MARWADI UNIVERSITY
FACULTY OF DIPLOMA STUDIES
COMPUTER ENGINEERING DEPARTMENT

AerisGo

Prepared by:

VANSH LAKHANI : 92300938064
MAAN TRAMBADIA : 92300938072
ESHA BHADJA : 92300938084
PARTH JOISAR : 92300938087
ADITYA PANERI : 92300938089

Guided by:

Prof. RUSHI RAVAL

SEMESTER : 6

Project Review - 4

CONTENT

- Abstract
- Aim / Objectives
- Specific Requirements(Software/Hardware)
- System Design(Diagrams)
- Reference and Sources
- Summary

Abstract

AerisGo is a complete airline management system that includes a passenger mobile app, a passenger web app, and an operations dashboard, all connected to a shared backend built with Node.js, Express.js, and MongoDB. Passengers can sign up, search flights, choose seats, book tickets, manage their profile, check in, view boarding passes, and use reward points through clean and user-friendly apps. Airline staff use the web dashboard to manage flights, aircraft, crew, seat maps, and bookings, and to update delays, cancellations, or gate changes in real time. With one shared backend, all data stays synchronized, reducing duplication and making the experience smoother for both passengers and staff.

AIM / Objective

The system aims to give passengers a smooth and user-friendly experience on both mobile and web, helping them search and book flights, choose seats, manage bookings, check in online, and use reward points. Staff and admins can use the dashboard to manage schedules, aircraft, crew, bookings, and update flight status like delays or gate changes. With a shared backend, the system keeps data consistent and easier to maintain. Overall, AerisGo aims to provide speed, convenience, and reliability for both passengers and the operations team.

Specific Requirements

1. **External Interface Requirements :-**

I. **Passenger App (Mobile & Web)**

- React Native + Expo (Mobile), React.js (Web)
- Clean, modern, consistent UI
- Key features:
  - light search & booking
  - Seat selection
  - Passenger details, baggage, meals
  - Booking summary & payment
  - Booking confirmation & e-ticket
  - My bookings
  - Profile & rewards
  - Notifications & help

Specific Requirements

II. **Operations Dashboard(React.js)**
- For airline staff (Admin & Staff)
- Desktop & tablet-friendly layout
- Key features:
  - Login & secure access
  - Dashboard overview (flights, bookings, alerts)
  - Flight management (add, edit, delay, cancel, gate updates)
  - Bookings management (filter, modify, cancel)
  - Seat assignment & real-time seat maps
  - Staff/user management
  - Notifications (email & in-app)
  - Reports & analytics (load factor, delays, revenue)

Specific Requirements

2. System & Communication Setup :-

I. Backend & Devices

- Common backend for both platforms
- Shared backend for all platforms
- Built using Node.js, Express, MongoDB
- Supports real-time updates (WebSocket/Polling)
- Secure API communication over HTTPS
- Hosted on a Linux VPS with Nginx and SSH
- Mobile apps communicate via HTTPS and use device APIs (storage, camera, notifications)

Specific Requirements

II. Notifications & Services

- Push: FCM (Android), APNs (iOS)
- Emails via SMTP
- Secure data handling & access control

Specific Requirements

3. Functional Requirements (Passenger App) :-

I. Passenger App Functions

- Sign up/login with email or phone
- Update profile details and upload documents
- Search flights with filters
- Seat selection and booking
- View and manage past & current bookings
- Earn and redeem reward points
- Receive notifications and alerts
- Access FAQs and help

