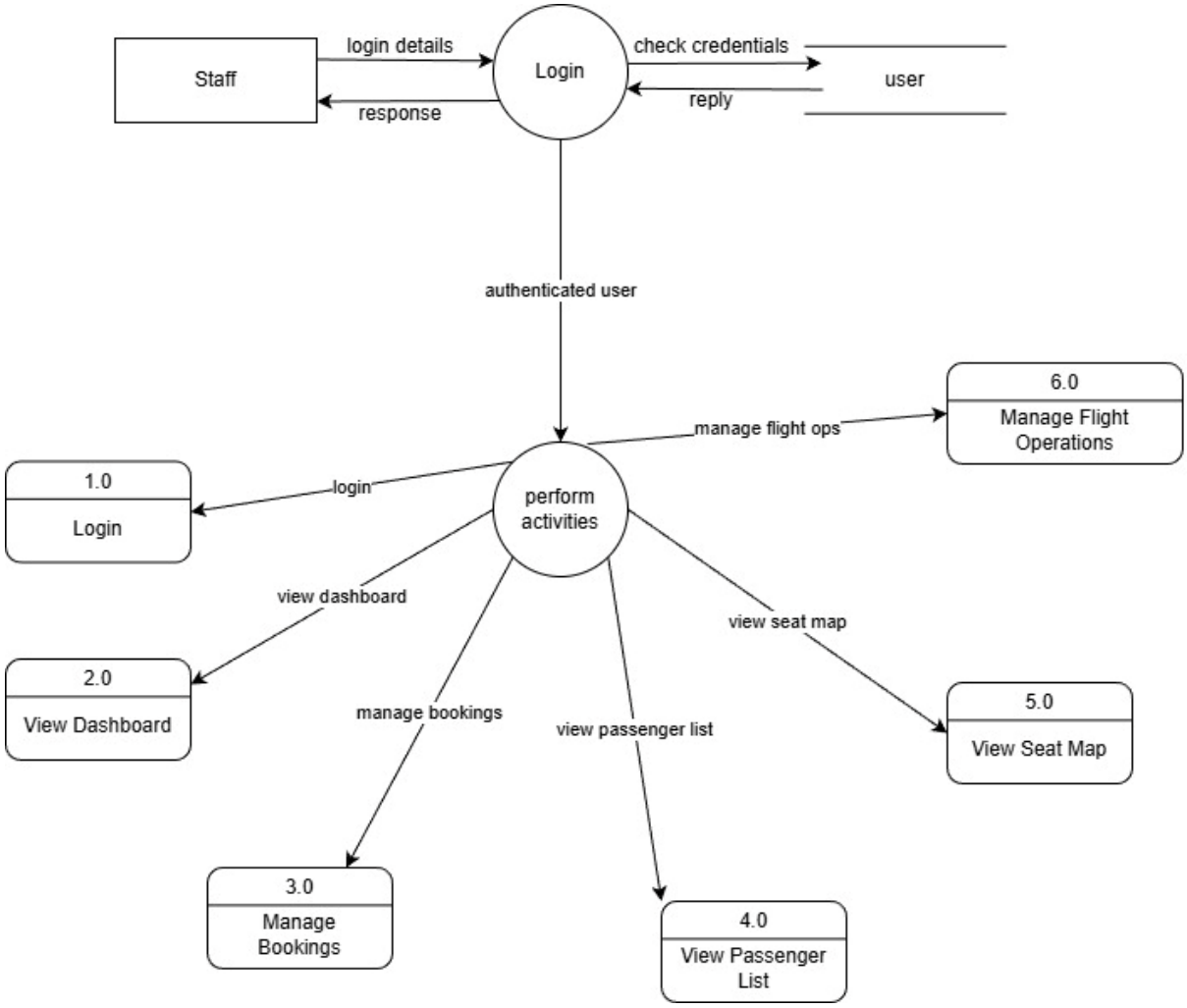


Figure 1.6.2 Staff DFD Diagram(Level 1)

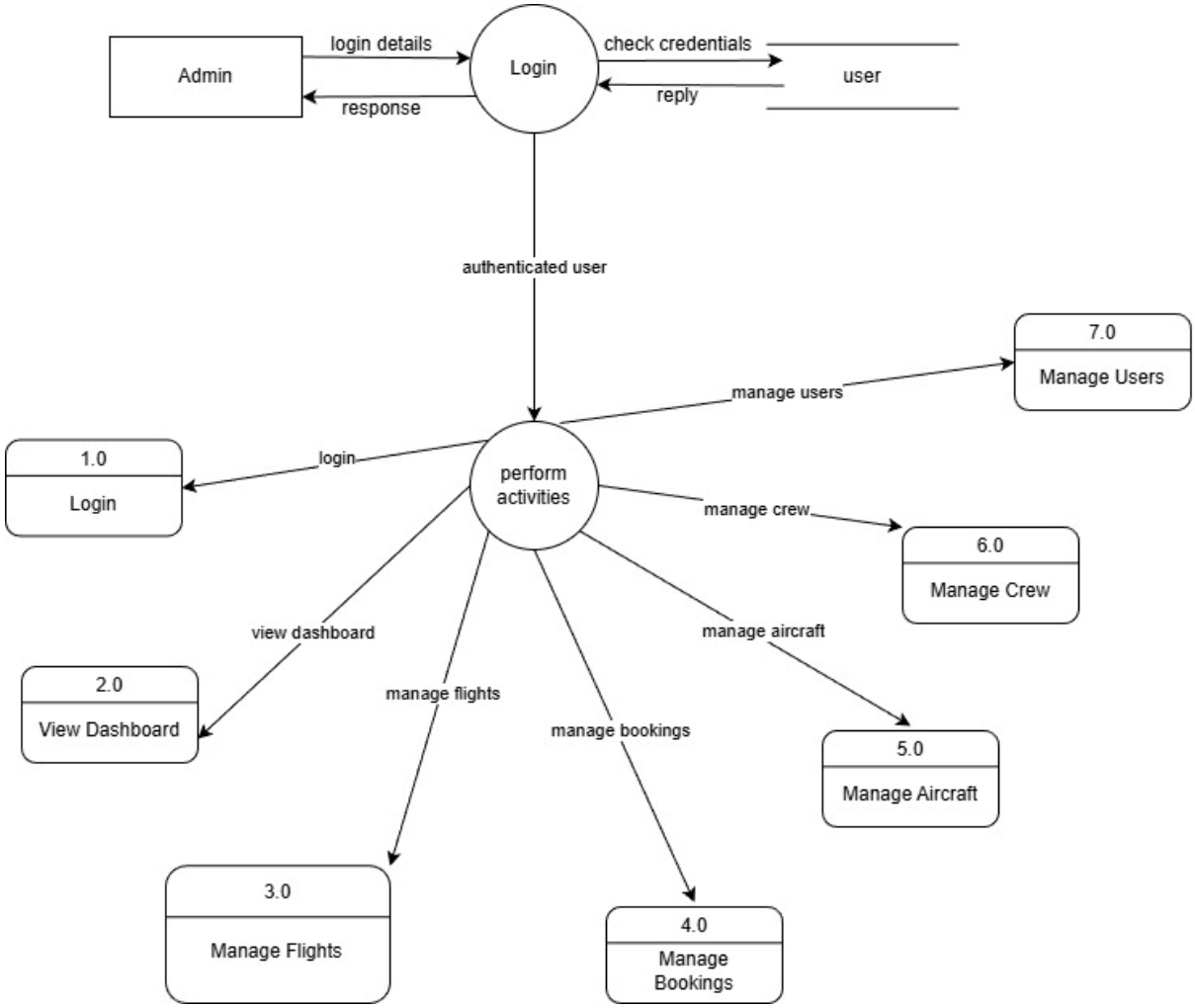


Figure 1.6.3 Admin DFD Diagram(Level 1)

7. DFD Diagram(Level 2)

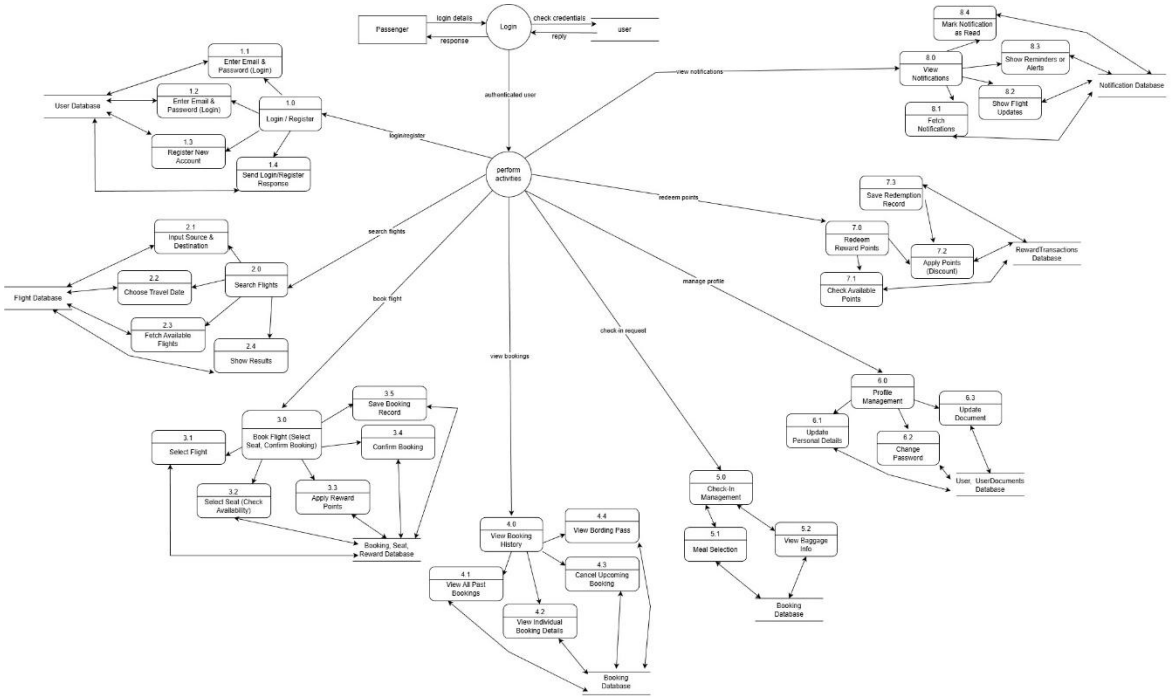


Figure 1.7.1 Passenger DFD Diagram(Level 2)

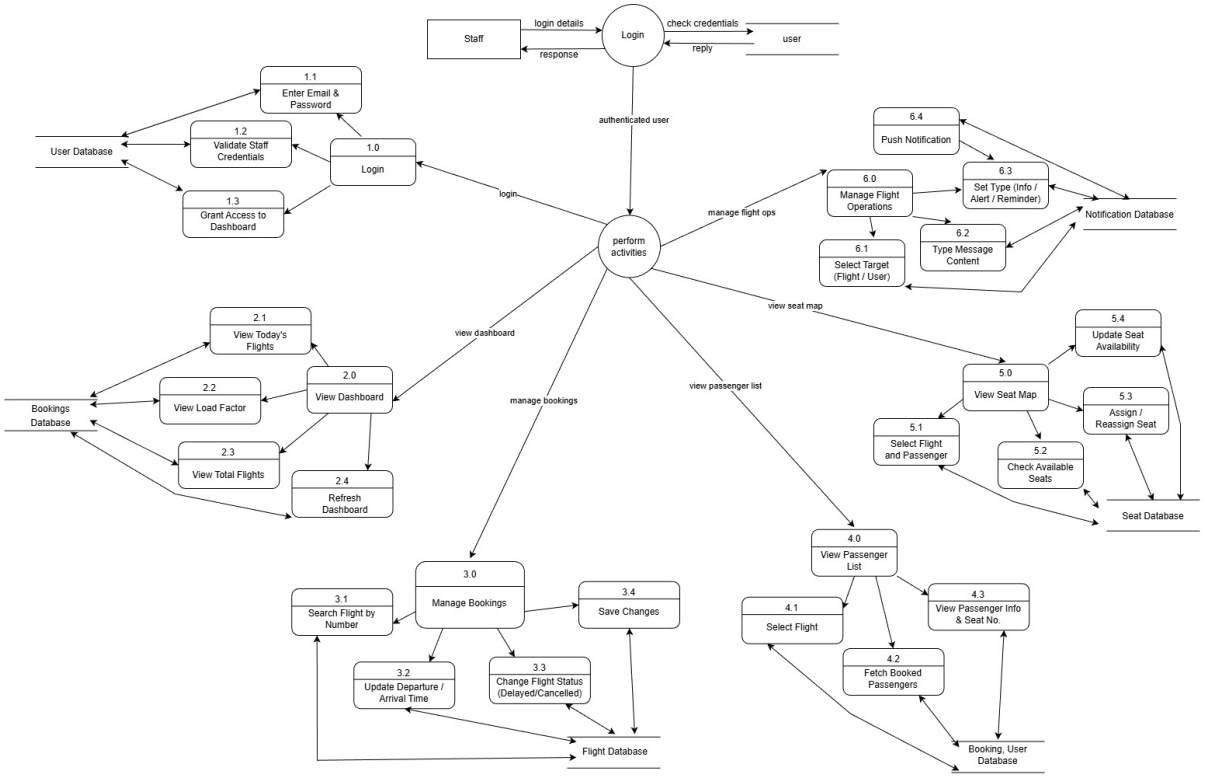


Figure 1.7.2 Staff DFD Diagram(Level 2)

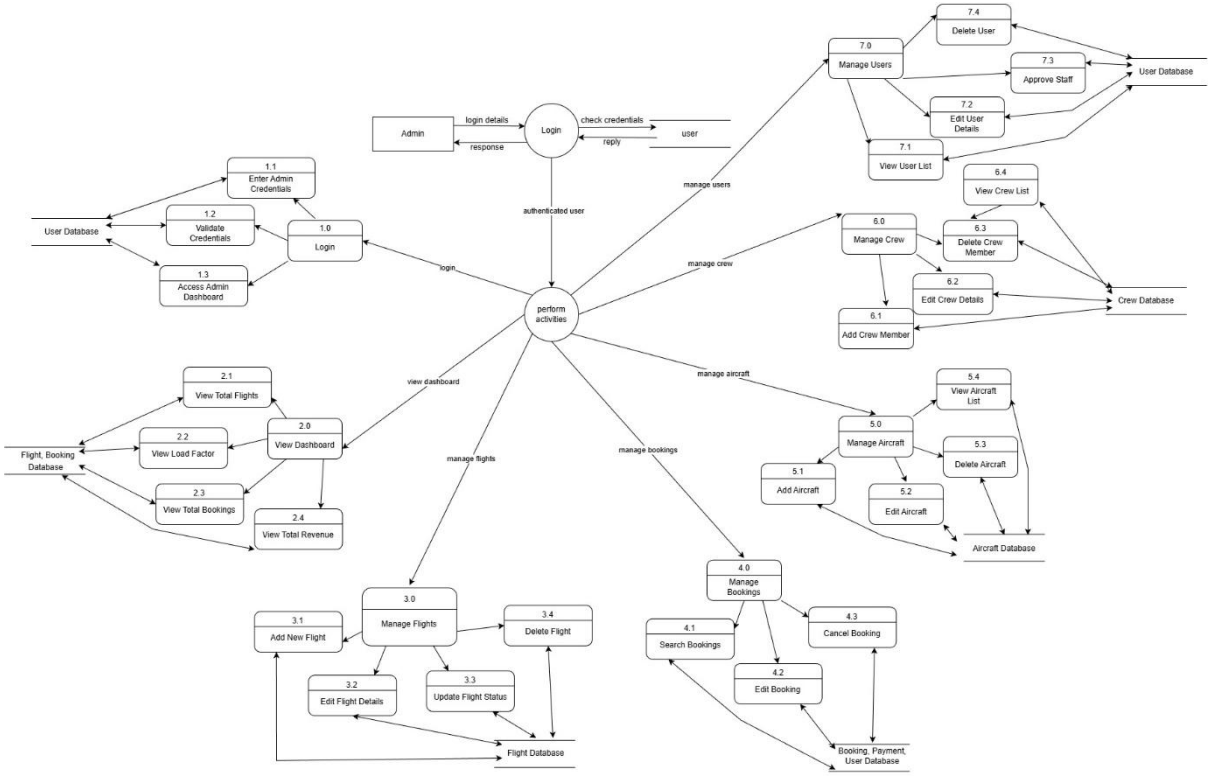


Figure 1.7.3 Admin DFD Diagram(Level 2)

8. Class Diagram

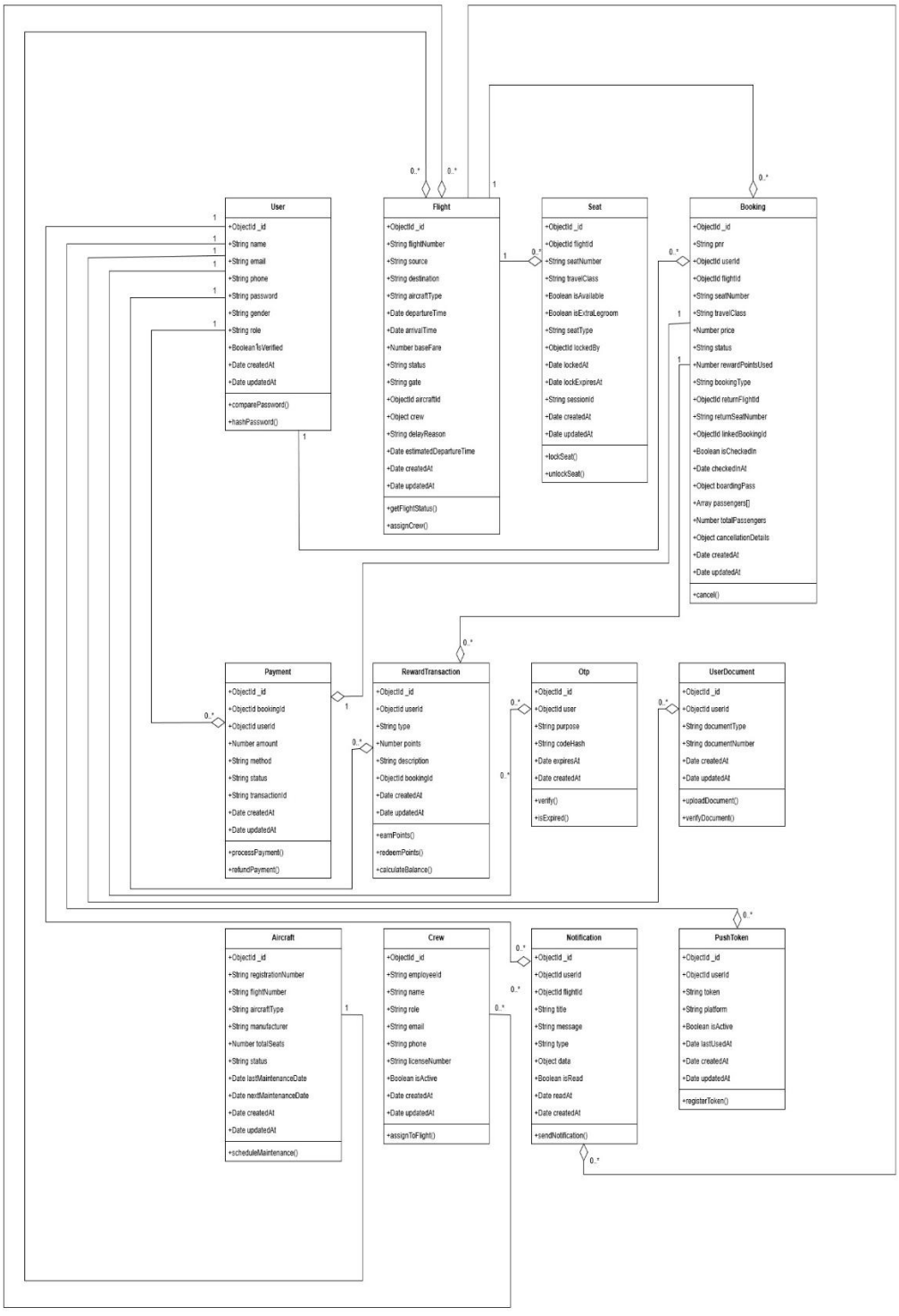
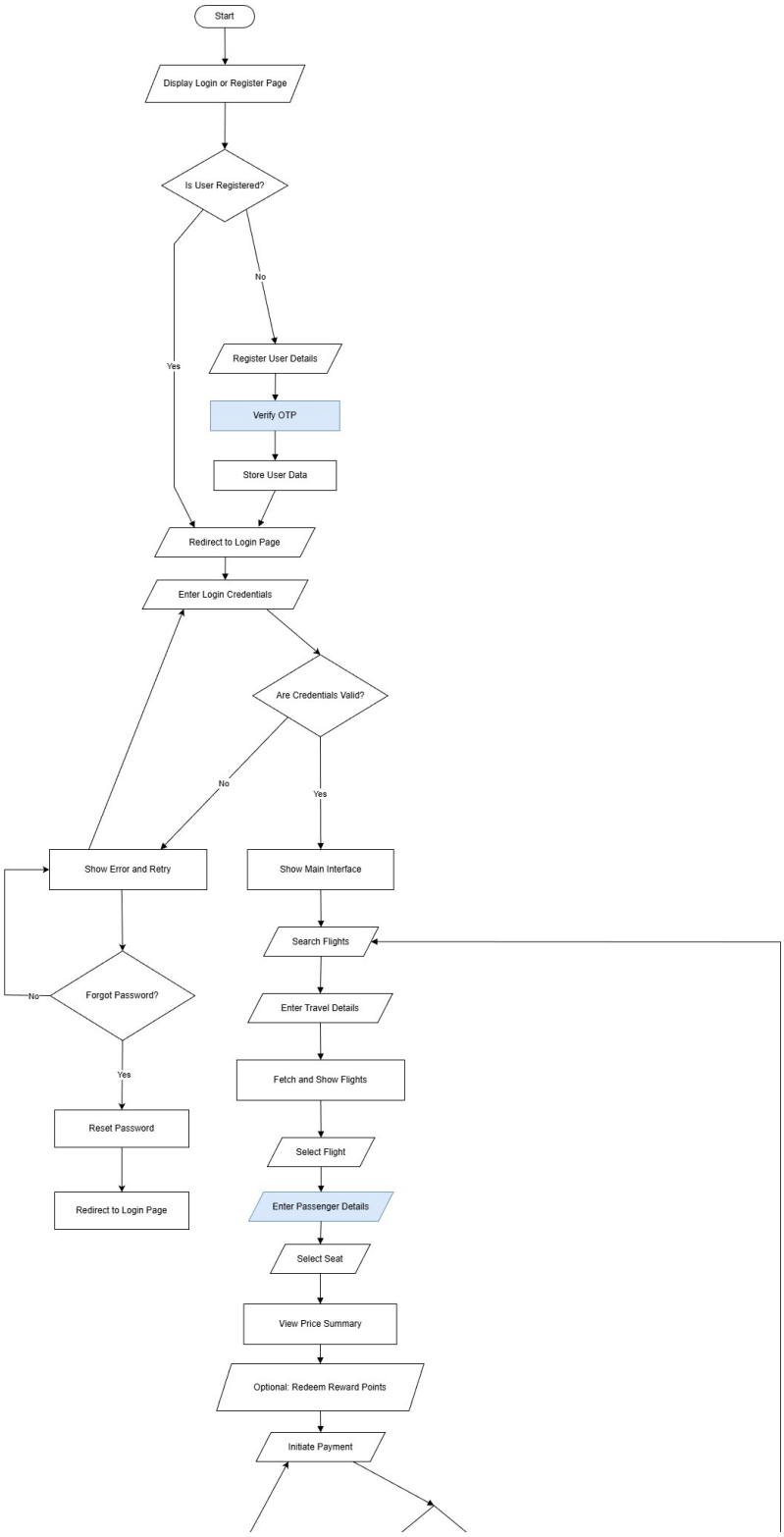