Specific Requirements

4. Functional Requirements (Operations & Backend) :-

I. Operations Web Portal

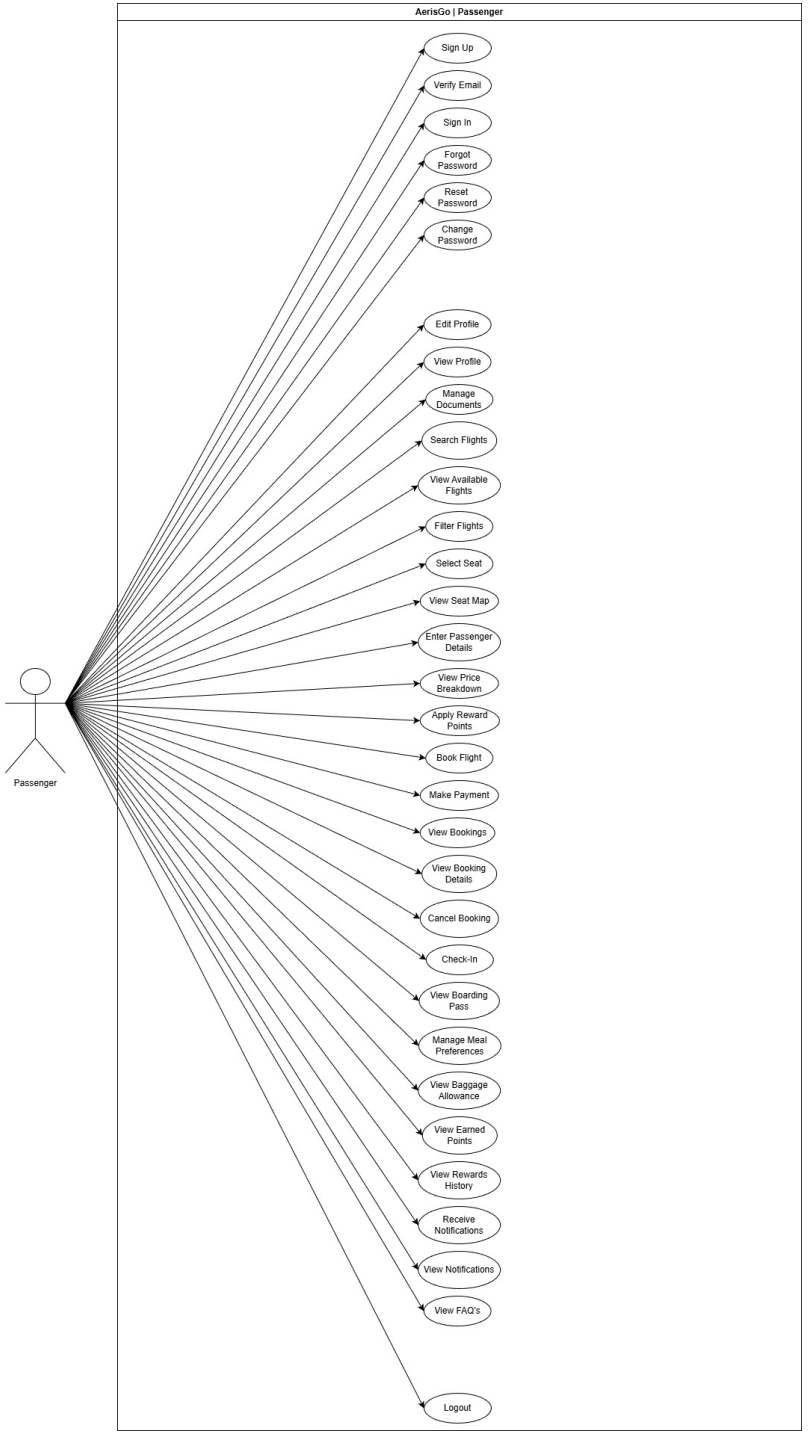
- Role-based login (Admin/Staff)
- Add/edit/cancel flights, mark delays
- View/modify bookings
- Assign/reassign seats
- Send push/email alerts
- View passenger, revenue, delay & load-factor stats