Figure 1.8 Class Diagram

9. Flowchart


Start
Display Login or Register Page
Is User Registered?
No
Yes
Register User Details
Verify OTP
Store User Data
Redirect to Login Page
Enter Login Credentials
Are Credentials Valid?
No
Yes
Show Error and Retry
Show Main Interface
Forgot Password?
No
Yes
Reset Password
Redirect to Login Page
Search Flights
Enter Travel Details
Fetch and Show Flights
Select Flight
Enter Passenger Details
Select Seat
View Price Summary
Optional: Redeem Reward Points
Initiate Payment

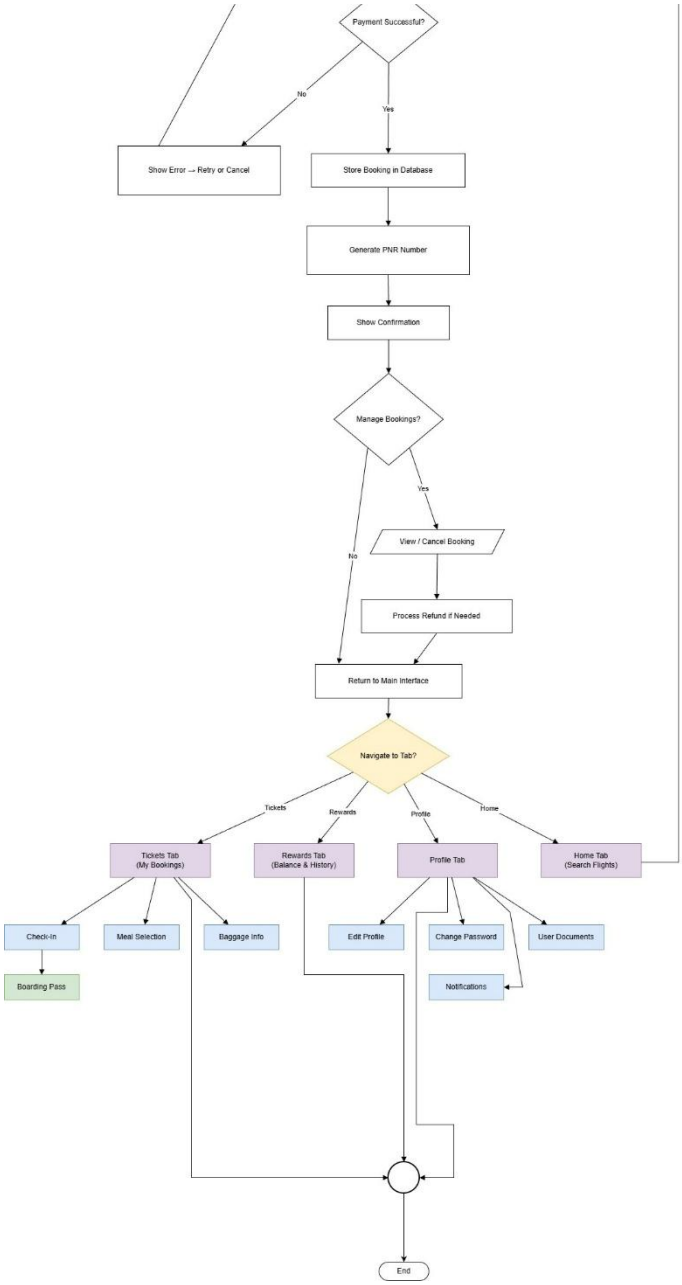
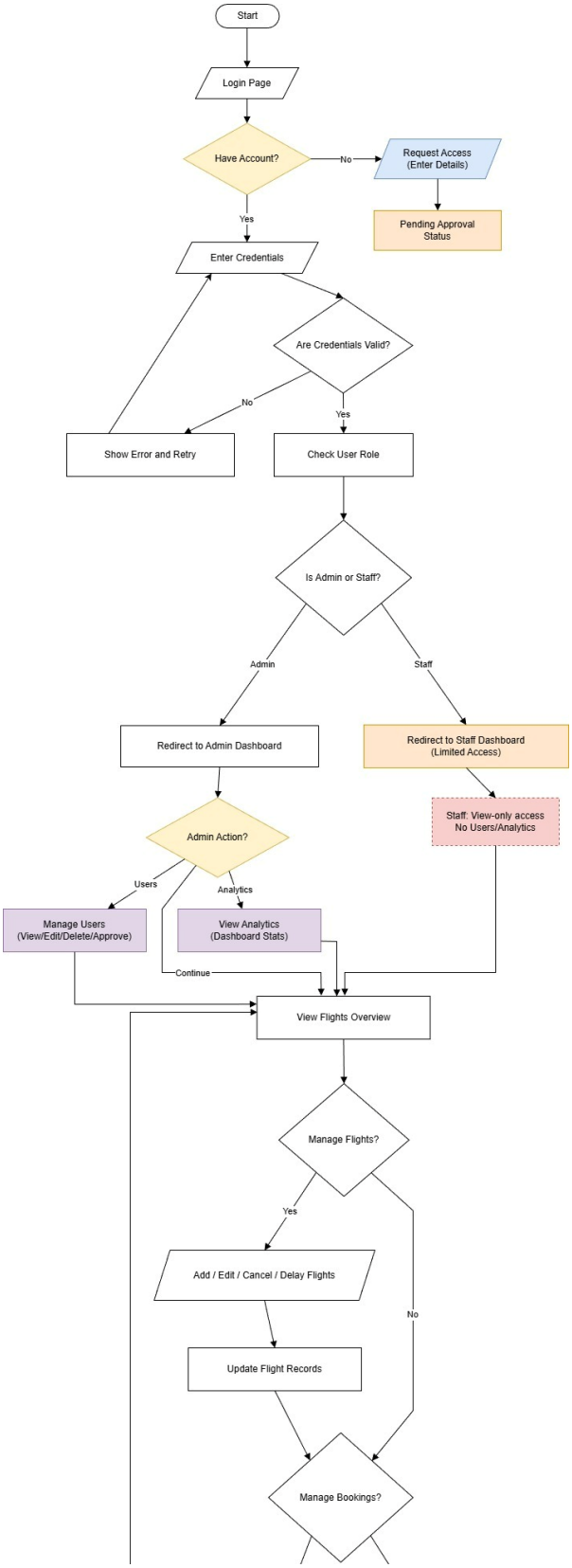


Figure 1.9.1 Passenger Flowchart


Start
Login Page
Have Account?
No
Request Access (Enter Details)
Pending Approval Status
Yes
Enter Credentials
Are Credentials Valid?
No
Yes
Show Error and Retry
Check User Role
Is Admin or Staff?
Admin
Staff
Redirect to Admin Dashboard
Redirect to Staff Dashboard (Limited Access)
Staff: View-only access No Users/Analytics
Admin Action?
Users
Analytics
Manage Users (View/Edit/Delete/Approve)
View Analytics (Dashboard Stats)
Continue
View Flights Overview
Manage Flights?
Yes
Add / Edit / Cancel / Delay Flights
No
Update Flight Records
Manage Bookings?

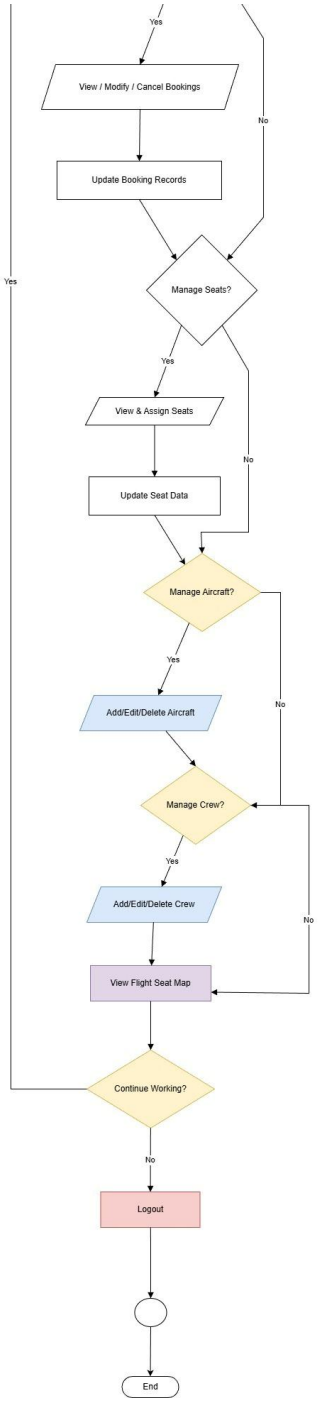


Figure 1.9.2 Dashboard Flowchart

1. Introduction

1.1 Document Purpose

This document is the SRS for our project titled "AerisGo - Airline Booking Platform with Passenger and Operations Interfaces." It explains the important details about the software system, including its main features, functions, and how it is expected to behave. This SRS is prepared for version 1.0 of the AerisGo application.

The main purpose of this document is to clearly define what the system should do so that everyone involved, such as the development team, project guide, testers, and other reviewers, has the same understanding. It covers all parts of the system: the mobile app for passengers, the web app for passengers, the web dashboard for airline staff, and the shared backend APIs and database. This document will act as the base for designing, developing, and testing the complete software solution.

1.2 Product Scope

AerisGo is a complete airline booking and management platform designed to serve both passengers and airline operations teams. The system has three main parts: a mobile application for passengers, a web application for passengers, and a web dashboard for airline staff and admins. The main purpose of this software is to make flight booking simple for passengers and to help the operations team manage flights, bookings, and related activities in an efficient way.

The system aims to offer a smooth and user-friendly experience to passengers on both mobile and web, allowing them to search and book flights, choose seats, view and manage their bookings, check in online, and earn reward points. On the other side, the staff and admins can use the dashboard to manage flight schedules, aircraft and crew, monitor and modify bookings, and update flight status such as delays or gate changes. By using a shared backend, the system ensures data consistency and easier maintenance. Overall, AerisGo aims to bring speed, convenience, and reliability to both passengers and the operations team involved in airline travel.

1.3 Intended Audience and Document Overview

This document is mainly prepared for our project guide and any client or evaluator who wants to understand the complete working of the AerisGo system. It will also be useful for developers, testers, and anyone who might work on or maintain this system in the future.

The SRS includes all the necessary details about the software, such as what features it has, how users interact with it, and how the system is designed behind the scenes. It covers the mobile app for passengers, the web app for passengers, the web dashboard for airline staff and admins, and the shared backend and database that connect all of them.

Readers can start with the introduction and scope sections to get a general understanding of the system. Then, based on their role, they can move to specific sections. Developers and testers may focus more on system features, flows, and data models, while clients or professors may be more interested in the overall goals, benefits, and expected outcomes.

1.4 Definitions, and Abbreviations

This section includes important terms and short forms used in this document. It helps make the content easy to understand for everyone reading it.

Table 1.1: Definitions

Term	Definition
Passenger App	The mobile and web applications used by airline customers to search and book flights, select seats, check bookings, and manage their profile and rewards.
Operations Interface	The web-based dashboard used by airline staff and admins to manage flights, aircraft, crew, bookings, delays, and basic reports or analytics.
Seat Selection System	A feature in the passenger apps that allows users to choose their seats from an interactive layout of the aircraft, with seat availability and pricing shown in real time.

Reward Points	Points earned by passengers when they book flights, which can be stored in their account and used later for discounts on future bookings.
Booking	The process of reserving a flight ticket through the system, including selected flights, seats, passenger details, and payment information.
Backend	The server-side logic and database that handle all data, APIs, business rules, and communication between the mobile app, passenger web app, and staff dashboard.
User Roles	Types of users in the system, such as passengers, staff, and admins, each with different permissions and access levels.
Push Notification	A real-time alert sent to a user's mobile device about booking updates, flight changes, or reminders related to upcoming trips.

Table 1.2: Abbreviations

Abbreviation	Full Form
SRS	Software Requirements Specification
API	Application Programming Interface
CRUD	Create, Read, Update, Delete
DB	Database
GUI	Graphical User Interface
JSON	JavaScript Object Notation
MERN	MongoDB, Express.js, React.js, Node.js
OTP	One-Time Password
UI	User Interface
UX	User Experience

VPS	Virtual Private Server
OS	Operating System
HTTPS	HyperText Transfer Protocol Secure
TLS	Transport Layer Security
REST	Representational State Transfer
FCM	Firebase Cloud Messaging
APNs	Apple Push Notification service
SMTP	Simple Mail Transfer Protocol
2FA	Two-Factor Authentication
RBAC	Role-Based Access Control
AES	Advanced Encryption Standard
PCI-DSS	Payment Card Industry – Data Security Standard
XSS	Cross-Site Scripting
DDoS	Distributed Denial of Service
QR	Quick Response (code)
RAM	Random Access Memory
CPU	Central Processing Unit
PDF	Portable Document Format
GPS	Global Positioning System
WiFi	Wireless Fidelity
FAQ / FAQs	Frequently Asked Question(s)

1.5 Document Conventions

This document uses a clean and consistent format to ensure readability for all users. Text is written in Times New Roman (size 11 or 12), single-spaced, with 1-inch margins. Headings follow the standard SRS template. Reviewer notes are italicized to distinguish them from main content. Naming conventions like camelCase and snake_case are used appropriately for variables and database fields. The overall layout is designed to be simple and accessible for both technical and non-technical readers.

1.6 References and Acknowledgments

This document has been prepared based on the planning and design discussions done by our project team.

The following references were used during the preparation of this document:

[1] React Native, "Getting Started," React Native Documentation, used for project setup and mobile app structure. [Online]. Available: https://reactnative.dev/docs/getting-started

[2] React, "Learn React," React Documentation, used for component patterns, hooks, and state management in the web app and staff dashboard. [Online]. Available: https://react.dev/learn

[3] MongoDB, "MongoDB Manual," MongoDB Documentation, used for schema design, indexing, and data modeling concepts. [Online]. Available: https://www.mongodb.com/docs/

[4] Express.js, "Guide," Express.js Documentation, used for routing, middleware, and REST API structure. [Online]. Available: https://expressjs.com/

[5] Node.js, "API Reference Documentation," Node.js Documentation, used for core modules and server configuration. [Online]. Available: https://nodejs.org/docs/latest/api/

[6] Expo, "Overview," Expo Documentation, used for app configuration, navigation, and native capabilities in the mobile app. [Online]. Available: https://docs.expo.dev/

2. Overall Description

2.1 Product Perspective

AerisGo is a new and independent software system that connects passengers and airline operations on a single platform. It is not based on any old system; everything is built from scratch to match today's travel needs and digital habits.

The system has three main interfaces: a passenger mobile app, a passenger web app, and an operations web dashboard for staff and admins. The passenger mobile app is built with React Native and Expo so it runs smoothly on both Android and iOS. The passenger web app is built with React and lets users book and manage their trips from any browser. The operations dashboard is a web application that staff and admins use to manage flights, aircraft, crew, bookings, delays, cancellations, and day to day operations.

Through the passenger apps, users can search and book flights, choose seats, manage bookings, check in online, view boarding passes, add baggage and meals, and use reward points. Through the operations dashboard, staff and admins can monitor passenger lists, update flight details, manage seat assignments, and view key analytics and reports that help them run flights more smoothly.

A core feature of AerisGo is the real time seat selection system. Passengers see a live seat map with clear colors for available, booked, selected, and premium seats. When a passenger selects a seat, it is locked for a short time so others cannot select it at the same moment. Staff can also see and adjust these seats in real time, which makes the whole process more transparent and under control.

All these interfaces share the same backend API built with Node.js, Express.js, and MongoDB, following a MERN based architecture. This shared backend keeps data consistent across mobile, web, and dashboard, and makes the system easier to maintain and grow in the future.

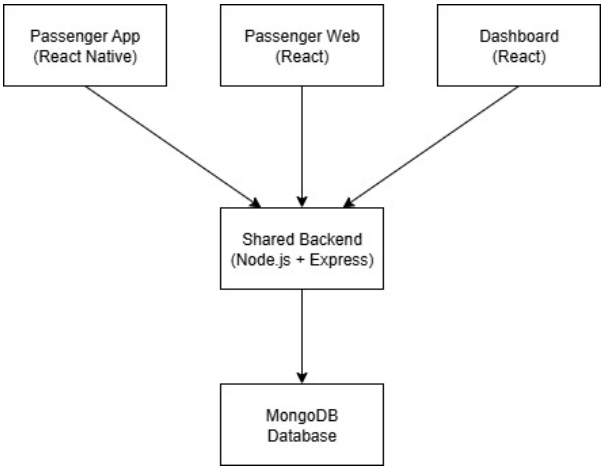


Figure System Diagram

2.2 Product Functionality

The AerisGo platform has two main sides:

The passenger side (mobile app and web app) and the operations side (web dashboard for staff and admins). Both sides work together to make booking and airline operations smooth and well managed.

Passenger side functions (mobile app and web app)

- Sign up, log in, and manage profile details
- Search flights by source, destination, date, and number of passengers
- View flight options with price, airline, duration, and basic flight info
- Select seats using an interactive seat map with real time seat availability and seat locking
- Book flights with a payment flow and see a clear price breakdown
- View booking history, tickets, and e tickets for past and upcoming trips
- Cancel bookings and view cancellation details and refund status
- Use reward points to earn on bookings and redeem on future trips
- Add or update baggage and meal preferences where available
- Check in online and view boarding passes
- Read FAQs and get basic support and guidance

On the mobile app, passengers also receive push and in app notifications for booking confirmations, flight reminders, delays, and gate changes.

Operations web dashboard functions (staff and admin)

The operations interface is a web dashboard built with a MERN based stack. Staff and admins can:

- Log in securely with role based access (admin and staff)
- Add, edit, cancel, or delay flights and update flight status
- Manage aircraft and crew details and assign them to flights
- View all bookings and passenger lists for each flight
- Manually book or cancel tickets in special or support cases
- Assign or change seats for passengers and view real time seat maps
- Manage user accounts and approvals where required
- View reports and analytics on daily or weekly bookings, revenue, and load factor
- Send email or in app notifications for important flight changes or alerts

Backend and shared functions

Behind both sides there is a single shared backend:

- One backend API used by mobile app, passenger web app, and operations dashboard
- Real time updates for seats, bookings, and flight status
- Common MongoDB database for users, flights, bookings, payments, rewards, and notifications
- Central handling of authentication, authorization, pricing, payments, rewards, and schedulers

This mix of passenger apps and staff dashboard, with a shared backend, helps both customers and airline staff use the system easily from anywhere and keeps all data in sync.

2.3 Users and Characteristics

AerisGo is designed to be used by two main groups of users passengers (customers) and airline staff (operations team). These users will use different parts of the system based on their needs, role, and experience level.

1. Passengers (Customers)

- These are the people who use the mobile app or web app to search for and book flights.
- Most of them may not be very technical, so the app needs to be simple, clean, and user-friendly.
- They will use features like flight search, booking, seat selection, profile updates, checking booking history, and managing reward points.
- They will also receive alerts or updates about their flight status.

Importance: This is the most important user group as the platform is mainly built to give them a smooth and modern flight booking experience.

2. Airline Staff (Operations Team)

- These users will use the web-based operations interface.
- They are usually trained staff with some technical knowledge.
- Their work includes managing flights, updating schedules, handling bookings, assigning seats, and sending notifications.
- Based on their role (Admin or Staff), they will have different levels of access. For example, only admins can cancel or reschedule flights.

Importance: This group is also very important as they keep the whole system running in the background and ensure everything is up to date for the passengers.

3. Admin

- Admins are part of the staff but with more control and access.
- They can manage user roles, check the stats, monitor revenues, and make important changes like cancelling flights or changing fares.

Importance: Admins play a key role in keeping everything well-managed and smooth from the backend.

2.4 Operating Environment

The AerisGo system is planned to run smoothly on both mobile and web platforms. The mobile app, built using React Native with Expo, will support Android (version 8 and above) and iOS (version 12 and above). This ensures that most modern smartphones and tablets can run the app without any issues.

The web application, used by the airline operations team or passenger, will be developed using the MERN stack. It will work properly on commonly used browsers like Google Chrome, Mozilla Firefox, and Microsoft Edge. A standard computer with at least 4 GB of RAM and a dual-core processor is suitable for running the operations interface.

Instead of using a cloud database service, the complete backend setup, including the database, will be hosted on a private VPS server. This server will run a Linux-based operating system, such as Ubuntu 24.04. It will host the Node.js backend, MongoDB database, and any supporting services like notifications and payments. Hosting on a VPS gives more control over performance, improves data privacy, and allows flexible configuration and upgrades.

The overall structure will follow a client-server model. Both the mobile and web applications will connect to the backend through APIs. The backend will manage user requests, booking logic, seat selection, and other services. Real-time updates, such as live seat availability and flight status, will be handled using sockets or periodic updates.

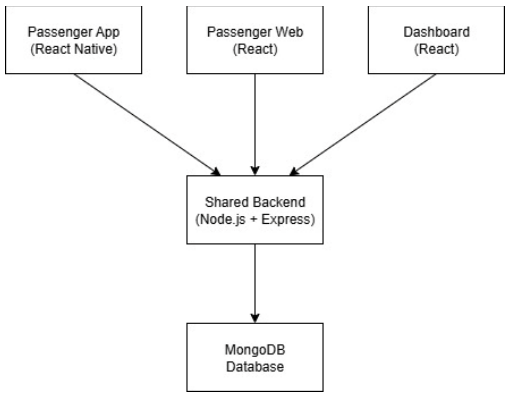


Figure System Diagram

2.5 Design and Implementation Constraints

While building this system, there are a few technical and practical limitations that developers will have to keep in mind. These constraints will affect the way the software is designed and developed. Based on the current setup and choices made for technologies, the following key constraints have been identified:

1. **Hosting on VPS only**

   The entire backend, database, and services will be hosted on a single VPS server. This means server resources like RAM, CPU, and storage must be managed carefully to avoid performance issues, especially during high traffic.

2. **Fixed Tech Stack (MERN + React Native)**

   The development must strictly use the MERN stack and React Native for the mobile app. This limits flexibility in choosing tools or libraries outside this stack.

3. **Cross-Platform Mobile Support (Expo Limitations)**

   Since Expo is being used for the mobile app, some advanced native features or deep customizations might not be possible unless detached from Expo, which adds complexity.

4. **Real-Time Seat Selection**

   Implementing real-time seat selection comes with its own set of challenges, such as data synchronization, conflict handling when multiple users try to book the same seat, and timely UI updates.

5. **Limited External APIs**

   Integration with only selected APIs like payment gateways and notification services means we must handle everything else manually, including user authentication, verification, and admin operations.

6. **Security and Data Privacy**

   Since passenger data, payment info, and admin access will be involved, the system must be secure. However, implementing strong security within limited server resources and a tight budget can be challenging.

2.6 User Documentation

For this flight seat booking system, proper user documentation will be needed to help both travellers and admin users operate the system easily. A user manual for the mobile app and web app will guide passengers through steps like signing up, searching for available flights, viewing seat layouts in real-time, selecting seats, and completing bookings smoothly and also for check-in. For the admin side, there will be a separate guide explaining how to manage flights, schedules, seat arrangements, and booking details through the dashboard. In-app hints and short help notes will be shown wherever necessary to make the experience self-explanatory. Additionally, basic tutorial videos or image-based walkthroughs may also be included. All documentation will be delivered in easy-to-read PDF format and made accessible within both the mobile app or web app and admin dashboard.

2.7 Assumptions and Dependencies

- The system assumes that the number of concurrent passengers using the application will not exceed 10,000 users at any given time.
- It is assumed that all users will have access to stable internet connectivity while using the passenger mobile app, web app or the operations web app.
- The system is designed under the assumption that all flights, their schedules, and seat configurations will be added and maintained by the airline operations team using the web dashboard.
- The real-time seat selection system assumes quick and reliable server responses from the backend hosted on a dedicated VPS server.
- Payment gateway integration will be mocked or simplified for development purposes and may not reflect a fully compliant production-grade payment system.
- The database and backend services assume the use of MongoDB initially, but since a dedicated VPS server is used, any required database can be installed and managed as needed.
- All notifications (push, in-app, and email) depend on third-party services or integrations, which may have rate limits, downtime, or other limitations.
- Reward points logic assumes a basic point-earning model per booking and simple redemption without complex tiering or partner integrations.

3. Specific Requirements

3.1 External Interface Requirements

3.1.1 User Interfaces

The system will include two major types of user interfaces:

1. Passenger Applications (Mobile and Web)

The passenger side includes a mobile app built with React Native and Expo and a web app built with React. Both follow a clean, modern, and simple design. The layout is Optimized for touch on mobile and for mouse and keyboard on web, but the overall look and flow stay consistent so that passengers can switch between mobile and web easily.

Key screens (common across mobile app and passenger web app):

- **Welcome / Home / Onboarding screen**

  First view of the app or site, highlights main features and shows quick access to flight search.

- **Login / Sign up screen**

  Fields for email, password and other login details, with options for forgot and reset password.

- **Flight search screen**

  Inputs for From, To, Date and number of passengers, with options to change trip type and basic filters.

- **Flight results screen**

  List of matching flights with main details such as price, timing, duration and airline, with sort and filter options.

- **Flight details and seat selection screen**

  Shows full flight information and an interactive real time seat map with color codes for available, booked, selected and premium seats.

- **Passenger details, baggage and meals screen**

  Forms to enter passenger information and choose baggage and meal options where available.

- **Booking summary and payment screen**

  Shows price breakdown, selected seats, applied rewards and total amount to pay, with a payment flow.

- **Booking confirmation screen**

  Shows booking ID, flight and seat details, and access to ticket or boarding pass information.

- **My bookings or tickets screen**

  List of upcoming, past and cancelled bookings, including flight status like delayed or on time.

- **Profile and rewards screen**

  View and edit profile data, see reward balance and history of points earned and used.

- **Notifications screen**

  List of in app or push notifications related to bookings, reminders, flight delays and other updates.

- **Help and FAQs screen**

  Static list of common questions and simple guidance for passengers.

2. Operations Dashboard

The operations side is a web portal used by airline staff and admins. It has a responsive layout suitable for desktop and tablet screens and focuses on clear data tables, charts and quick actions.

Key Screens:

- **Login screen**

  Secure login form for staff and admins.

- **Dashboard screen**

  Overview of today's flights, active bookings, key revenue and load factor stats and recent alerts.

- **Flight management screen**

  Add, edit, cancel or delay flights, update gates and statuses using structured forms.

- **Bookings management screen**

  View and filter bookings by flight, date or status, open passenger lists and manually modify or cancel bookings when needed.

- **Seat management screen**

  View real time seat maps for flights and assign or change seats for passengers.

- **User and staff management screens**

  Manage user accounts, approvals and staff access where required.

- **Notifications screen**

  Create and send email or in app alerts to passengers or groups based on flight or booking status.

- **Reports and analytics screen**

  View visual reports on load factor, delay trends, revenue and reward usage to support operational decisions.

3.1.2 Hardware Interfaces

The AerisGo system is primarily a software-based platform designed for flight booking and operations management. However, there are a few essential hardware interface considerations to ensure smooth interaction between the user devices and the system:

1. Mobile Devices (Passenger App)

- The mobile application will run on Android and iOS smartphones and tablets.
- No direct hardware-level interfacing is required. The app will communicate with the backend via HTTPS over the internet.
- **Supported Features**:
  - Push notifications (requires integration with Firebase Cloud Messaging for Android and Apple Push Notification service for iOS).
  - Device local storage (AsyncStorage or SecureStore for caching login tokens).
  - Network connection check using device-level APIs.

- **Hardware Dependencies**:
    - Internet connection (Wi-Fi or mobile data).
    - Working GPS is optional (can be used for future enhancements like airport check-ins).
    - QR code scanner (camera access) for viewing/validating tickets.

2. Web Browsers on PCs (Dashboard or Passenger Web App)

- The web portal will be accessed via modern desktop or laptop browsers (e.g., Chrome, Firefox, Edge).
- There is no direct physical hardware communication beyond the device's network card and display.
- **Minimum Requirements**:
    - Internet connectivity.
    - Standard keyboard and mouse input.
    - At least 8 GB RAM and a dual-core processor for smooth use of admin dashboards.

3. Server-Side (VPS Hosting)

- All backend components (Node.js server, MongoDB, etc.) will be deployed on a private VPS (Virtual Private Server).
- The server must support:
    - Node.js (version 18 or higher)
    - MongoDB Community Edition
    - Nginx (for reverse proxy setup)
    - Secure Shell (SSH) access for deployment and maintenance
- All data exchange between client and server will use REST APIs over HTTPS.

4. Push Notification Hardware Services

- Utilizes external cloud-based services to interface with mobile hardware:
    - **FCM** for Android and **APNs** for iOS.

3.1.3 Software Interfaces

The AerisGo system includes two main interfaces: a mobile application for passengers or web app for passengers and a web application for the airline operations team. These applications, along with the shared backend, will interact with the underlying operating systems on both client and server sides.

1. Mobile Application (passenger app)

- **Operating Systems**: Android (version 8.0 and above), iOS (version 12 and above)
- **Interface Type**: The app will interact with the mobile operating system through React Native APIs provided by Expo.
- **Functions**:
    - Access device storage (for login/session tokens)
    - Manage permissions (e.g., camera for QR scanning)
    - Push notification services
    - Network status detection
- **Libraries Used**:
    - expo, react-native, expo-notifications, expo-secure-store, axios, etc.

2. Web Application (dashboard or web app of passenger)

- **Operating Systems**: Cross-platform (Web-based, accessed via any OS with a modern browser – Windows, Linux, macOS)
- **Interface Type**: Runs within browsers; does not directly depend on OS-level features.
- **Functions**:
    - Fetch data using REST APIs from backend
    - Display real-time updates using polling or future WebSocket support

3. Backend Server

- **Operating System**: Linux (Ubuntu 24.04 LTS recommended)
- **Interface Type**: Node.js runtime will be installed on the Linux VPS.

- **Functions**:
    - Serve REST APIs to both mobile and web clients
    - Interact with MongoDB (self-hosted on the same VPS)
    - Handle request routing, authentication, and data validation

The backend will communicate with the MongoDB database using native drivers (mongoose), and all client-server communication will happen over HTTPS to ensure security.

3.1.4 Communications Interfaces

The AerisGo system will rely on standard internet communication protocols to enable secure and efficient data exchange between the mobile app, web interfaces, and backend services. All communication between clients and the server will use HTTPS with TLS encryption, ensuring that user credentials, booking data, and payment details are transmitted securely.

The system will primarily use RESTful APIs for typical request-response communication, with all messages formatted in JSON. In addition to REST APIs, WebSockets will be used for real-time features such as seat availability updates, flight status changes (e.g., delays or cancellations), and live notifications within both the passenger and operations interfaces. WebSockets will maintain persistent, low-latency, bi-directional communication channels between clients and the backend.

Push notifications will be delivered using FCM for Android and APNS for iOS, while emails and alerts may be sent through SMTP services if required. All communication systems will be secured with encryption, and the backend infrastructure on the VPS server will enforce strict access controls and firewall protections.

3.2 Functional Requirements

This section explains all the main things our system should do. We've divided the features into different parts based on their purpose. Each part includes the functions and tasks the user or the system will be able to perform. These are the things the mobile app and the web systems will handle for both passengers and airline staff.

3.2.1 Passenger App or Web App Functionalities

This part covers what normal users (passengers) can do from the mobile app or web app.

1. Account Creation and Login

- A person should be able to sign up using email or phone number along with a password.
- After registration, they can log in anytime using the same details.
- If they forget the password, they should be able to reset it using a link or OTP.

2. Profile Management

- Users can view and update their personal details like name, phone, email, gender, etc.
- They can also upload important documents like Aadhar or passport for faster bookings.

3. Search and Book Flights

- Passengers can search flights by entering source, destination, travel date, and number of passengers.
- The system will show available flights with details like time, airline, fare, and duration.
- They can filter and sort based on price, travel time, airline, etc.
- Users can select their preferred seat from an interactive seat layout.
- Final step will be booking the flight with a mock payment system.

4. Booking History

- Users should be able to view all their upcoming and past bookings.
- They can cancel bookings if needed and view cancellation or refund details.
- They can also view their tickets or boarding pass.

5. Reward Points System

- Each booking will give some reward points.
- These points can be seen inside the app and can be used for future bookings.
- A record of how and where points are used or earned will be visible.

6. Notifications

- Users will receive notifications for booking confirmations, flight delays, reminders, etc.
- Some messages will also appear directly inside the app as alerts.

7. Help and Support

- A help section will guide users with FAQs and basic support info.

3.2.2 Dashboard Functionalities

This section is for the airline staff and admin users who use the web-based system.

1. Login and Roles

- Staff and admin can securely log into the system.
- Admins can control access, while normal staff will only see what they are allowed to.
- Some actions like canceling flights will only be allowed to admins.

2. Managing Flights

- Admin or staff can add new flights with details like flight number, source, destination, aircraft type, timings, and fare.
- They can edit flight info, cancel flights, or mark delays.
- If a flight is canceled or delayed, the system will automatically notify affected passengers.

3. Booking Control

- Airline staff can view all bookings for any flight.
- They can also cancel or change bookings manually in special cases.
- Manual bookings can be created for VIP or walk-in passengers.

4. Seat Management

- Staff can see the seat map of any flight in real time.
- They can assign or reassign seats if needed, like for group passengers or elderly travelers.

5. Dashboard and Stats

- Admins can view stats on number of passengers, revenue, flight delays, load factor (how full flights are), and reward usage.
- Stats can be daily, weekly, or monthly.

6. Notifications and Alerts

- Staff can send push notifications or emails to individual passengers or groups for updates like delays or gate changes.

3.2.3 Shared Backend Functionalities

These features are common between mobile app and web dashboard, handled by the backend.

1. Common APIs

- One single backend system will provide all data and services to both mobile and web interfaces.
- This helps reduce errors and makes things easier to maintain.

2. Real-Time Seat Updates

- When someone selects or books a seat, the seat layout will update in real time so others don't select the same seat.
- This will be done using WebSockets.

3. Secure Data Handling

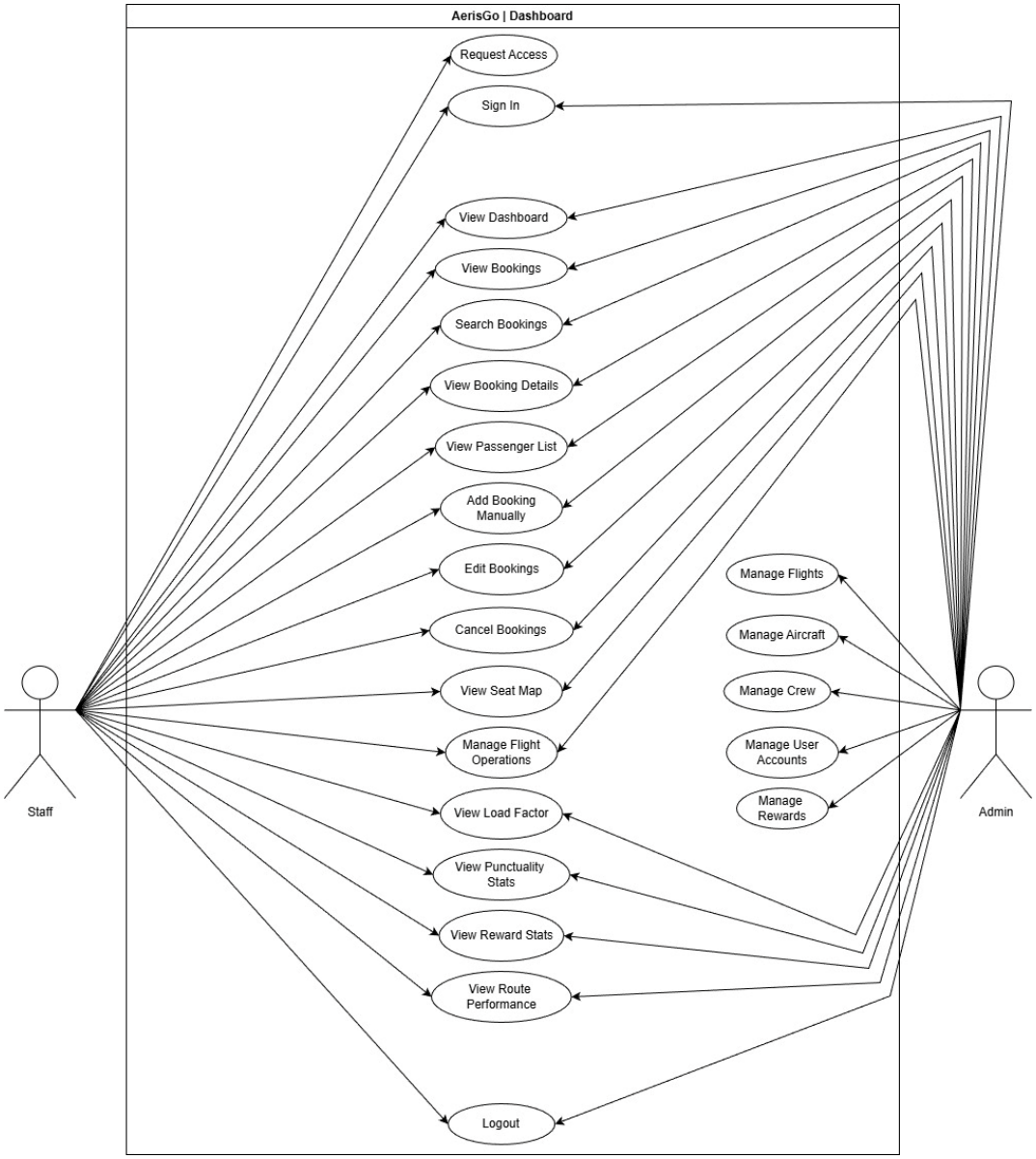
- All personal data, booking info, and payment records will be safely stored and managed.
- Passwords and sensitive data will be encrypted.

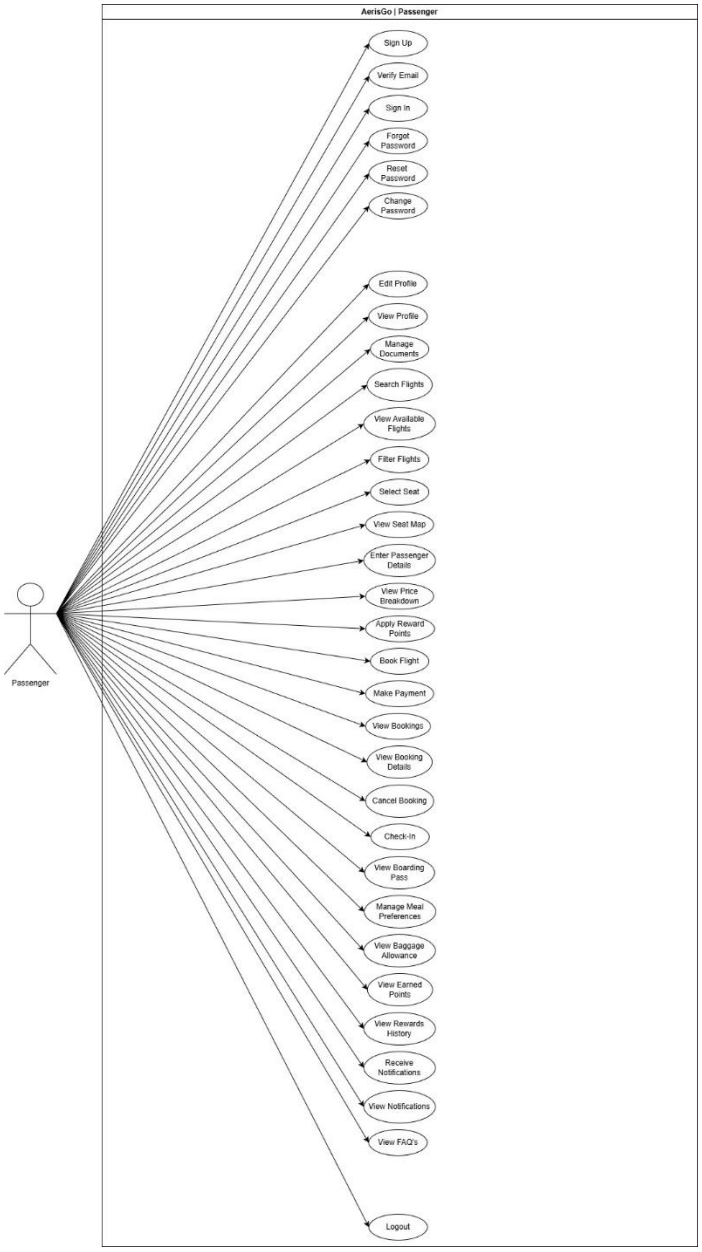
4. Push Notification System

- Backend will be responsible for triggering timely alerts based on user activity or flight updates.

3.3 Behaviour Requirements

3.3.1 Use Case View





1. AerisGo Dashboard – For Staff and Admin

This part of the system is used by airline staff and admins to manage everything behind the scenes. Here's a quick idea of what each role can do:

- **Staff can:**
    - Login to the system
    - View the dashboard
    - Manage bookings – view, search, cancel/modify, or even add manually
    - View passenger lists

    - Assign or reassign seats
    - Handle flight changes like delays or rescheduling
    - View stats and analyze data like load factors, punctuality, reward stats, etc.
    - Send push notifications or trigger email alerts
    - Logout once done
- **Admin has all the above access and in addition, they can:**
    - Add new flights or edit flight details
    - Cancel flights
    - Manage staff accounts

2. AerisGo Passenger App or Web App – For Customers

This is the mobile app or web portal that passengers use for their journey. It's designed to be smooth, friendly, and easy to use. Here's what passengers can do:

- Sign up and sign in
- Reset password and manage profile (edit/view)
- Search for flights and apply filters to find the right one
- Select seats, view seat maps and heatmaps
- Use seat perks (extra legroom, meals, etc.)
- Book flights and make payments
- View, cancel , check-in or check details of bookings
- View boarding pass
- View and redeem earned points (rewards)
- Check their rewards history
- Get real-time alerts (push or in-app notifications)
- Access help centre
- Logout after use

4. Other Non-functional Requirements

4.1 Performance Requirements

To make sure the AerisGo system works smoothly and gives users a good experience, we have set a few important performance expectations. These are meant to keep both passengers and airline staff happy, even during busy times like holidays or sale periods.