II. Shared Backend

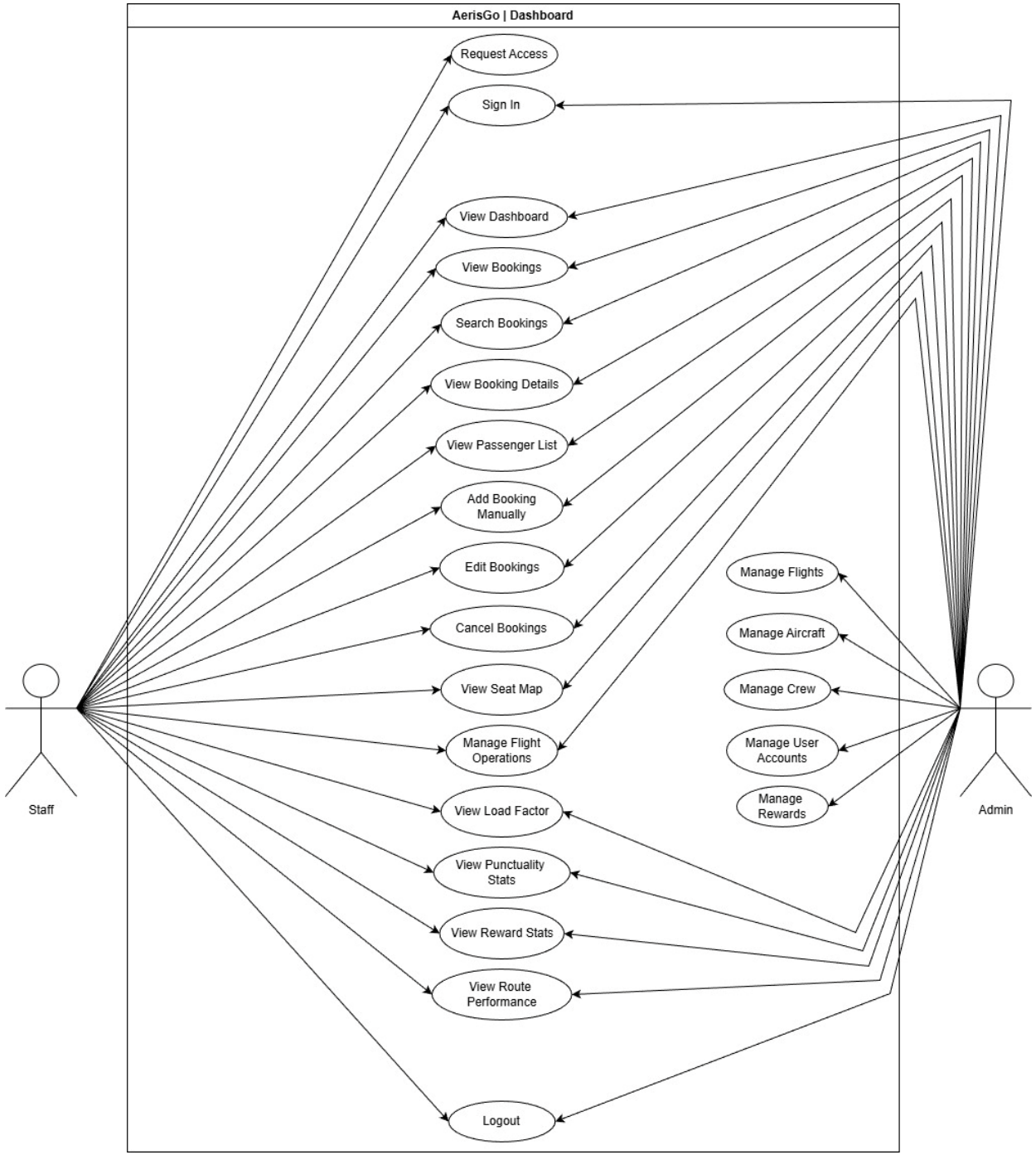
- REST APIs for app & web
- Real-time seat updates using WebSockets
- Secure user data handling
- Push notifications for flight updates and user actions

System Design(Diagrams)