Below are the key performance requirements:

1. **Login Response Time**

   The system should allow users (passengers or staff) to log in within 3 seconds under normal internet conditions. Nobody likes waiting too long just to get in.

2. **Flight Search Speed**

   When a passenger searches for flights, the available results should load within 5 seconds. This helps users make faster booking decisions, especially when they're comparing dates and prices.

3. **Booking Confirmation**

   Once a user makes a payment and books a flight, the confirmation and ticket generation process should not take more than 10 seconds. Quick booking gives passengers peace of mind.

4. **Dashboard Load Time for Staff**

   When staff members access the operations dashboard, all graphs, numbers, and booking info should appear within 5 seconds, even if there are thousands of records. Fast access helps the staff manage flights efficiently.

5. **Push Notification Delivery**

   Important alerts like delays, reschedules, or promotional messages should be delivered to users' phones or web app within 2 seconds after being triggered from the system.

6. **Peak Hour Handling**

   The system should be able to support up to 1000 concurrent users without crashing or slowing down. This includes both passengers and staff during high-traffic times like festivals, weekend sales, or airport rush hours.

7. **Seat Map Load Time**

   While selecting seats, the seat map should load in less than 3 seconds, even for fully booked flights. This ensures a smooth and user-friendly booking experience.

4.2 Safety and Security Requirements

Safety and security are two of the most important aspects of the AerisGo platform. Since we're dealing with real-time airline operations and sensitive passenger data, we must ensure the system is trustworthy, reliable, and protected against misuse, errors, or cyber threats.

Safety Requirements

1. **Avoid Booking Overlaps**

   The system should not allow double-booking of a single seat on a flight. If a booking is already confirmed for a particular seat, it must be locked until payment is processed or the session times out.

2. **Flight Delay/Reschedule Controls**

   Only authorized airline staff (with the proper role) can delay, cancel, or reschedule flights. An accidental click or unauthorized access could create chaos for passengers and must be strictly avoided.

3. **Backup and Recovery System**

   All booking data, passenger records, and flight schedules must be backed up daily. In case of server failure, data loss, or any technical issue, the system should be able to restore all records within 2 hours.

Security Requirements

The client expects a high level of security, similar to what is standard in the travel and banking industry. This includes encryption of personal data, role-based access control, secure transactions, and protection from cyber attacks.

Here are the key security measures that must be followed:

1. **User Authentication**

   Every user (passenger, staff, or admin) must log in using secure credentials. Optional two-factor authentication (2FA) should be available for staff and admins.

2. **Role-Based Access Control (RBAC)**

   Different users will have different access rights. For example, only admins can edit flight details or manage staff, while passengers can only access their own profile and bookings.

3. **End-to-End Encryption**

   All personal and sensitive data (like passwords, payment details, and booking history) should be encrypted during storage and transfer using secure standards like AES or TLS.

4. **Secure Payment Gateway Integration**

   Payments should be processed through certified and PCI-DSS-compliant payment gateways to avoid credit card fraud or leaks.

5. **Session Timeout and Auto-Logout**

   If a user is inactive for more than 10 minutes, the session should automatically expire to avoid unauthorized access on public or shared devices.

6. **Audit Logging**

   All sensitive activities (like canceling flights, assigning staff roles, or changing schedules) should be logged with time stamps and user details for auditing and accountability.

7. **Protection from Common Attacks**

   The system must be protected against:

   - Cross-Site Scripting (XSS)
   - Distributed Denial of Service (DDoS) attacks
   - Brute-force login attempts

4.3 Software Quality Attributes

AerisGo is being built to be more than just a booking tool it needs to be smooth, stable, secure, and scalable. Below are the most important quality aspects that both our client and our development team care about. Each of these qualities ensures the system will work well for both passengers and staff, and will be easy to manage and grow in the future.

4.3.1 Reliability

The system should work without unexpected crashes or errors, even when used continuously. Since flight operations and bookings are time-sensitive, reliability is a must.

How we'll ensure reliability:

- Use of proven backend frameworks (like Node.js) known for handling traffic reliably
- Implementing automated server health checks and database monitoring
- Setting up fallback systems (like retry queues for booking or payment failures)
- Performing load testing to simulate peak times like festival or holiday seasons

Goal:
The system should be available 99.5% of the time on a monthly average and recover from non-critical failures within 1 minute.

4.3.2 Usability

The interface must be simple and easy to understand for both passengers and airline staff. Since users may come from different regions and backgrounds, we want the system to feel natural and friendly.

How we'll ensure usability:

- Use of icons, hints, and tooltips for clarity
- Clean and minimal UI with Indian design sensibilities (mobile-friendly, light mode by default)
- Testing with users from different regions to make sure everyone finds it easy
- Separate views for passengers, staff, and admins so no one sees extra or confusing options

Goal:
A new user should be able to perform key tasks (like booking or seat selection) without training, in under 3 steps.

4.3.3 Maintainability

Our software should be easy for developers to update, fix, and improve in the future. As flight operations grow and client needs change, we'll need to make modifications quickly.

How we'll ensure maintainability:

- Writing clean and modular code, following standard naming and structure practices
- Keeping documentation updated after every major feature or change
- Version-controlling everything using Git with branches, pull requests, and peer reviews
- Using Docker to manage deployment so future developers don't face "it works on my machine" issues

Goal:
Most small updates (like adding a new filter or fixing a bug) should be possible in under 2 hours by any trained developer.

4.3.4 Portability

The platform must run smoothly on different devices and operating systems, especially since passengers may access it through Android, iOS, or even older laptops.

How we'll ensure portability:

- Use of responsive web design and cross-platform frameworks like React Native for mobile
- Testing on common browsers (Chrome, Firefox, Safari) and devices

Goal:
The system should run consistently on at least 95% of modern browsers and mobile devices used in India.

4.3.5 Interoperability

The system may need to exchange data with other airline tools, third-party analytics systems, or payment gateways in the future.

How we'll ensure interoperability:

- Building APIs with clear documentation and support for JSON
- Following REST standards to make integration with external systems easy
- Keeping an API key-based access system for third-party access with security layers

Goal:
Any third-party developer should be able to connect their tool to AerisGo's data using our public API documentation in less than 1 day.

4.3.6 Design for Change (Flexibility)

We know things change, flight patterns, policies, user preferences. Our system should be ready for change, without needing to rewrite the entire codebase.

How we'll ensure flexibility:

- Using a microservices architecture or at least clearly separated modules for bookings, payments, notifications, etc.
- Storing configurations (like refund policies, fare rules) in the database instead of hardcoding them
- Keeping UI logic separated from backend logic so design changes won't affect core logic

Goal:
Policy changes (like increasing reward points or changing booking rules) should be possible without code changes, ideally via admin panel.

Appendix A – Data Dictionary

Table 2.1: User

Sr. No.	Field	Size	Datatype	Description	Keys
1	_id	24	OBJECT_ID	Unique id of user	Primary Key
2	name	255	VARCHAR	Store user name	-
3	email	255	VARCHAR	Store user email	Unique, Indexed
4	phone	10	VARCHAR	Store user phone number	Unique
5	password	255	VARCHAR	Store hashed user password	-
6	gender	10	ENUM	Store user gender	-
7	role	20	ENUM	Store user role (admin, staff, passenger)	Indexed
8	isVerified	1	BOOLEAN	Flag to show if user email is verified	-
9	createdAt	-	TIMESTAMP	Record created date and time	-
10	updatedAt	-	TIMESTAMP	Record last updated date and time	-

Table 3.1: Flight

Sr. No.	Field	Size	Datatype	Description	Keys
1	_id	24	OBJECT_ID	Unique id of flight	Primary Key
2	flightNumber	20	VARCHAR	Store flight number	Indexed
3	source	100	VARCHAR	Store flight source	Indexed
4	destination	100	VARCHAR	Store flight destination	Indexed
5	aircraftType	50	ENUM	Type of aircraft (for example a320 neo)	-
6	departureTime	-	DATETIME	Planned departure date and time	-
7	arrivalTime	-	DATETIME	Planned arrival date and time	-
8	baseFare	-	INT	Base fare of flight in INR	-
9	status	20	ENUM	Flight status (scheduled, cancelled, delayed, completed)	-

10	gate	10	VARCHAR	Gate number for boarding	-
11	aircraftId	24	OBJECT_ID	Link to aircraft document	Foreign Key (Aircraft)
12	crew.pilot	24	OBJECT_ID	Link to pilot crew member	Foreign Key (Crew)
13	crew.coPilot	24	OBJECT_ID	Link to co pilot crew member	Foreign Key (Crew)
14	crew.flightAttendants	-	ARRAY(OBJECT_ID)	List of flight attendant ids	Foreign Key (Crew)
15	delayReason	255	VARCHAR	Reason for delay if any	-
16	estimatedDepartureTime	-	DATETIME	Updated departure time in case of delay	-
17	createdAt	-	TIMESTAMP	Record created date and time	-
18	updatedAt	-	TIMESTAMP	Record last updated date and time	-

Table 4.1: Booking

Sr. No.	Field	Size	Datatype	Description	Keys
1	_id	24	OBJECT_ID	Unique id of booking	Primary Key
2	pnr	6	VARCHAR	Booking PNR code	Unique, Indexed
3	userId	24	OBJECT_ID	Link to user who created booking	Foreign Key (User), Indexed
4	flightId	24	OBJECT_ID	Link to main flight	Foreign Key (Flight), Indexed
5	seatNumber	10	VARCHAR	Seat number of primary passenger	-
6	travelClass	10	ENUM	Travel class (economy, business, first)	-
7	price	-	INT	Total price charged for booking	-
8	status	20	ENUM	Booking status (confirmed,	Indexed

				cancelled, pending)	
9	rewardPointsUsed	-	INT	Points used for this booking	-
10	bookingType	20	ENUM	Type of booking (one way, round trip)	Indexed
11	returnFlightId	24	OBJECT_ID	Link to return flight if round trip	Foreign Key (Flight)
12	returnSeatNumber	10	VARCHAR	Seat number for return flight	-
13	linkedBookingId	24	OBJECT_ID	Link to return flight if round trip	Foreign Key (Flight)
14	isCheckedIn	1	BOOLEAN	Flag to show if primary passenger has checked in	Indexed
15	checkedInAt	-	DATETIME	Date and time of check in	-
16	boardingPass.gate	10	VARCHAR	Gate	-

				assigned in boarding pass	
17	boardingPass.boardingTime	-	DATETIME	Boarding time	-
18	boardingPass.sequence	-	INT	Boarding sequence number	-
19	passengers.seatNumber	10	VARCHAR	Seat number of each passenger	-
20	passengers.isPrimary	1	BOOLEAN	Flag for primary passenger	-
21	passengers.fullName	255	VARCHAR	Full name of passenger	-
22	passengers.dateOfBirth	-	DATE	Date of birth of passenger	-
23	passengers.gender	10	ENUM	Gender of passenger	-
24	passengers.email	255	VARCHAR	Email of passenger (optional)	-
25	passengers.phone	15	VARCHAR	Phone of passenger (optional)	-
26	passengers.documentType	20	ENUM	Document	-

				type (aadhar, passport)	
27	passengers.documentNumber	50	VARCHAR	Document number	-
28	passengers.mealPreference	50	VARCHAR	Meal preference for passenger	-
29	totalPassengers	-	INT	Total number of passengers in booking	-
30	mealPreference	50	VARCHAR	Main meal preference for booking (single value)	-
31	cancellationDetails.cancelledAt	-	DATETIME	Date and time when booking was cancelled	-
32	cancellationDetails.cancellationFee	-	INT	Cancellation fee amount	-
33	cancellationDetails.refundAmount	-	INT	Refund amount after cancellation	-
34	cancellationDetails.refundStatus	20	ENUM	Refund	-

				status (pending, processed, failed, none)	
35	cancellationDetails.refundProcessedAt	-	DATETIME	Time when refund was processed	-
36	cancellationDetails.cancellationReason	255	VARCHAR	Reason provided for cancellation	-
37	createdAt	-	TIMESTAMP	Record created date and time	-
38	updatedAt	-	TIMESTAMP	Record last updated date and time	-

Table 5.1: Seat

Sr. No.	Field	Size	Datatype	Description	Keys
1	_id	24	OBJECT_ID	Unique id of seat record	Primary Key
2	flightId	24	OBJECT_ID	Link to flight	Foreign Key (Flight), Indexed
3	seatNumber	10	VARCHAR	Seat number	Unique with flightId
4	travelClass	10	ENUM	Travel class (economy, business, first)	-
5	isAvailable	1	BOOLEAN	Flag to show if seat is available for booking	-
6	isExtraLegroom	1	BOOLEAN	Flag to show if seat has extra legroom	-
7	seatType	20	ENUM	Seat type (window, aisle, middle, standard)	-
8	lockedBy	24	OBJECT_ID	User who has currently locked this seat	Foreign Key (User)
9	lockedAt	-	DATETIME	Time when seat was locked	Indexed for expiry queries

10	lockExpiresAt	-	DATETIME	DATETIME	Indexed
11	sessionId	100	VARCHAR	Session id used for seat lock tracking	-
12	createdAt	-	TIMESTAMP	Record created date and time	-
13	updatedAt	-	TIMESTAMP	Record last updated date and time	-

Table 6.1: Aircraft

Sr. No.	Field	Size	Datatype	Description	Keys
1	_id	24	OBJECT_ID	Unique id of aircraft	Primary Key
2	registrationNumber	20	VARCHAR	Aircraft registration number	Unique
3	flightNumber	20	VARCHAR	Default flight number for this aircraft	Unique
4	aircraftType	50	ENUM	Type of aircraft	-
5	Manufacturer	50	VARCHAR	Aircraft manufacturer name	-
6	totalSeats	-	INT	Total number of seats	-
7	status	20	ENUM	Status (active, maintenance, retired)	-
8	lastMaintenanceDate	-	DATE	Last maintenance date	-
9	nextMaintenanceDate	-	DATE	Next planned maintenance date	-

10	createdAt	-	TIMESTAMP	Record created date and time	-
11	updatedAt	-	TIMESTAMP	Record last updated date and time	-

Table 7.1: Crew

Sr. No.	Field	Size	Datatype	Description	Keys
1	_id	24	OBJECT_ID	Unique id of crew member	Primary Key
2	employeeId	20	VARCHAR	Employee id of crew member	Unique
3	name	255	VARCHAR	Crew member name	-
4	role	20	ENUM	Role (pilot, co pilot, flight attendant)	-
5	email	255	VARCHAR	Crew email	Unique
6	phone	15	VARCHAR	Crew phone number	-
7	licenseNumber	50	VARCHAR	License number (mainly for pilots)	-
8	isActive	1	BOOLEAN	Flag to show if crew member is active	-
9	createdAt	-	TIMESTAMP	Record created date and time	-
10	updatedAt	-	updatedAt	Record last updated date and time	-

Table 8.1: Payment

Sr. No.	Field	Size	Datatype	Description	Keys
1	_id	24	OBJECT_ID	Unique id of payment	Primary Key
2	bookingId	24	OBJECT_ID	Link to booking	Foreign Key (Booking), Indexed
3	userId	24	OBJECT_ID	Link to user who paid	Foreign Key (User), Indexed
4	amount	-	INT	Amount paid in INR	-
5	method	20	ENUM	Payment method (card)	-
6	status	20	ENUM	Payment status (pending, success, failed, refunded)	Indexed
7	transactionId	100	VARCHAR	External transaction id	-
8	createdAt	-	TIMESTAMP	Record created date and time	-
9	updatedAt	-	TIMESTAMP	Record last updated date and time	-

Table 9.1: RewardTransaction

Sr. No.	Field	Size	Datatype	Description	Keys
1	_id	24	OBJECT_ID	Unique id of reward transaction	Primary Key
2	userId	24	OBJECT_ID	Link to user	Foreign Key (User), Indexed
3	type	10	ENUM	Type of transaction (earn, redeem)	-
4	points	-	INT	Number of points added or deducted	-
5	description	255	VARCHAR	Short description of transaction	-
6	bookingId	24	OBJECT_ID	Related booking if any	Foreign Key (Booking)
7	createdAt	-	TIMESTAMP	Record created date and time	-
8	updatedAt	-	TIMESTAMP	Record last updated date and time	-

Table 10.1: Notification

Sr. No.	Field	Size	Datatype	Description	Keys
1	_id	24	OBJECT_ID	Unique id of notification	Primary Key
2	userId	24	OBJECT_ID	User who receives notification	Foreign Key (User), Indexed
3	flightId	24	OBJECT_ID	Related flight	Foreign Key (Flight), Indexed
4	title	255	VARCHAR	Short title of notification	-
5	Message	500	VARCHAR	Full notification message	-
6	Type	20	ENUM	Type (info, alert, reminder)	Indexed
7	data	-	JSON	Extra data payload	-
8	isRead	1	BOOLEAN	Flag to show if notification is read	Indexed
9	readAt	-	DATETIME	Time when notification was read	-
10	createdAt	-	TIMESTAMP	Record created date and time	-

11	updatedAt	-	TIMESTAMP	Record last updated date and time	-

Table 11.1: Otp

Sr. No.	Field	Size	Datatype	Description	Keys
1	_id	24	OBJECT_ID	Unique id of OTP record	Primary Key
2	user	24	OBJECT_ID	Link to user	Foreign Key (User), Indexed
3	purpose	30	ENUM	Purpose (email verification, password reset)	Indexed
4	codeHash	255	VARCHAR	Hashed OTP code	-
5	expiresAt	-	DATETIME	Time when OTP expires	TTL Indexed
6	createdAt	-	DATETIME	Time when OTP was created	-

Table 12.1: PushToken

Sr. No.	Field	Size	Datatype	Description	Keys
1	_id	24	OBJECT_ID	Unique id of push token record	Primary Key
2	userId	24	OBJECT_ID	Link to user	Foreign Key (User), Indexed
3	token	255	VARCHAR	Push token string from device	Indexed
4	platform	10	ENUM	Device platform (android, ios)	Indexed
5	isActive	1	BOOLEAN	Flag to show if token is active	Indexed
6	lastUsedAt	-	DATETIME	Last time this token was used	-
7	createdAt	-	TIMESTAMP	Record created date and time	-
8	updatedAt	-	TIMESTAMP	Record last updated date and time	-

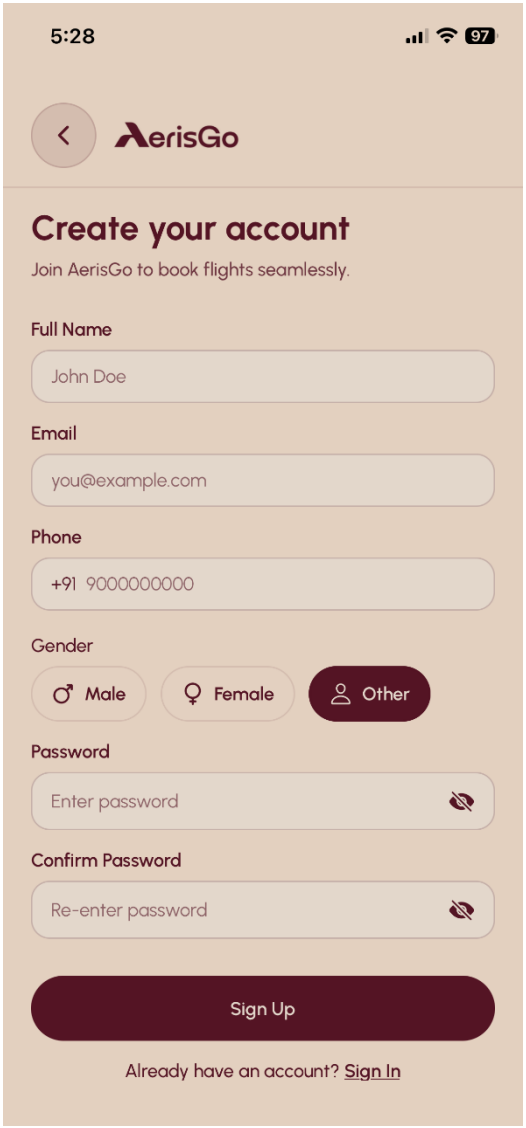
Table 13.1: UserDocument

Sr. No.	Field	Size	Datatype	Description	Keys
1	_id	24	OBJECT_ID	Unique id of user document record	Primary Key
2	userId	24	OBJECT_ID	Link to user	Foreign Key (User), Indexed
3	documentType	20	ENUM	Type of document (aadhar, passport)	Unique per user with documentType
4	documentNumber	50	VARCHAR	Document number	-
5	createdAt	-	TIMESTAMP	Record created date and time	-
6	updatedAt	-	TIMESTAMP	Record last updated date and time	-

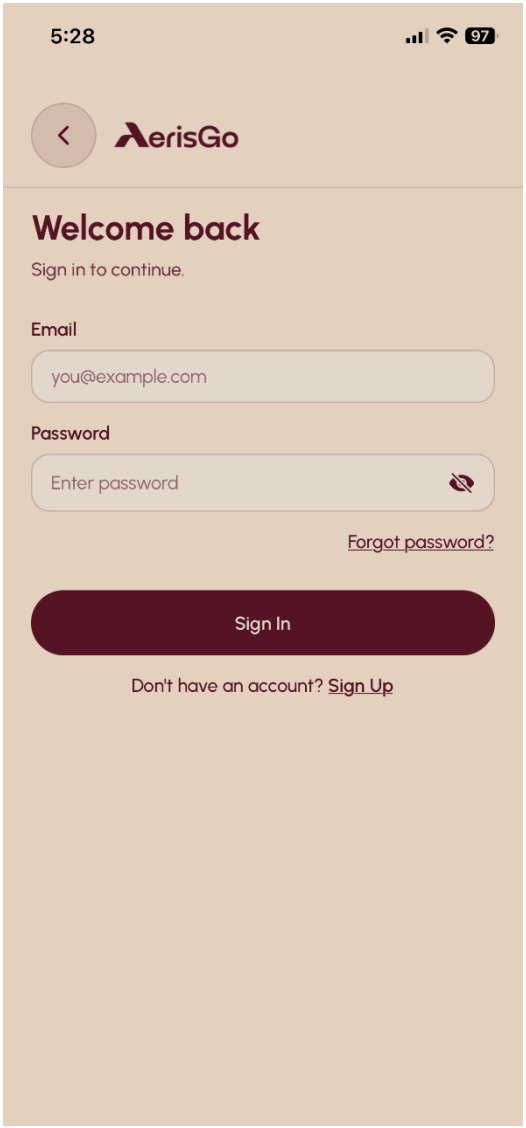
Appendix B – User Manual

Mobile App

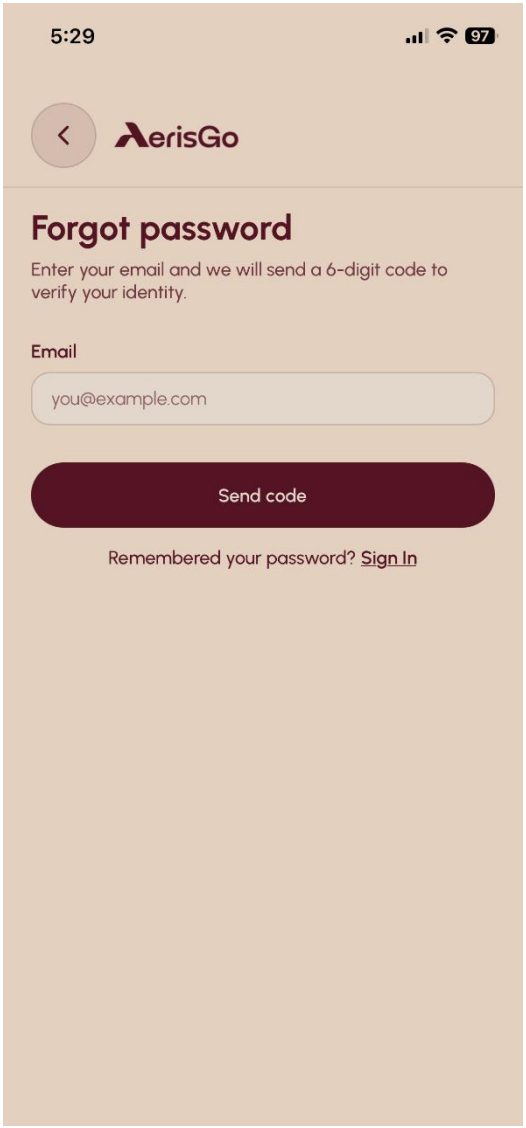
Welcome Screen: The welcome screen offers a quick introduction and navigates users to the main features of the app.



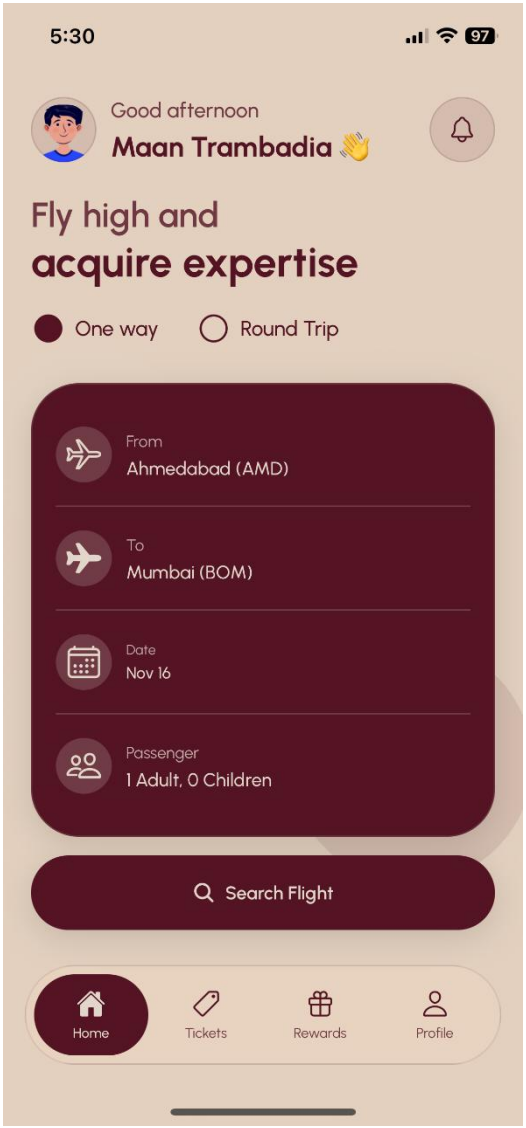
SignUp Screen: The sign-up screen allows new users to create an account by providing required information.



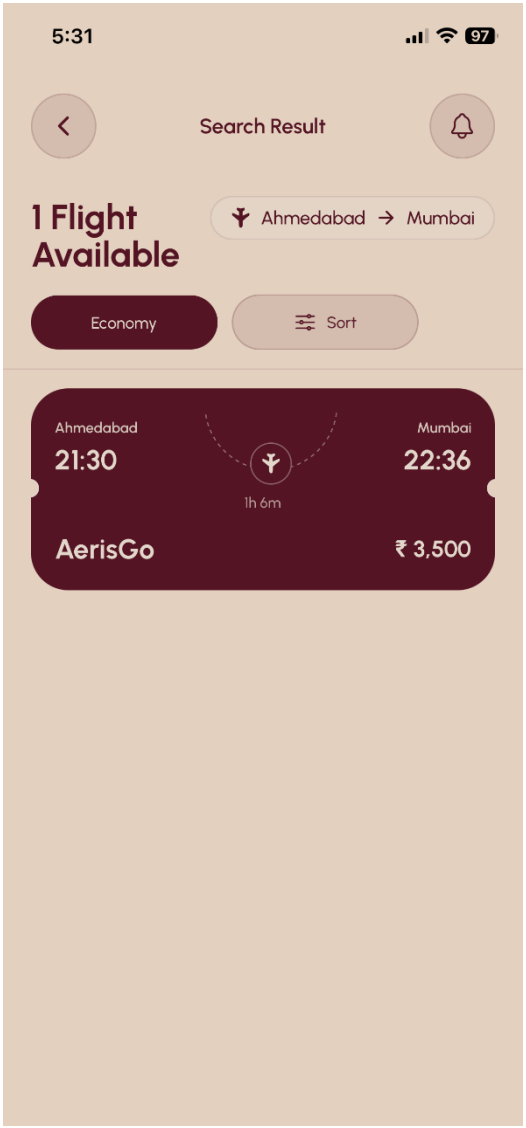
SignIn Screen: The sign-in screen allows registered users to securely access their accounts.



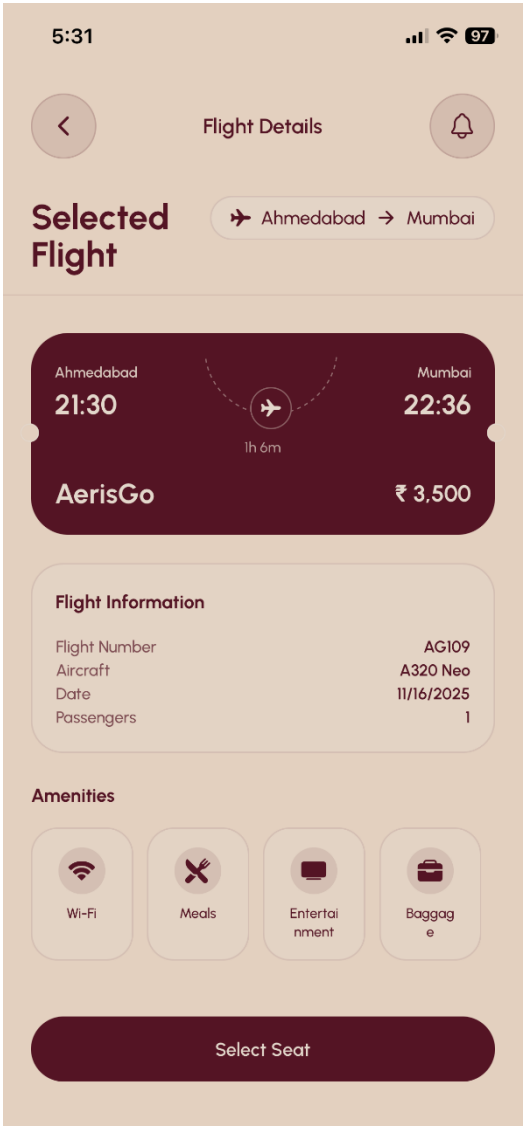
Forgot Password Screen: The forgot-password screen allows registered users to securely reset their account password.



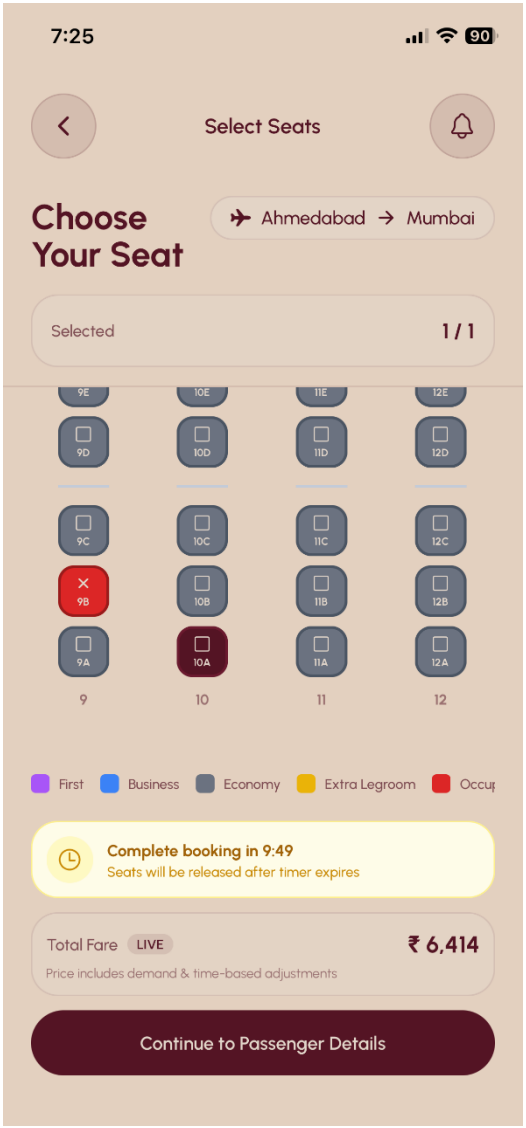
Home Screen: The home page serves as the main dashboard, providing access to core features of the app.



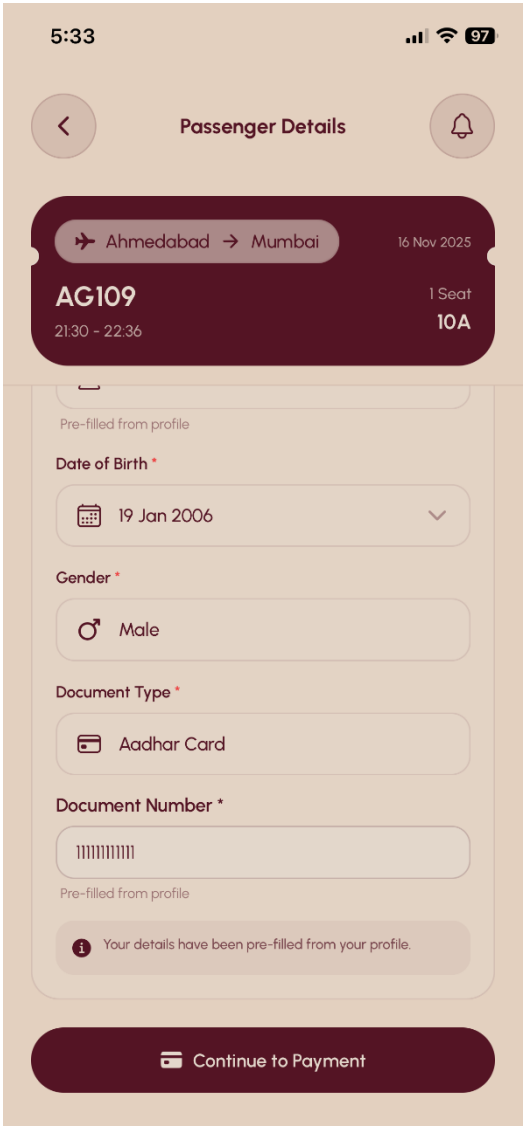
Search Screen: The search result screen displays a list of available flights based on the user's search criteria.



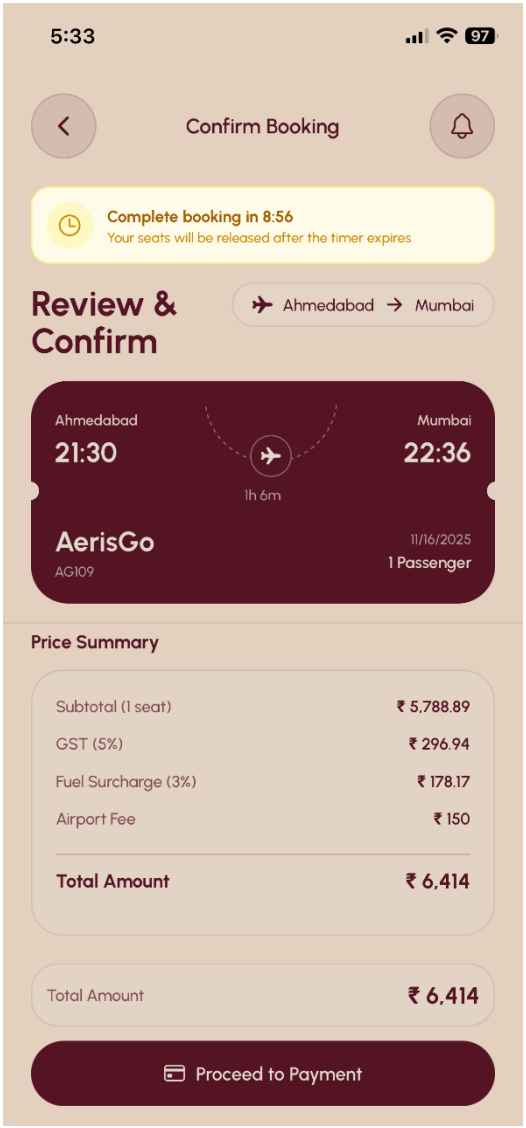
Flight Details Screen: The flight details screen displays flight details like flight number, aircraft, date and amenities.



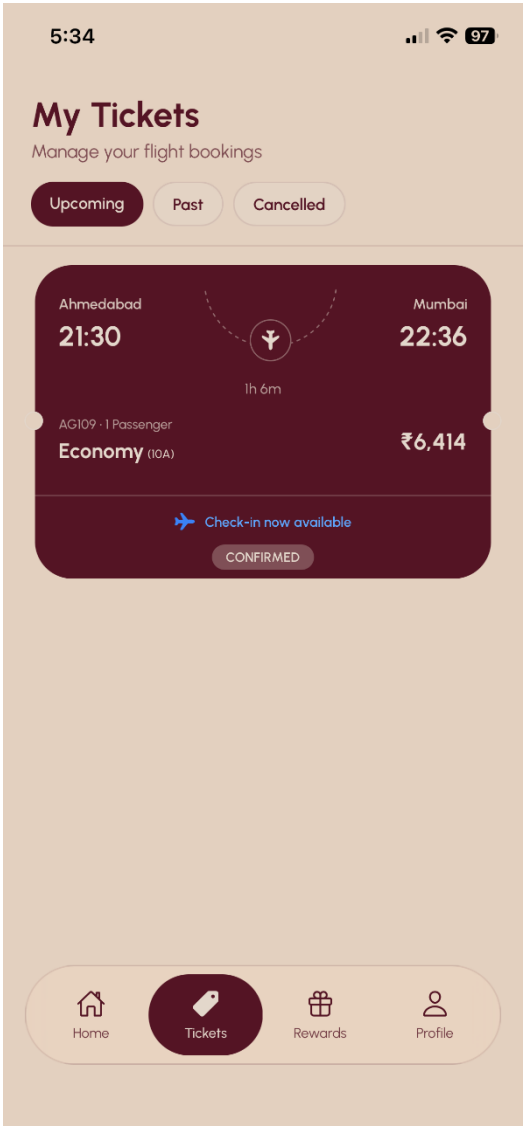
Seat Select Screen: In this screen passenger can select their own choose of seats with real-time feature in it.



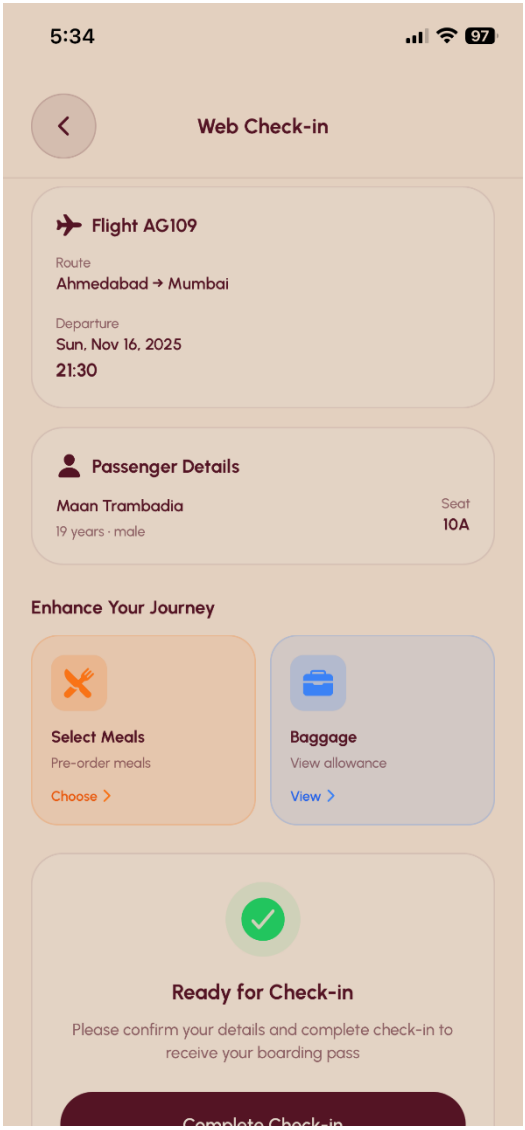
Passenger Details Screen: The passenger details screen displays a form to fill passenger details.



Confirm Booking Screen: This is last screen and final screen of booking flow it displays the price summary and all other flight, seat details after this it redirects to payment model.

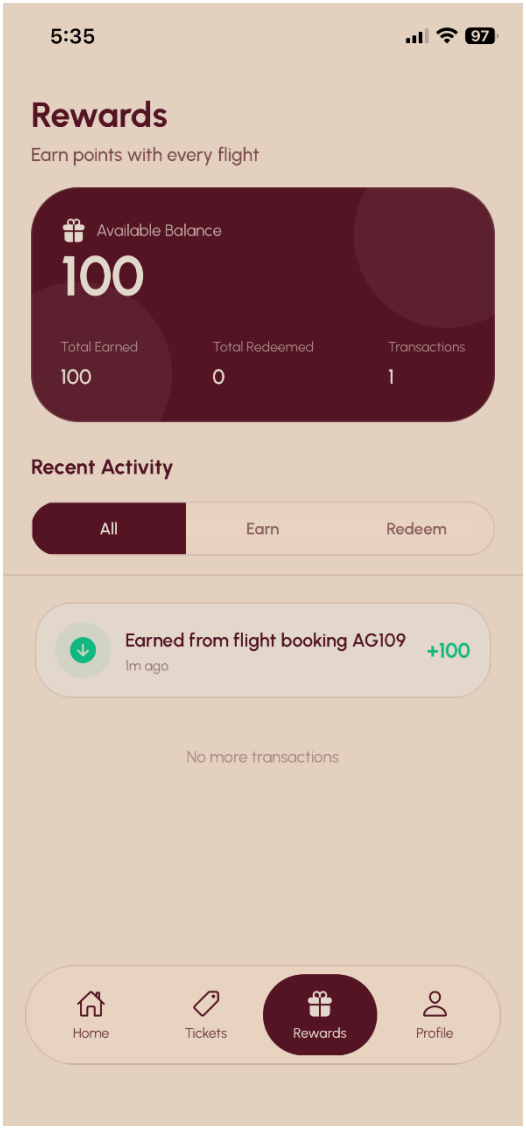


Tickets Screen: In this screen user can check there upcoming tickets, past and cancelled and my pressing on ticket it shows details and option to check-in or cancel the ticket.

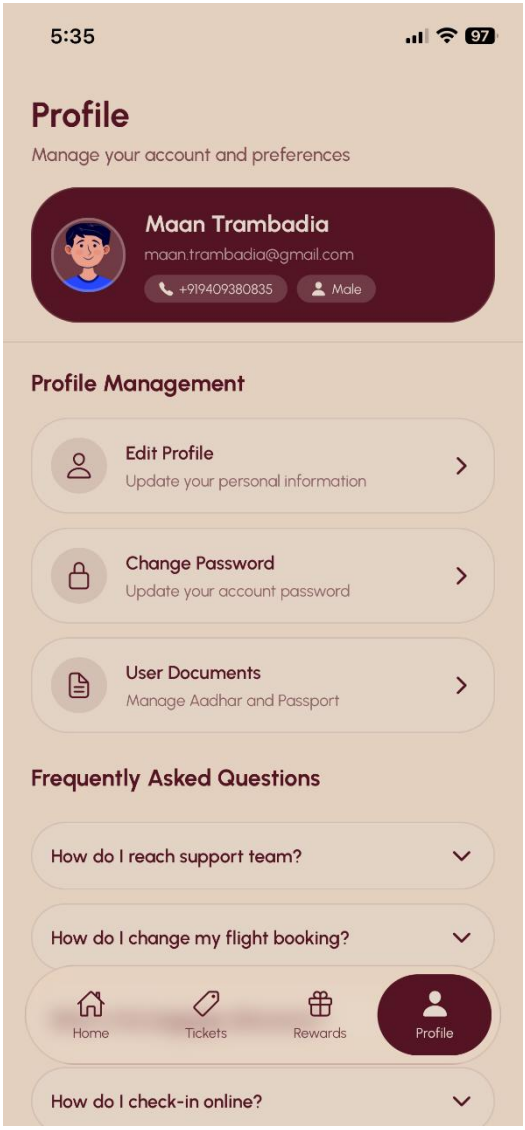


Check-in Screen: In this screen passenger can check-in and select their meals and also view baggage allowance.

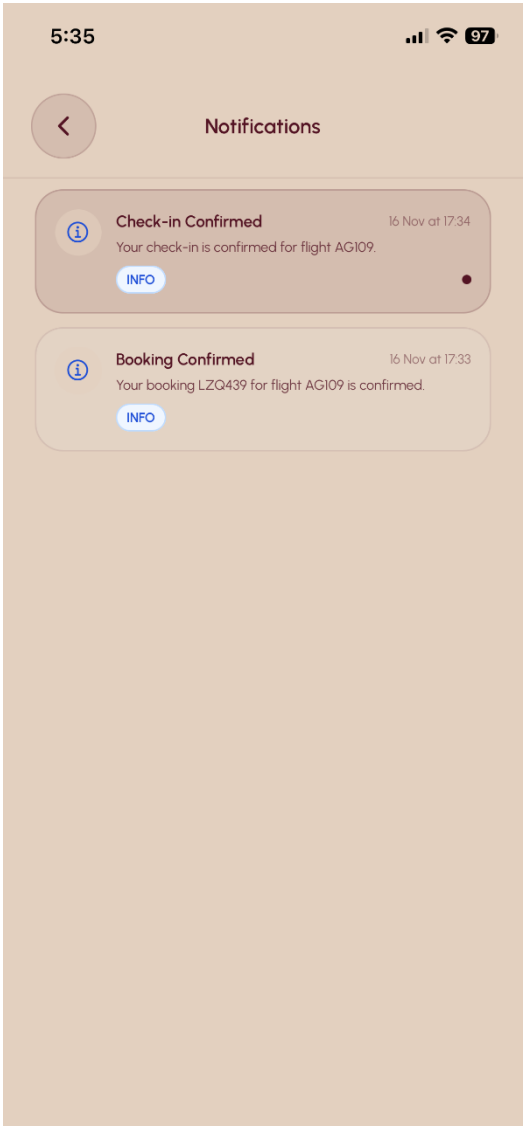
Boarding Pass Model: This is passengers boarding pass and they can see all the details like gate, class, PNR and etc.



Rewards Screen: In this screen user can see their current balance of points and history.

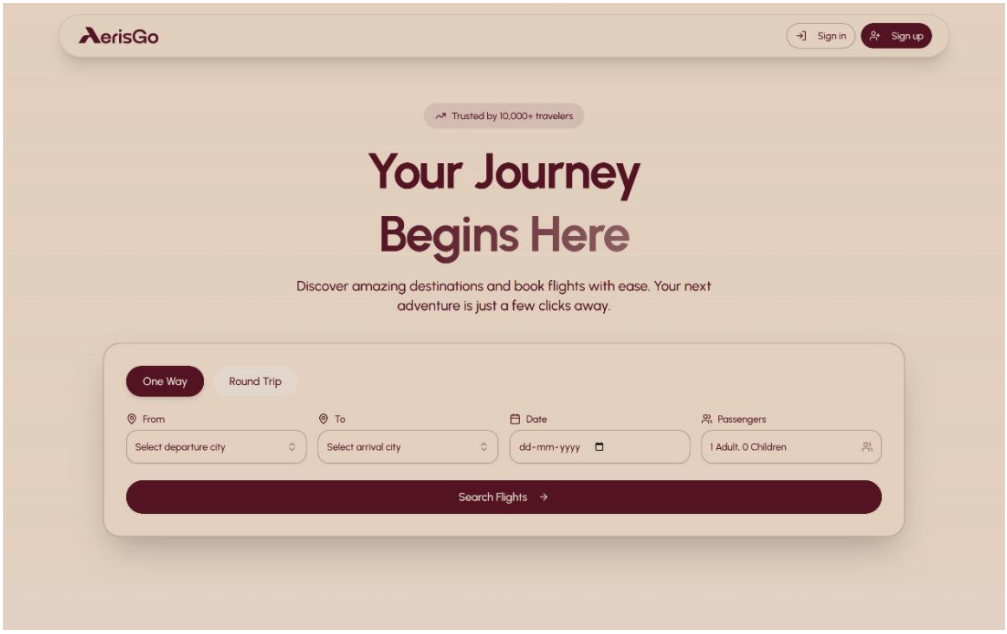


Profile Screen: In this screen user can edit their profile, change password, and manage documents and also check FAQ's.

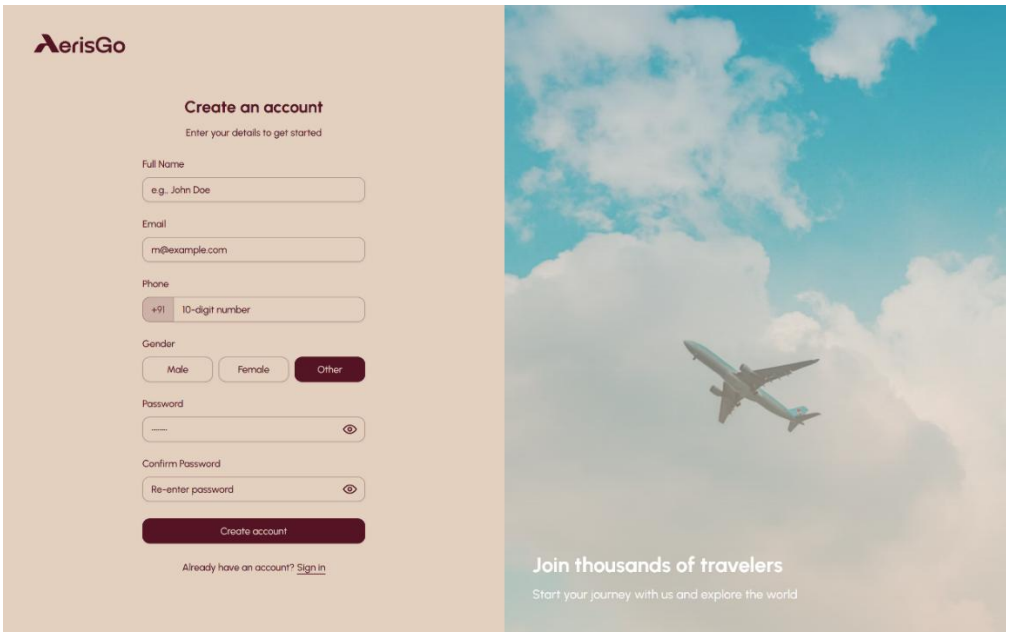


Notifications Screen: In this screen user can check their notifications and my pressing on card it will be mark as read.

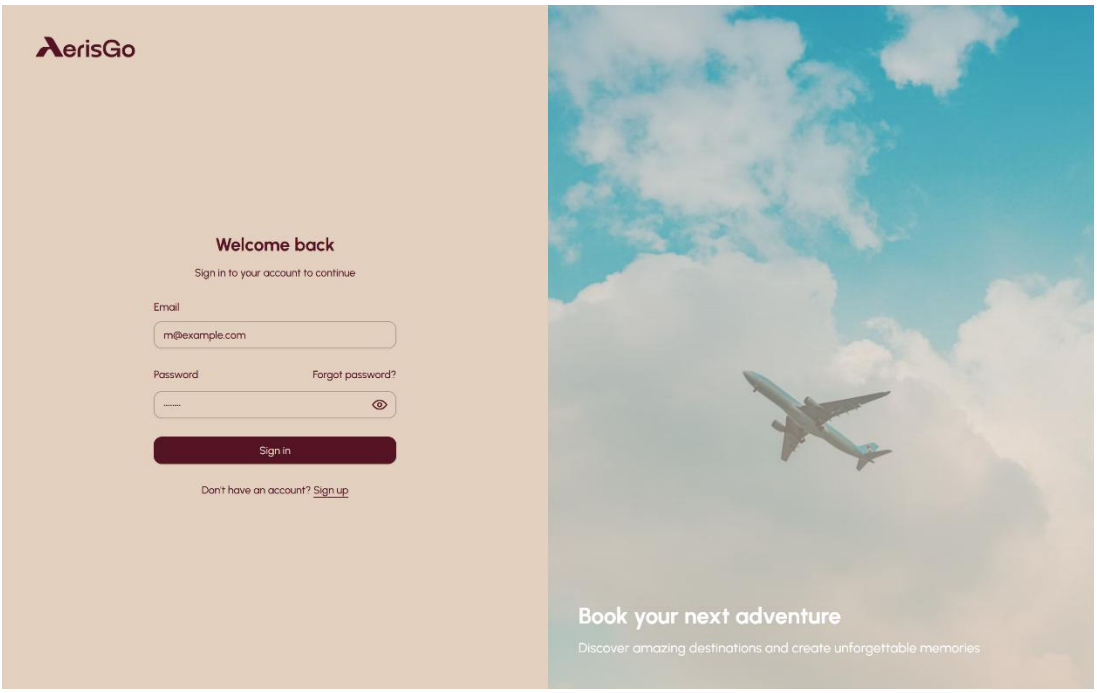
Web App



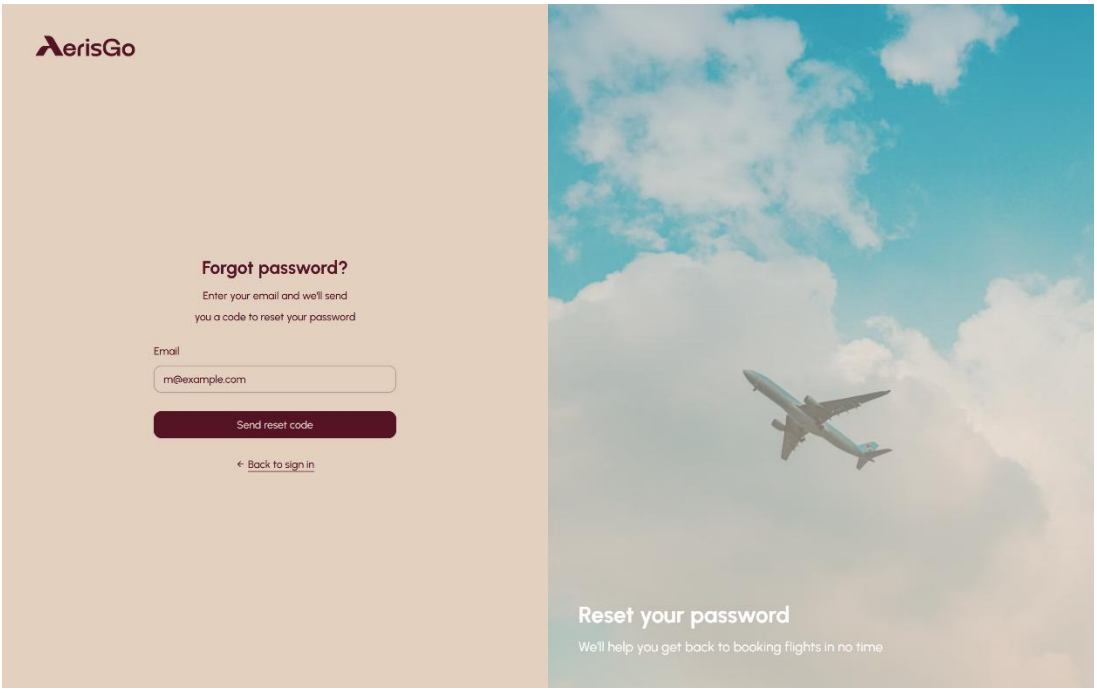
Landing Page : This offers a quick introduction and navigates users to the main features of the app.



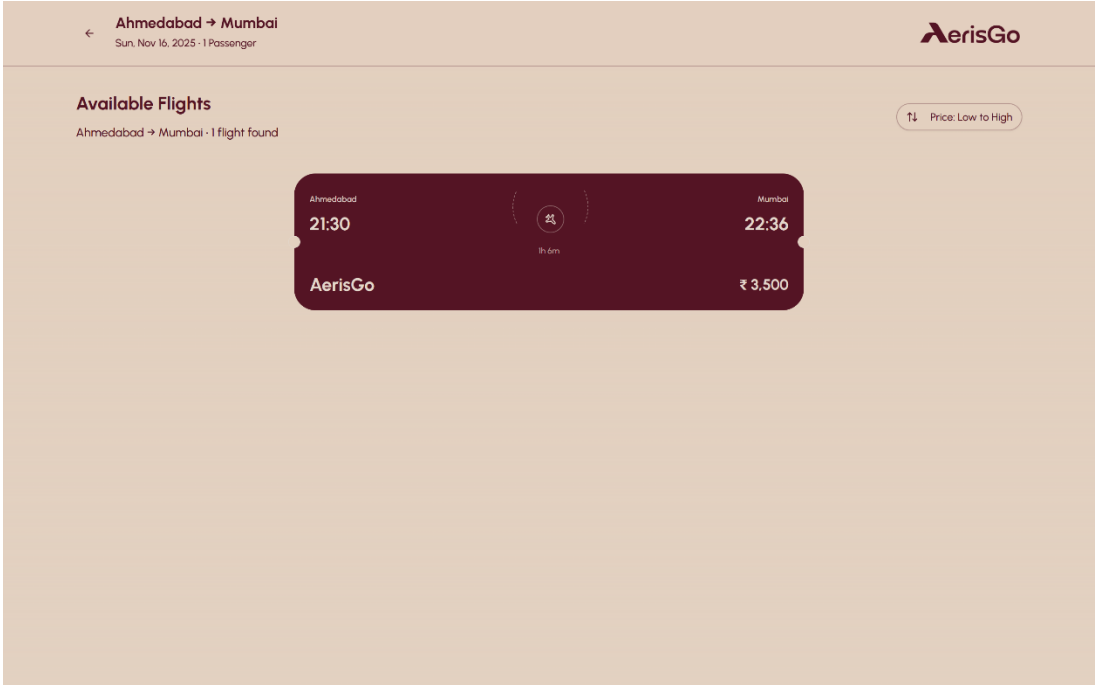
SignUp Page : The sign-up screen allows new users to create an account by providing required information.



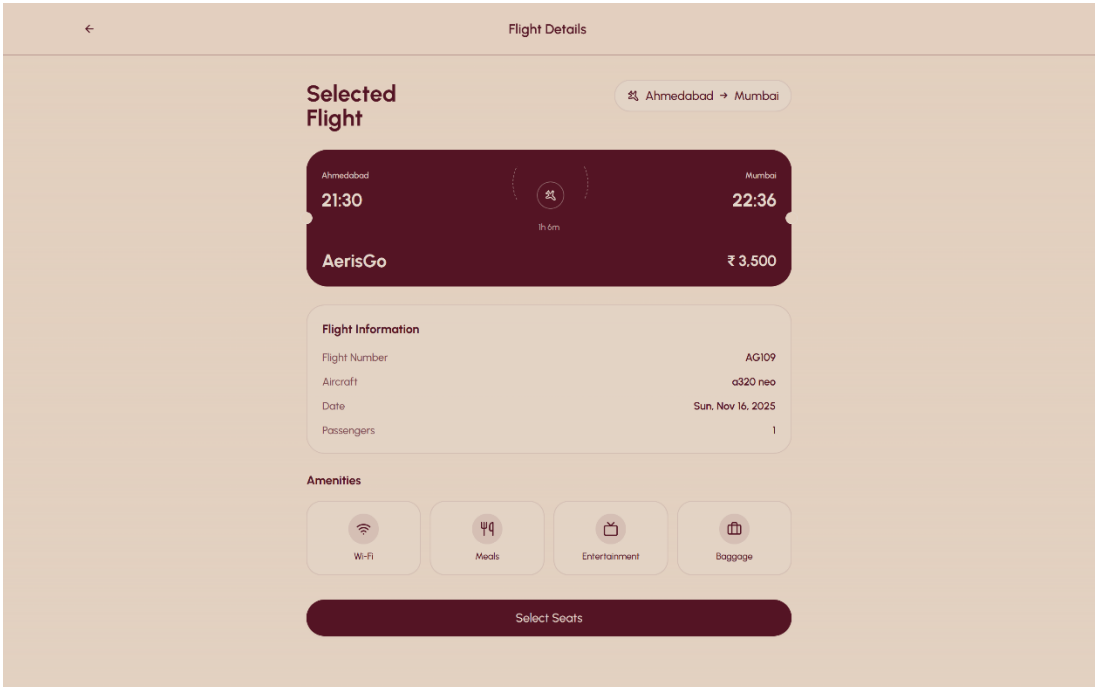
SignIn Page : The sign-in screen allows registered users to securely access their accounts.



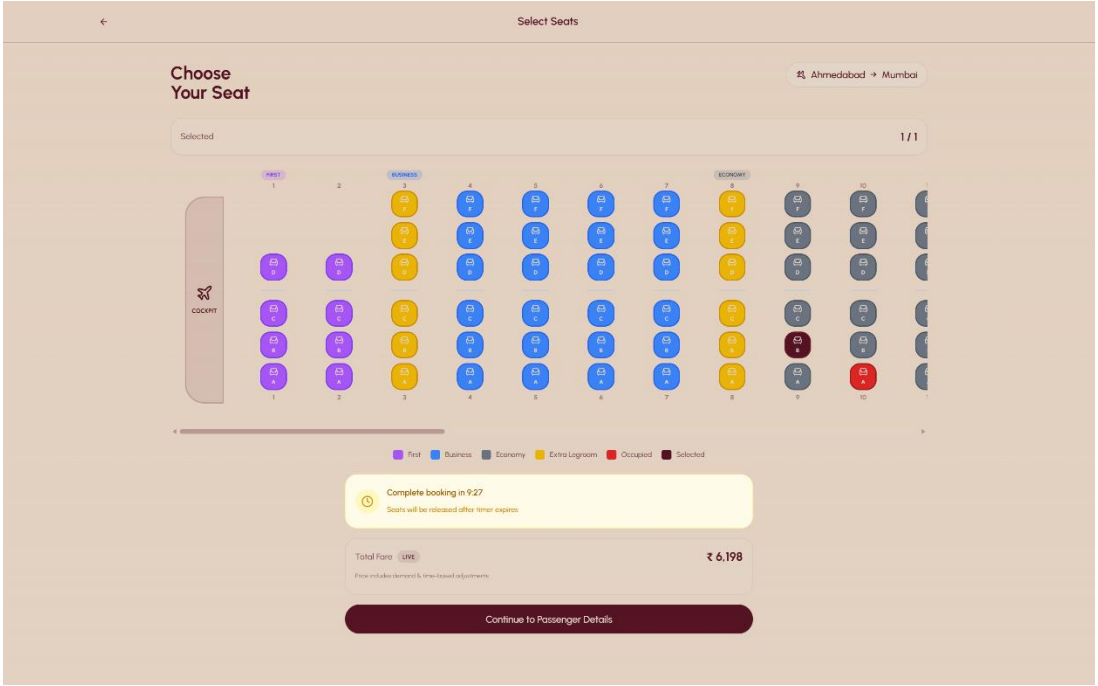
Forgot Password Page : The forgot-password screen allows registered users to securely reset their account password.



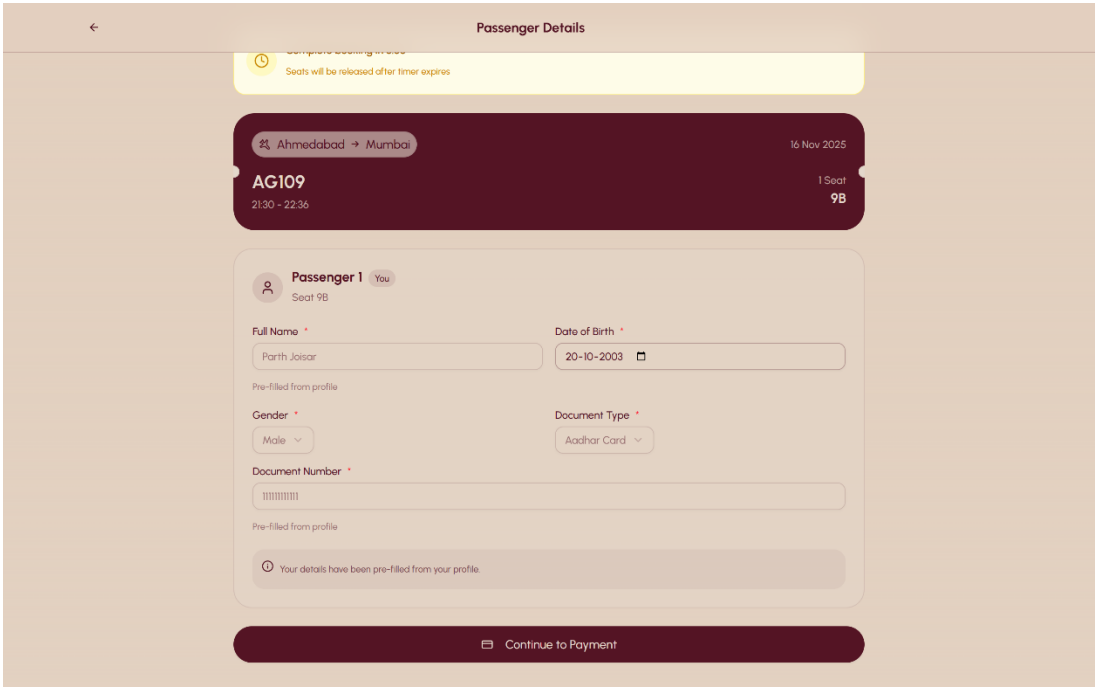
Search Result Page : The search result screen displays a list of available flights based on the user's search criteria.



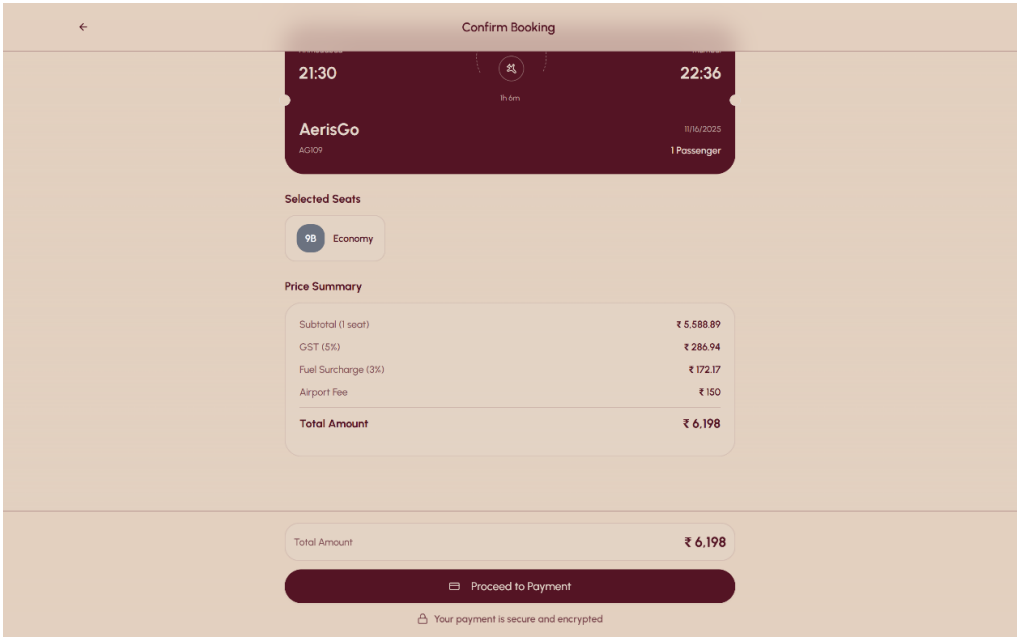
Flight Details Page : The flight details screen displays flight details like flight number, aircraft, date and amenities.



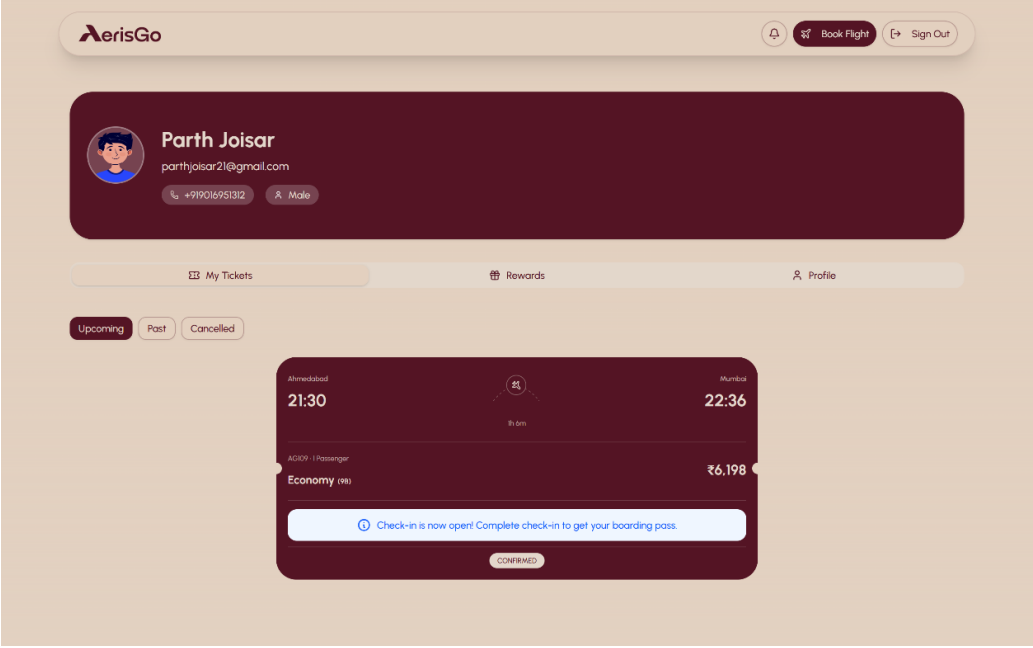
Seat Select Page : In this screen passenger can select their own choose of seats with real-time feature in it.



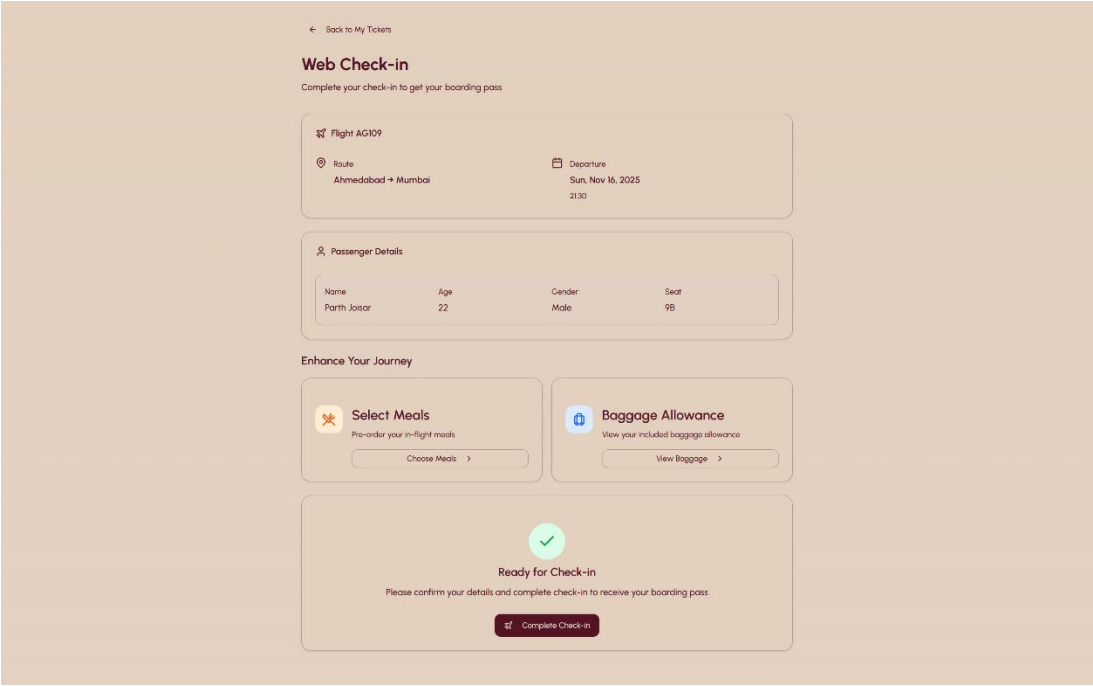
Passenger Details Page : The passenger details screen displays a form to fill passenger details.



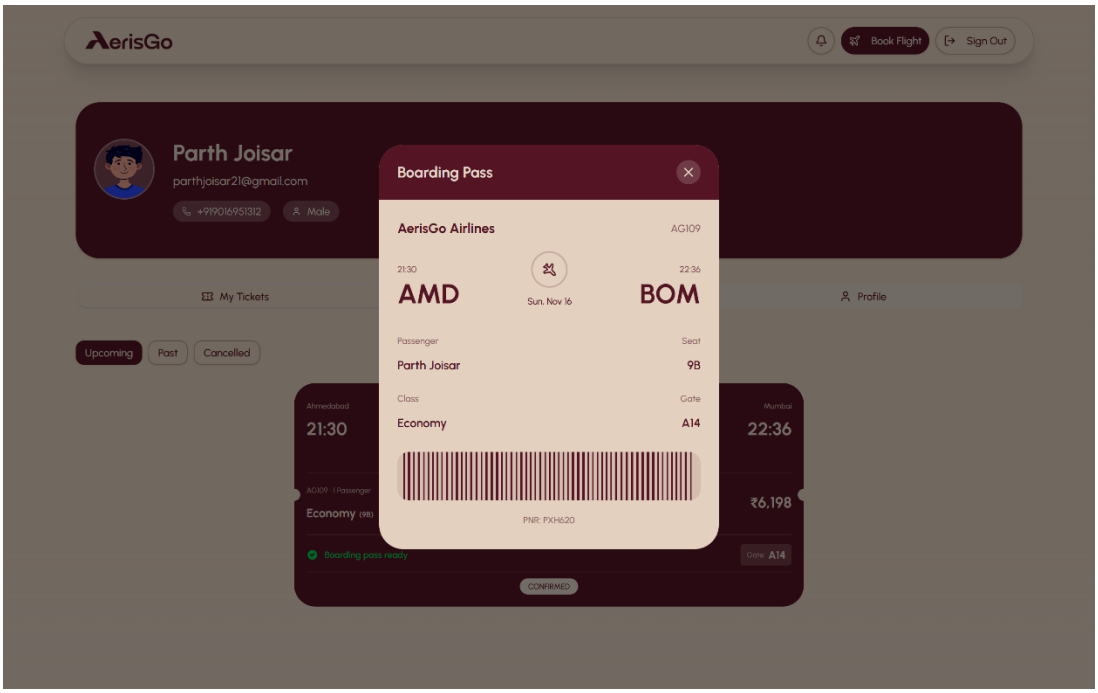
Confirm Booking Page : This is last screen and final screen of booking flow it displays the price summary and all other flight, seat details after this it redirects to payment model.



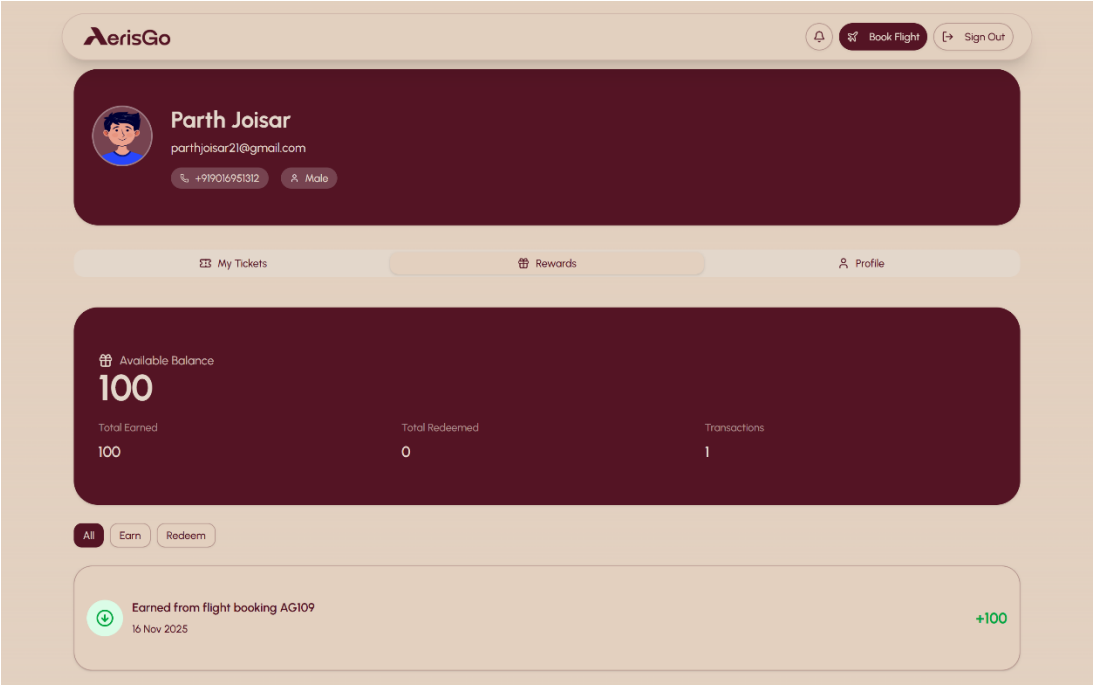
Tickets Page : In this screen user can check there upcoming tickets, past and cancelled and my pressing on ticket it shows details and option to check-n or cancel the ticket.



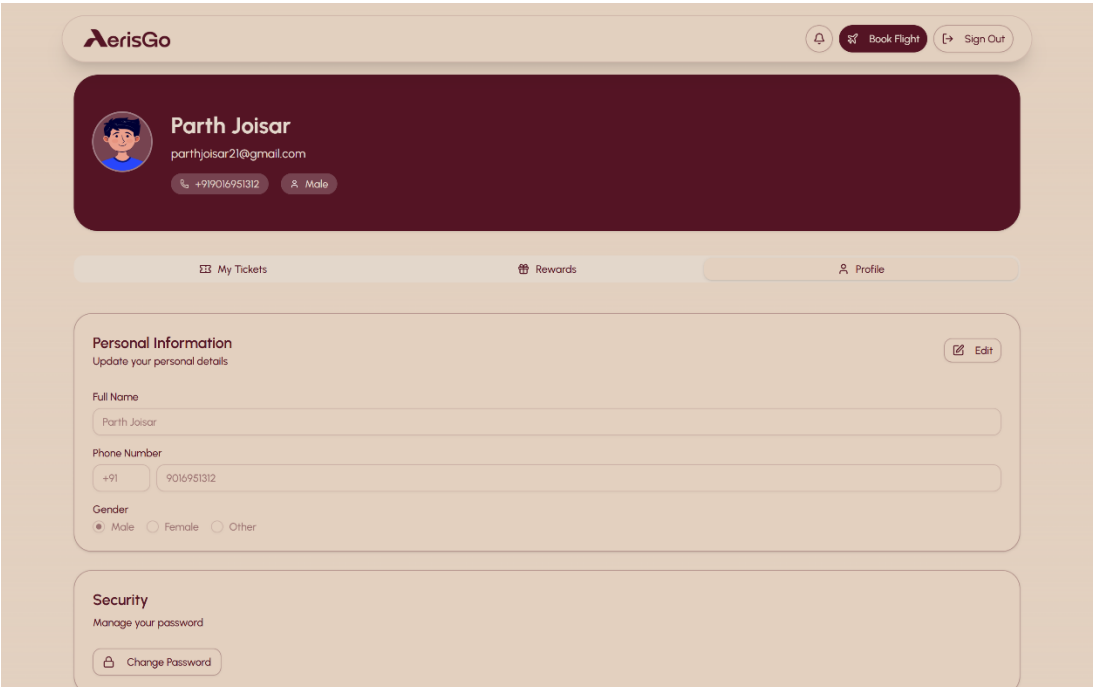
CheckIn Page : In this screen passenger can check-in and select their meals and also view baggage allowance.



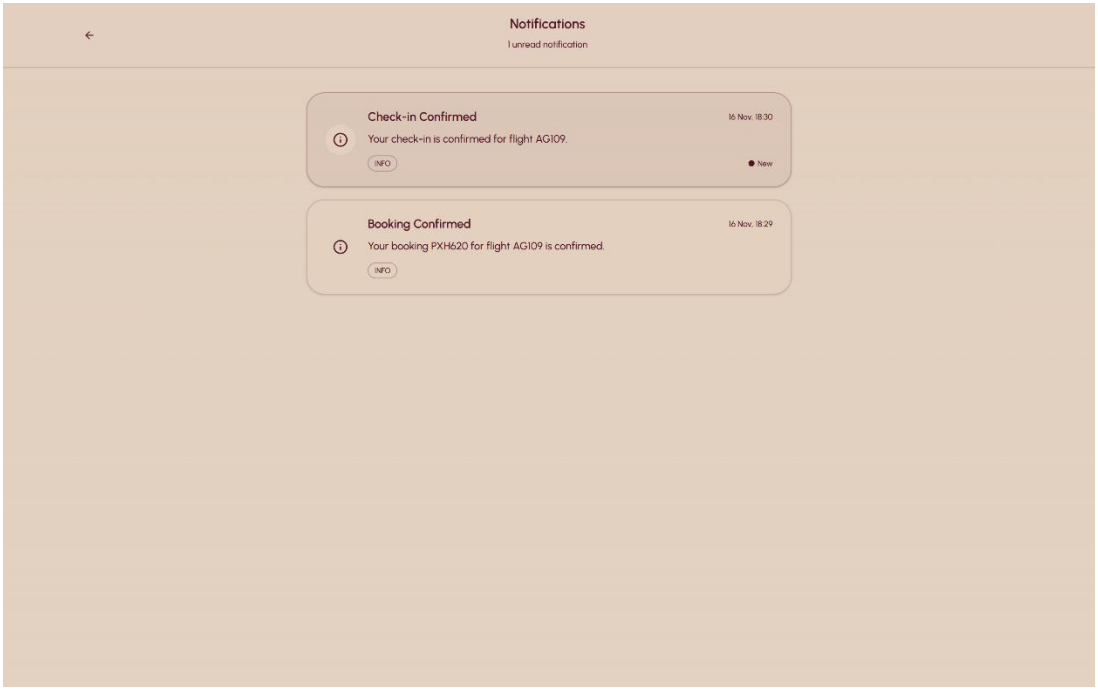
Boarding Pass Model: This is passengers boarding pass and they can see all the details like gate, class, PNR and etc.



Rewards Page: In this screen user can see their current balance of points and history.

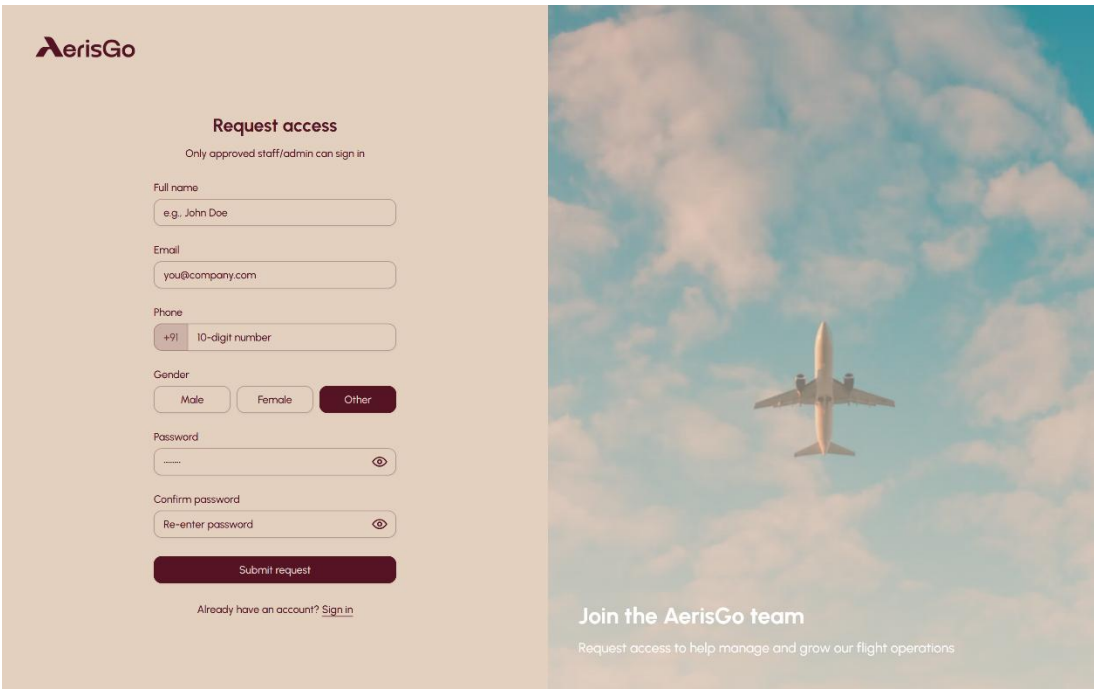


Profile Page: In this screen user can edit their profile, change password, and manage documents and also check FAQ's

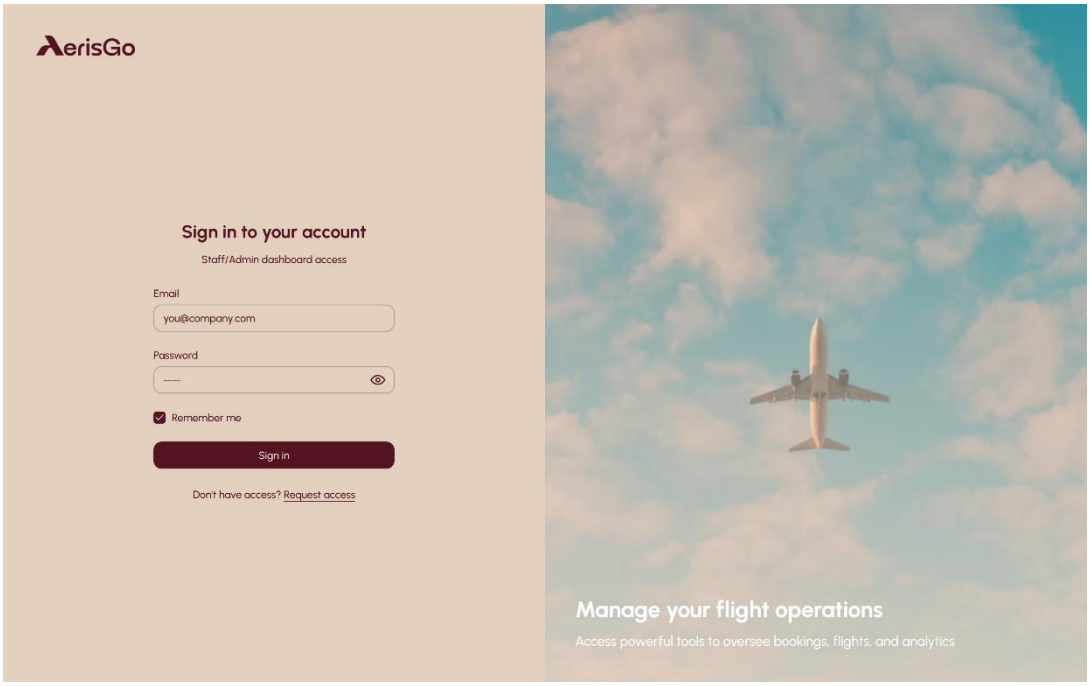


Notifications Page: In this screen user can check their notifications and my pressing on card it will be mark as read.

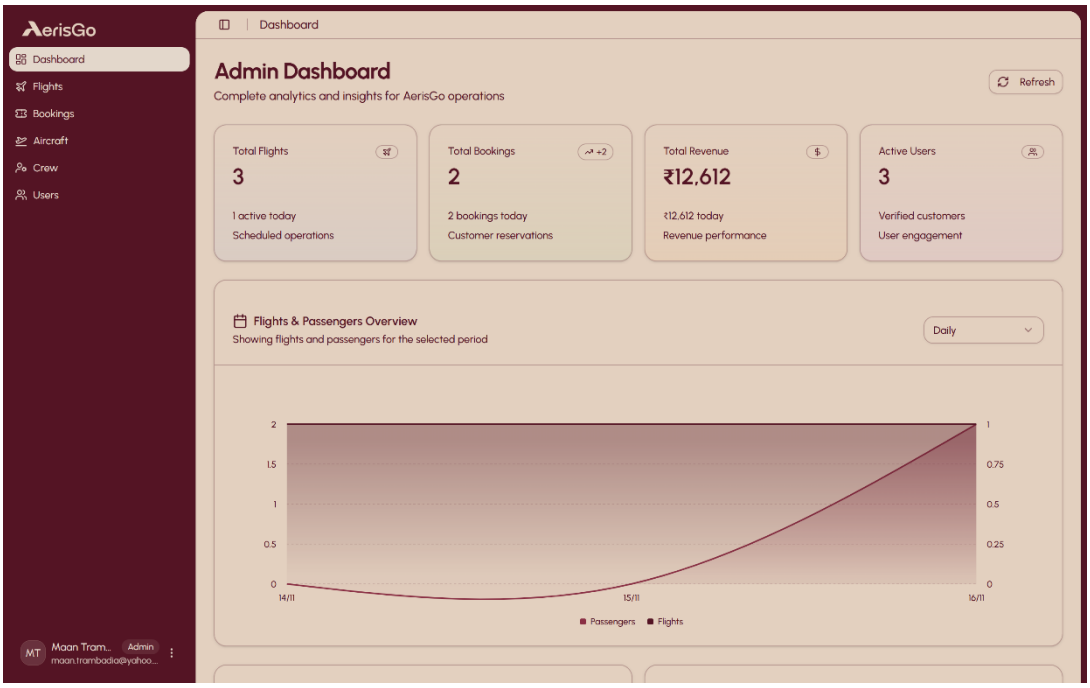
Web Dashboard



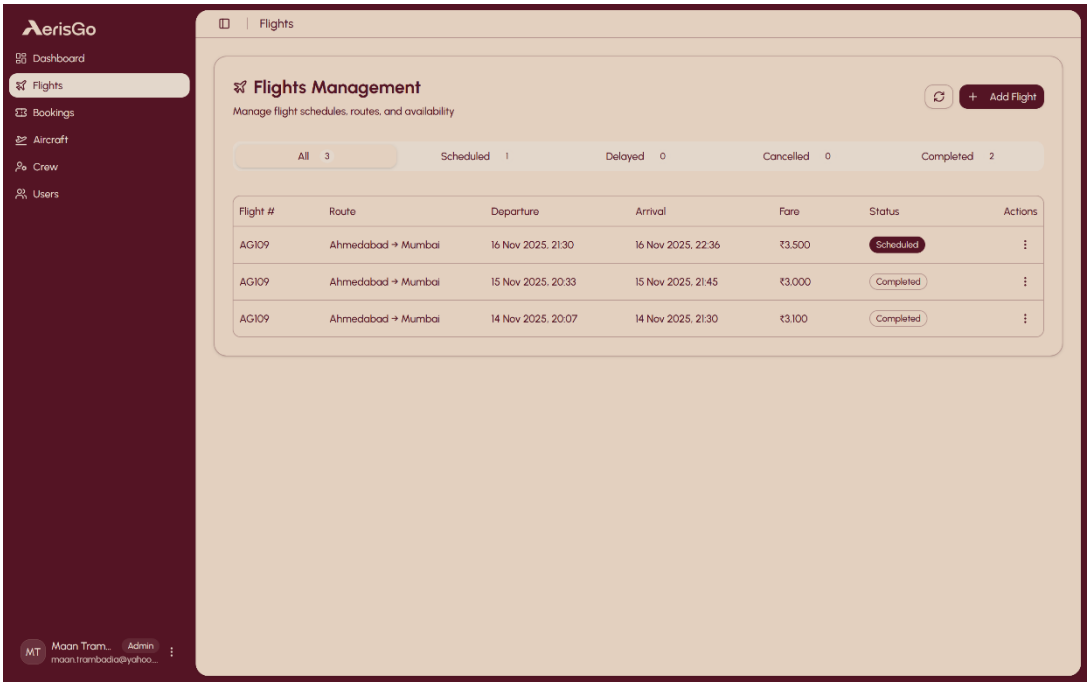
Request Access Page : The request access page allows staff members to submit a request for authorized access to the system.



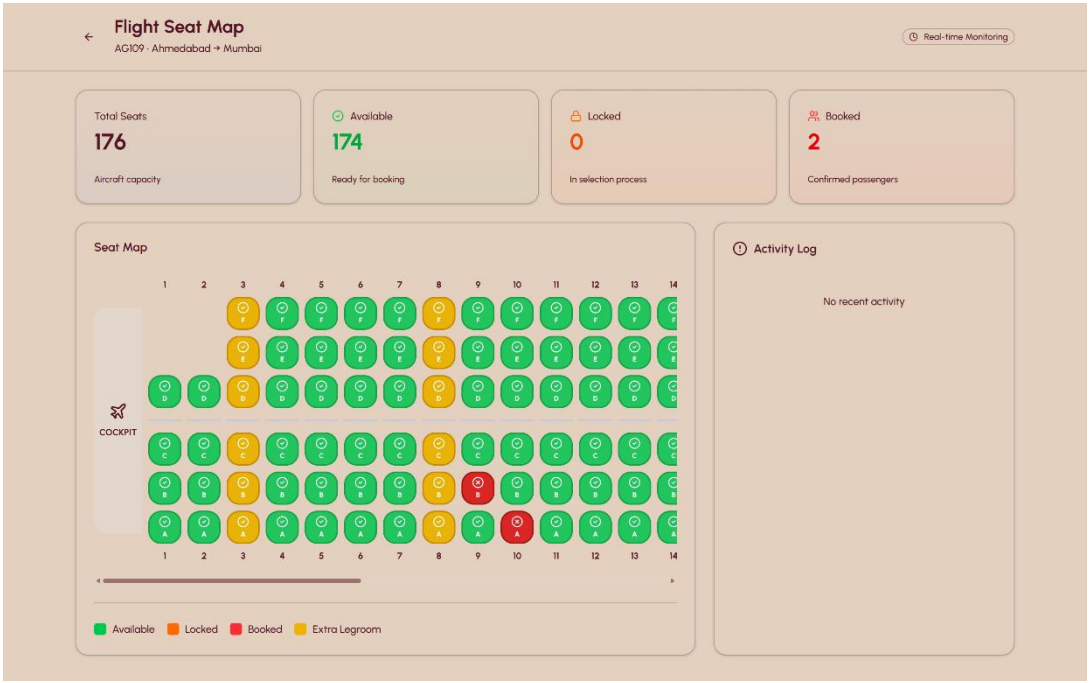
SignIn Page: The staff and admin can sign-in here



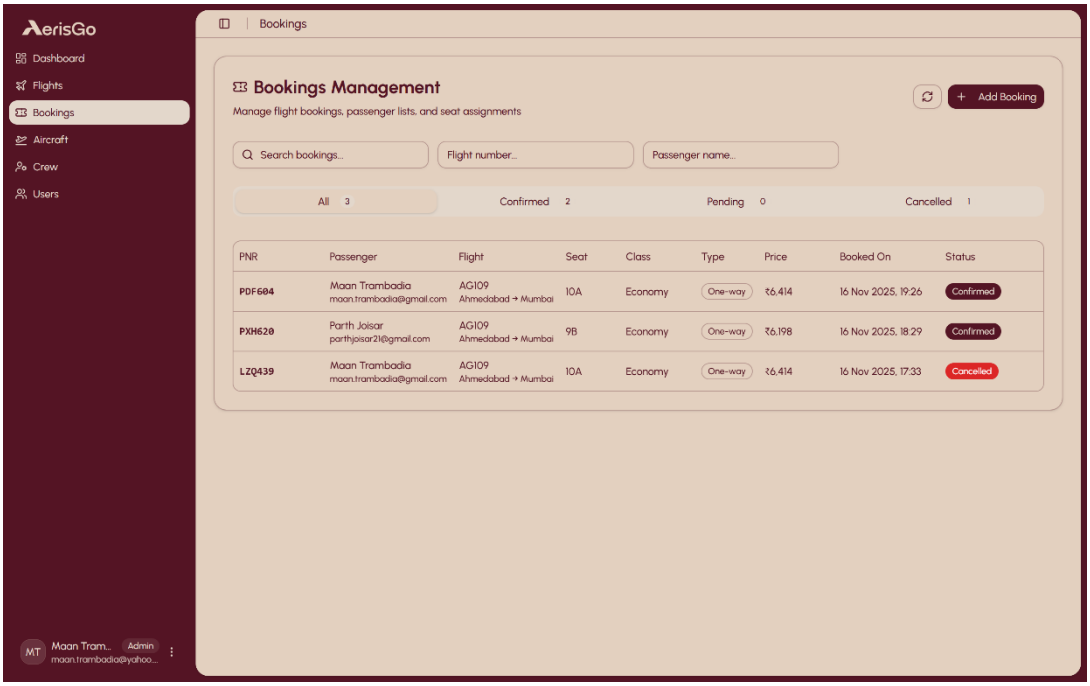
Dashboard Page: The admin dashboard provides a centralized interface for managing and monitoring system operations.



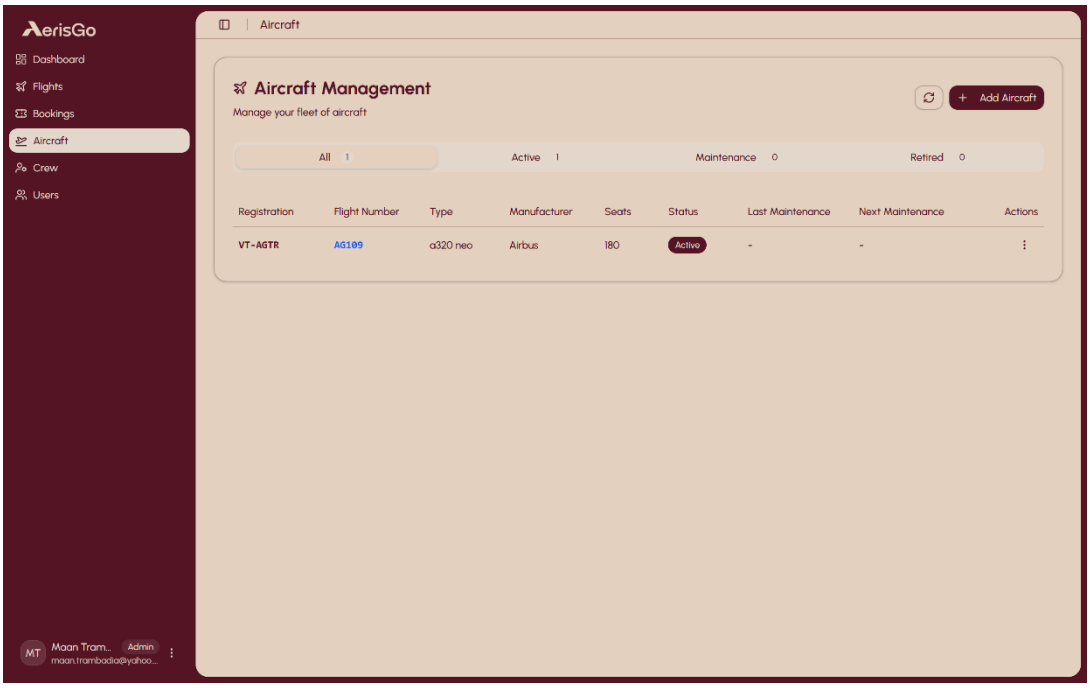
Flights Page: The flights page allows admins to create, read, update, and delete flight records within the system.



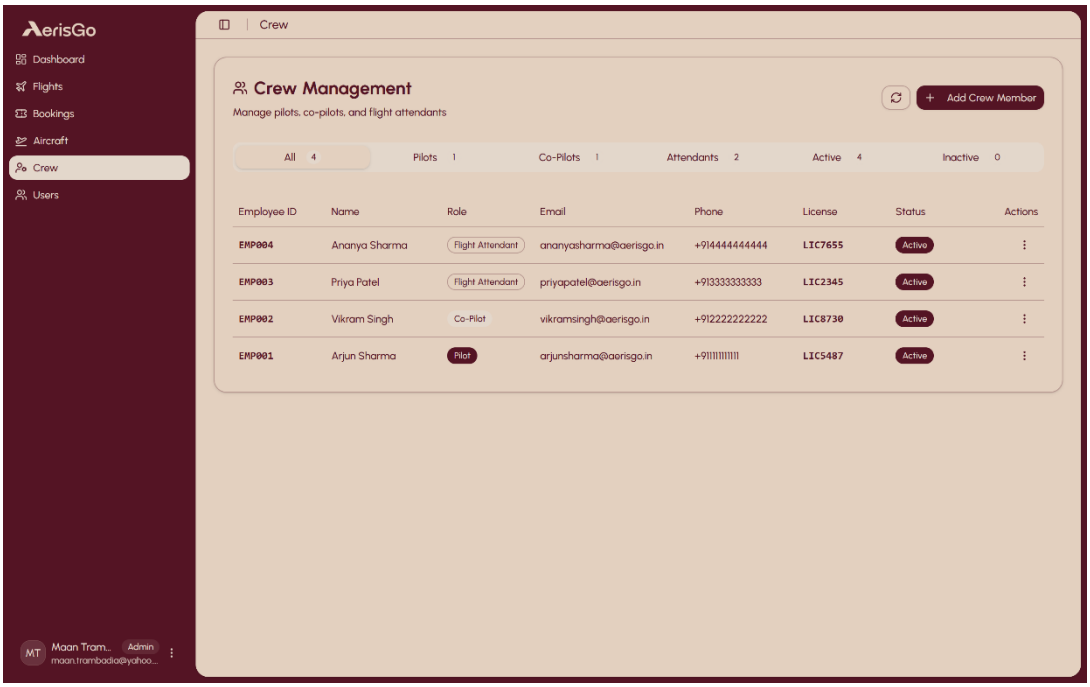
Flight Seat Map Page: This page provides real time seat map view of the flight selected.



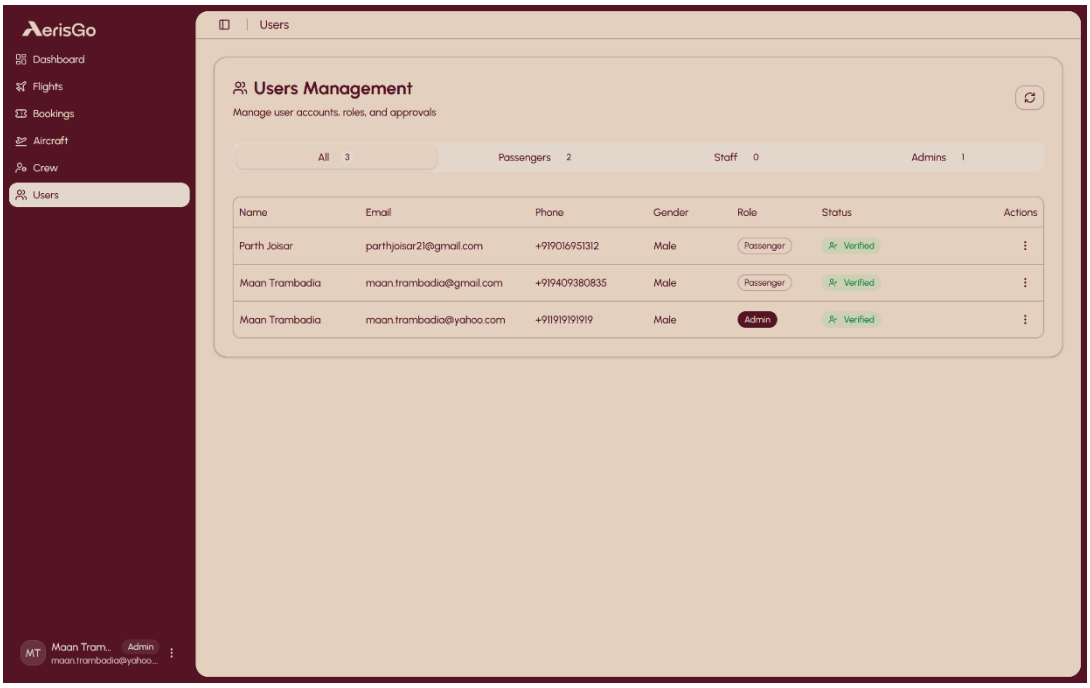
Booking Page: This page provides admin and staff role to manage the bookings and CURD operations on it.



Aircraft Page: This page provides admin role to manage the aircrafts.

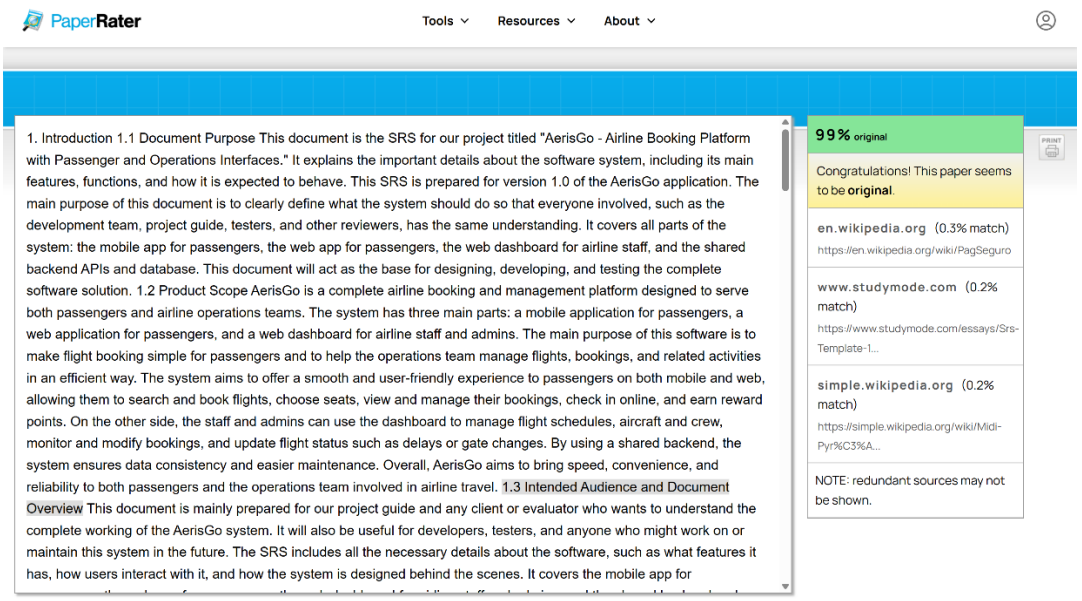


Crew Page: This page provides admin role to manage the crew.

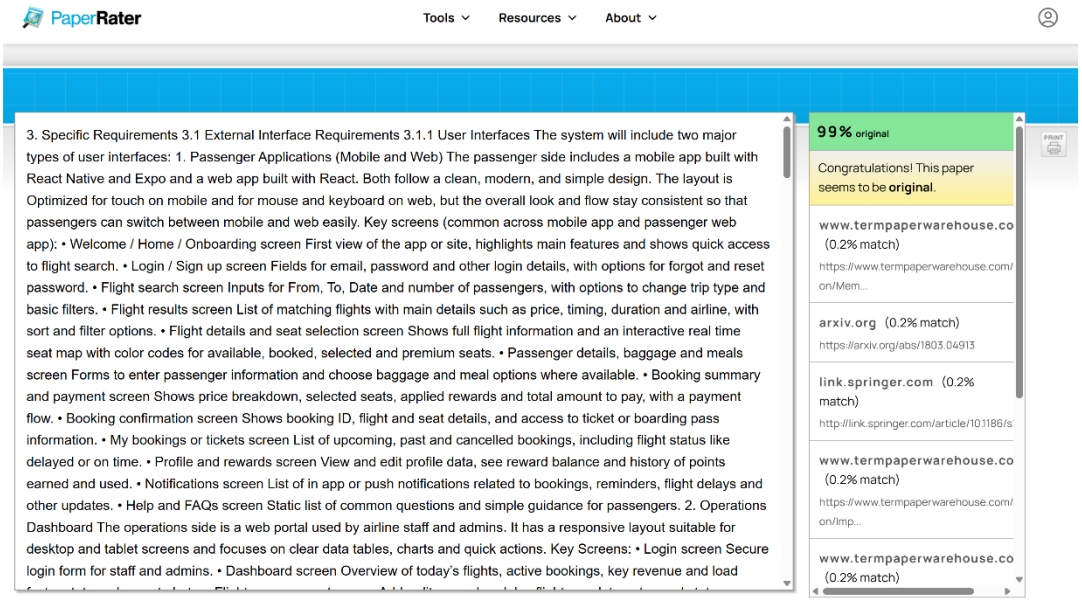


Users Page: This page provides admin role to manage the users.

Appendix C – Plagiarism Report



1. Section-1 and Section-2 Plagiarism Report



2. Section-3 and Section-4 Plagiarism Report