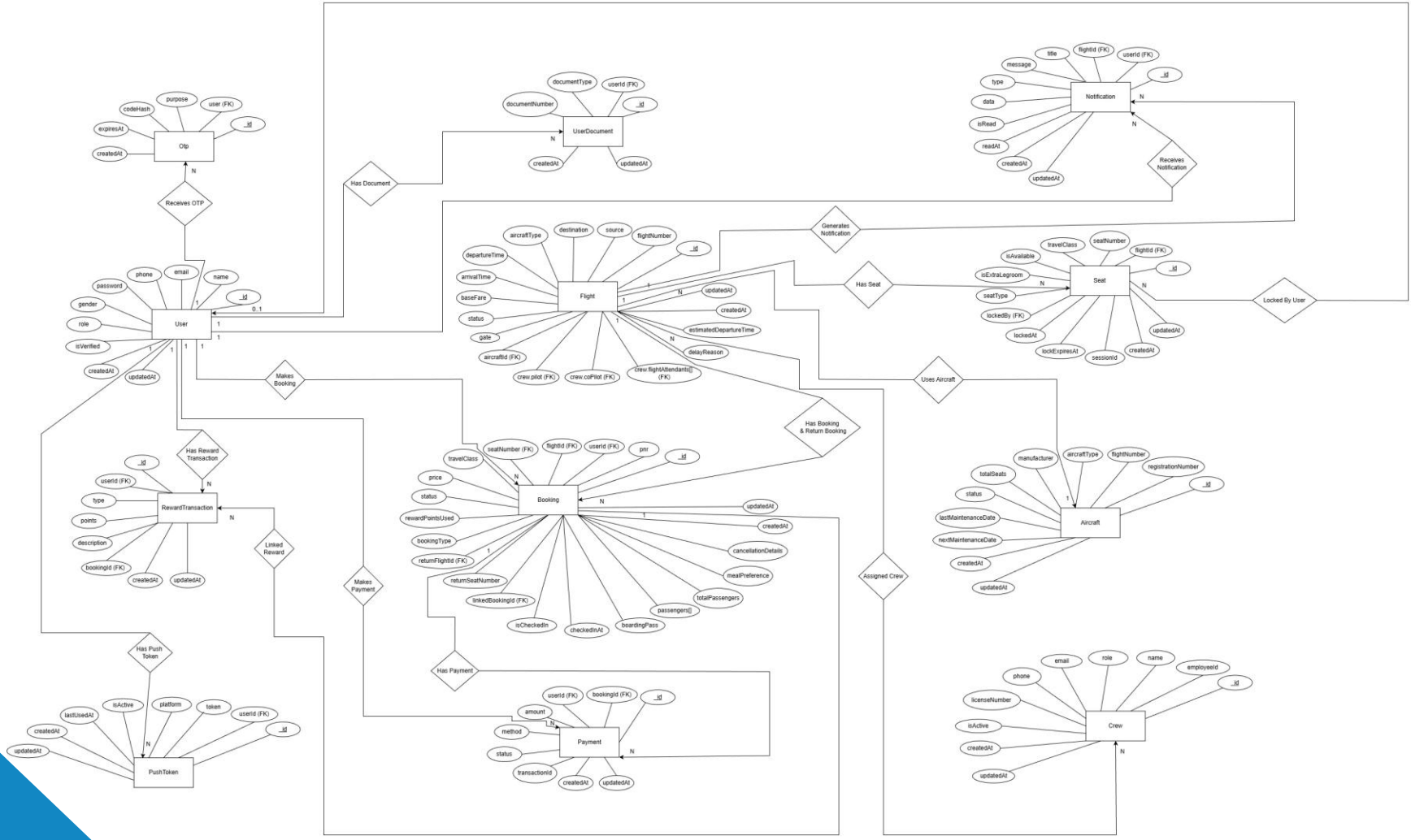
1.1. Passenger Use case Diagram:



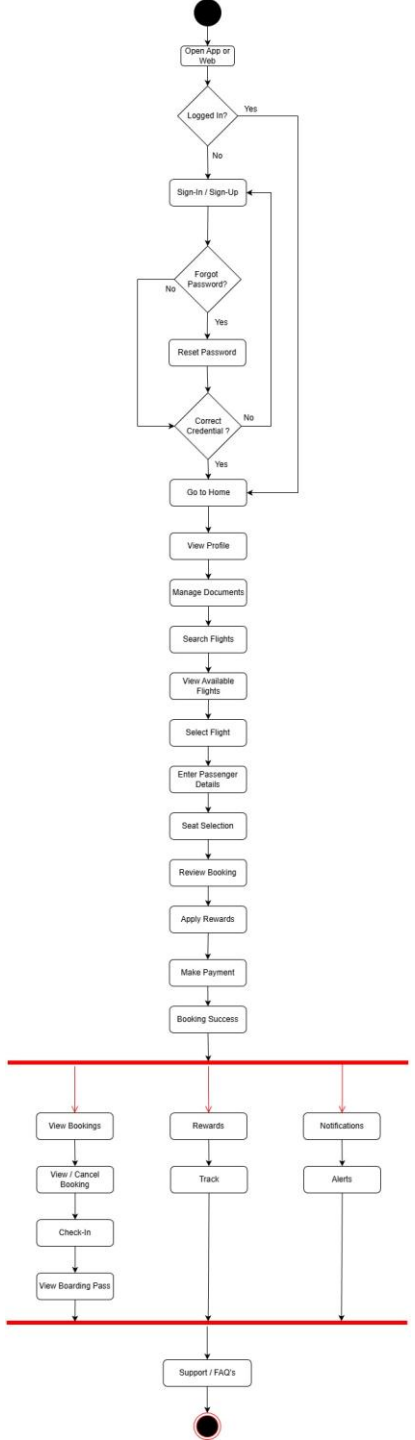
1.2. Dashboard Use case Diagram:



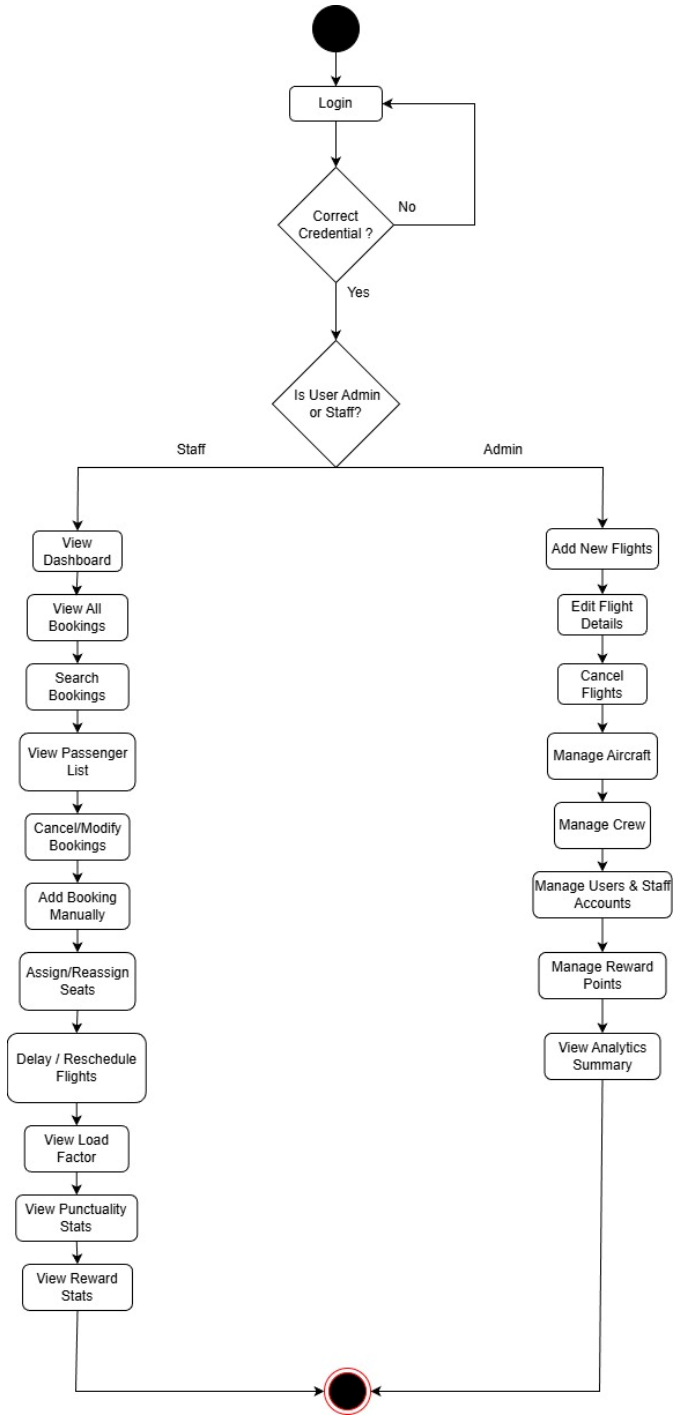
2.1. E.R Diagram:



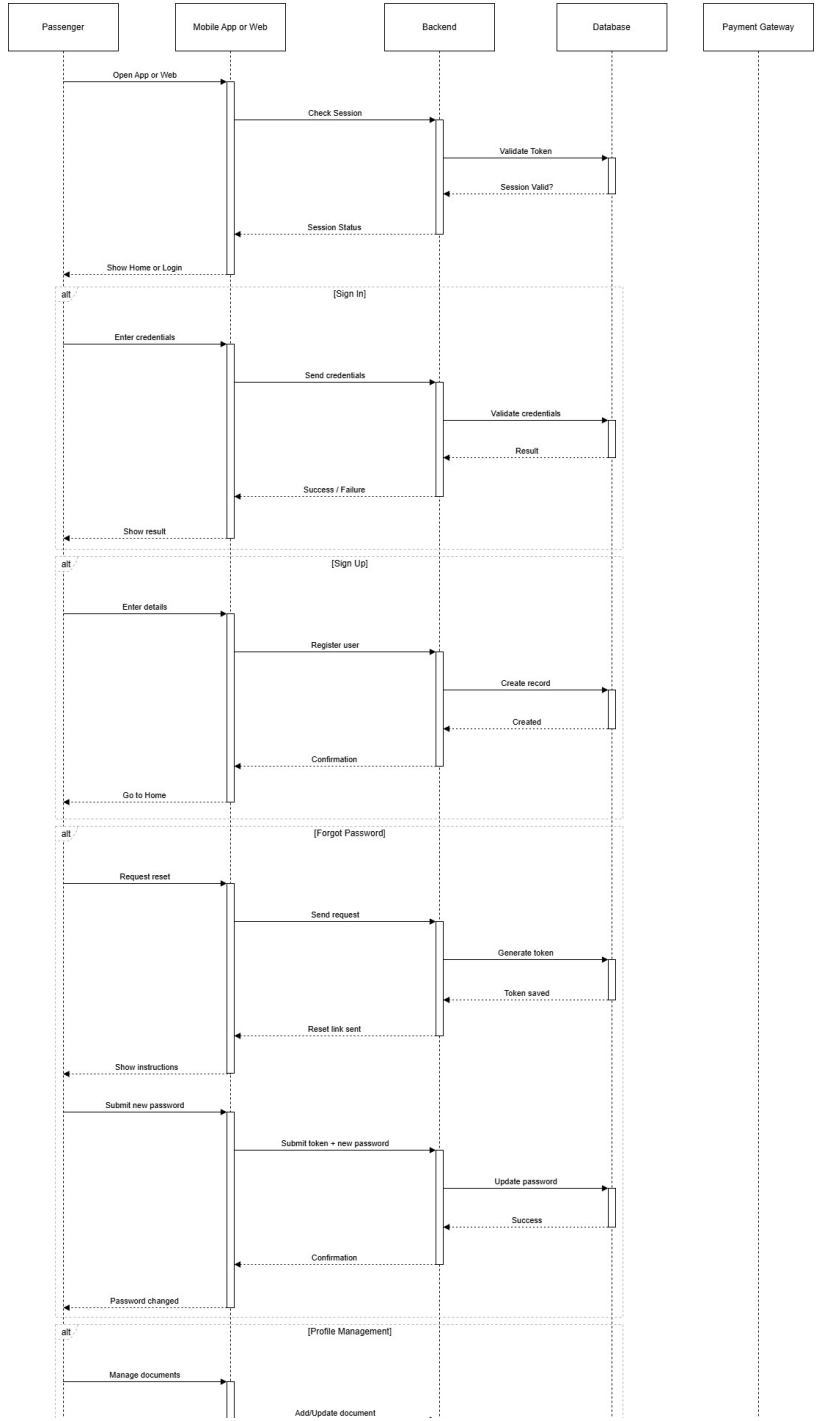
3.1. Passenger Activity Diagram:



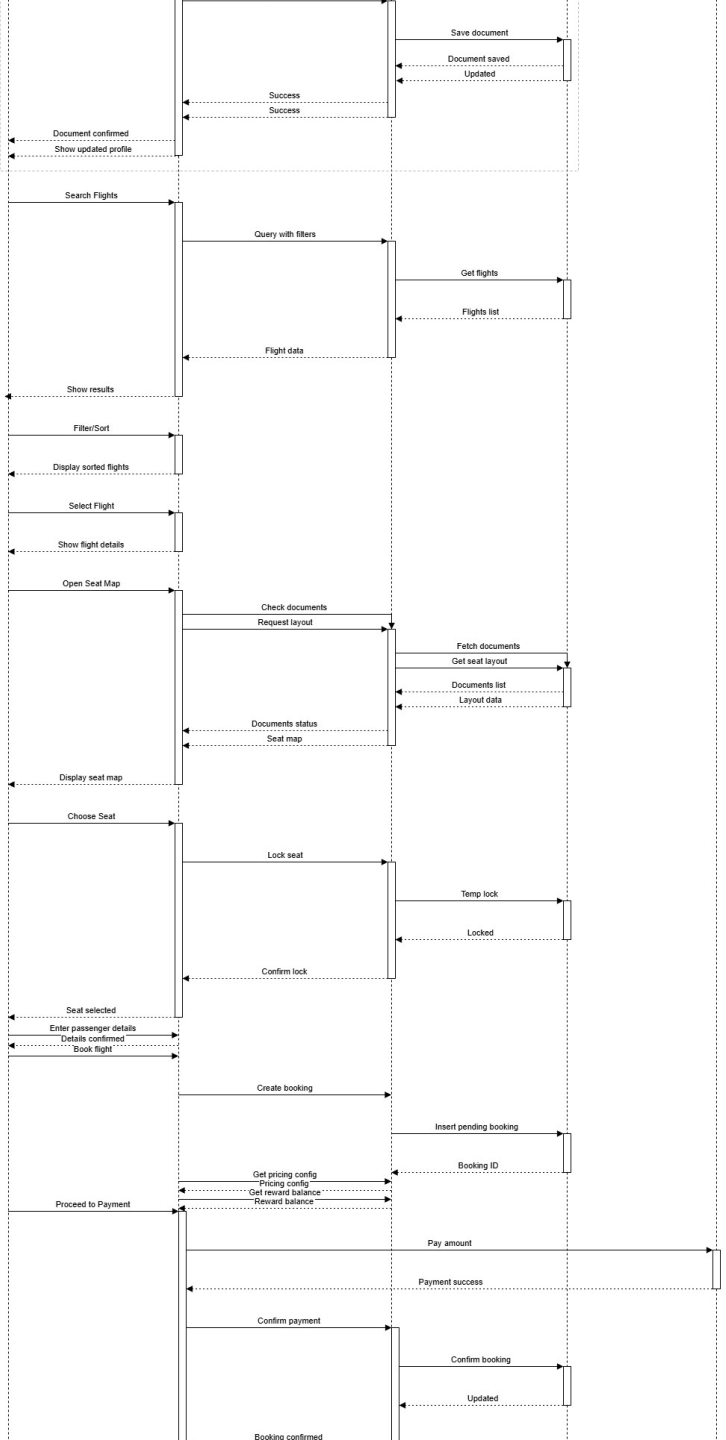
3.2. Dashboard Activity Diagram:



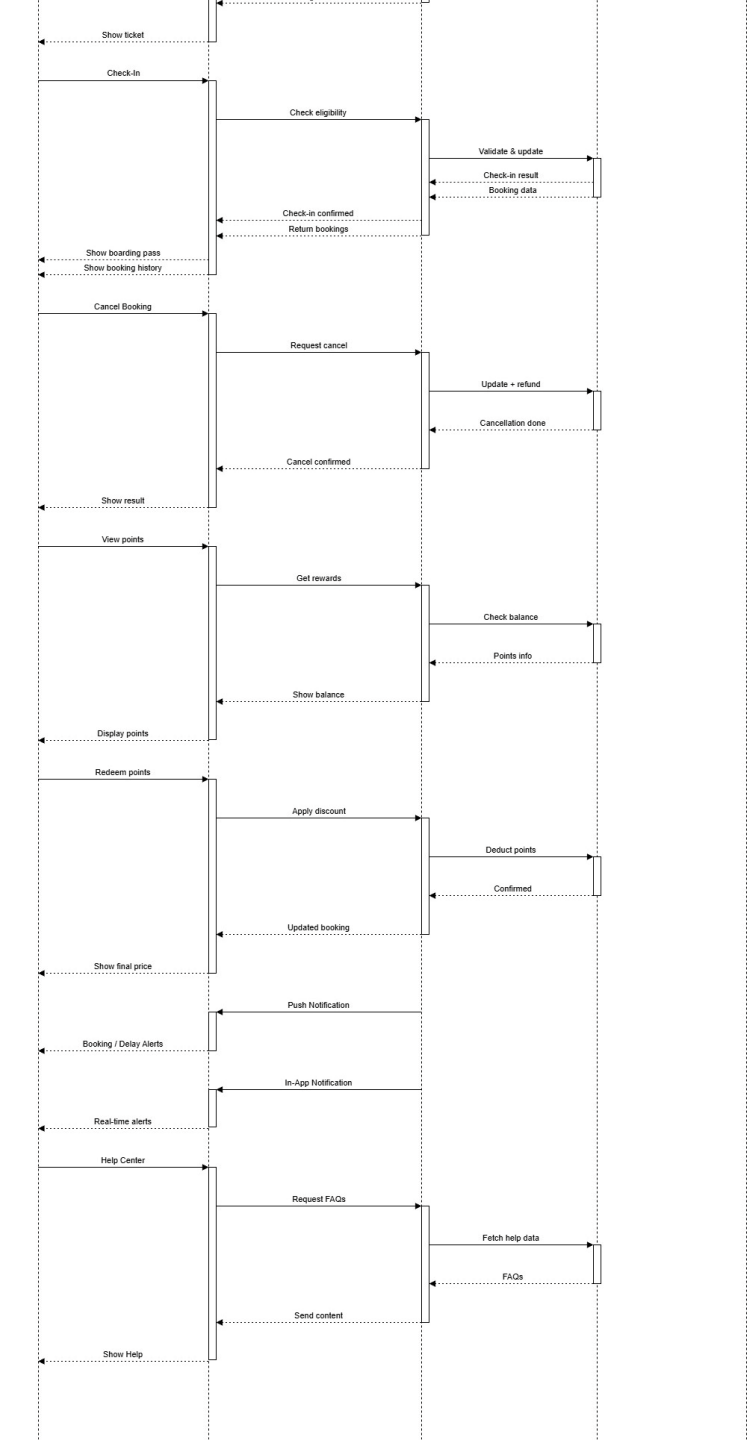
4.1. Passenger Sequence Diagram:



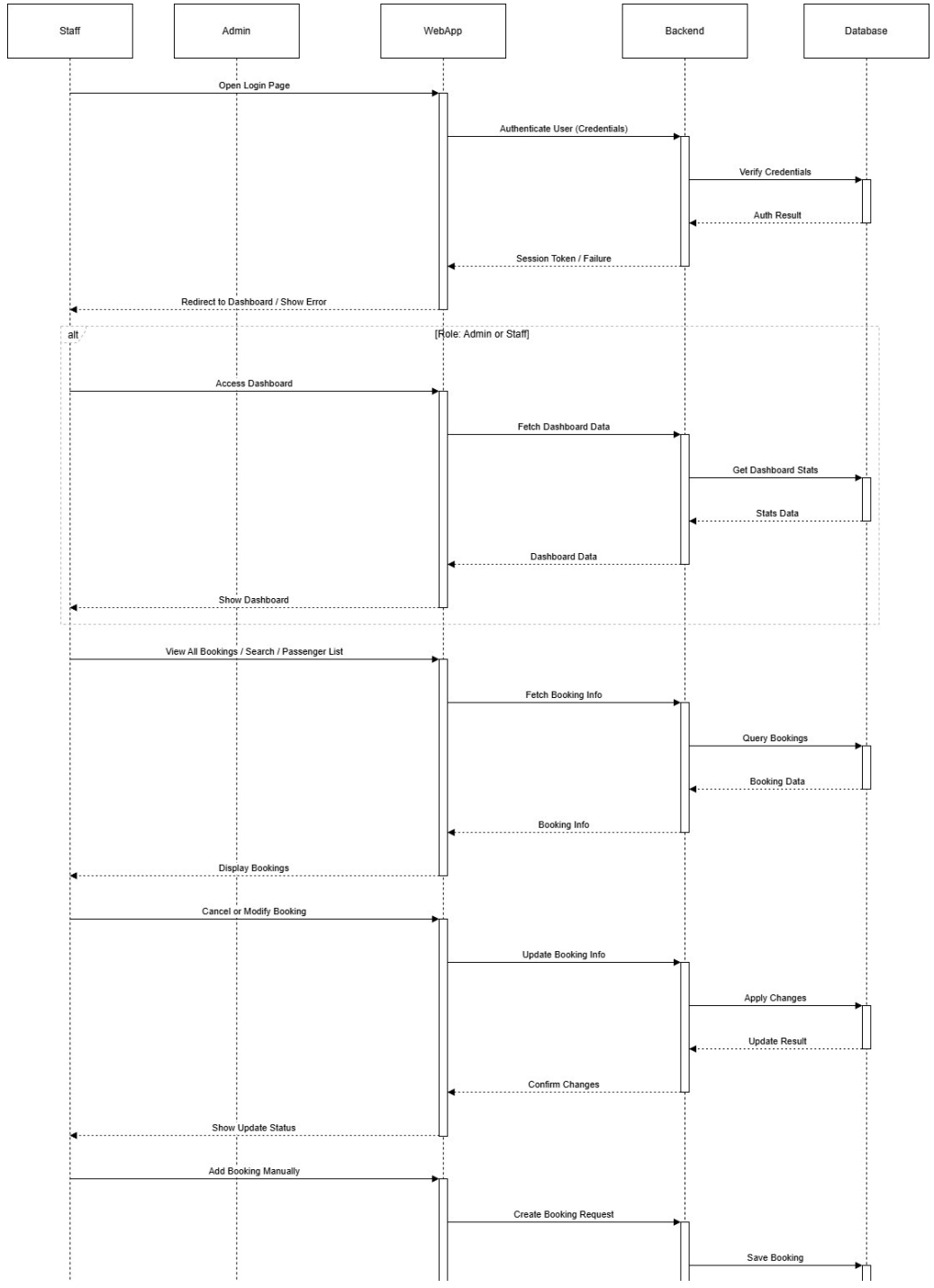
4.1. Passenger Sequence Diagram:



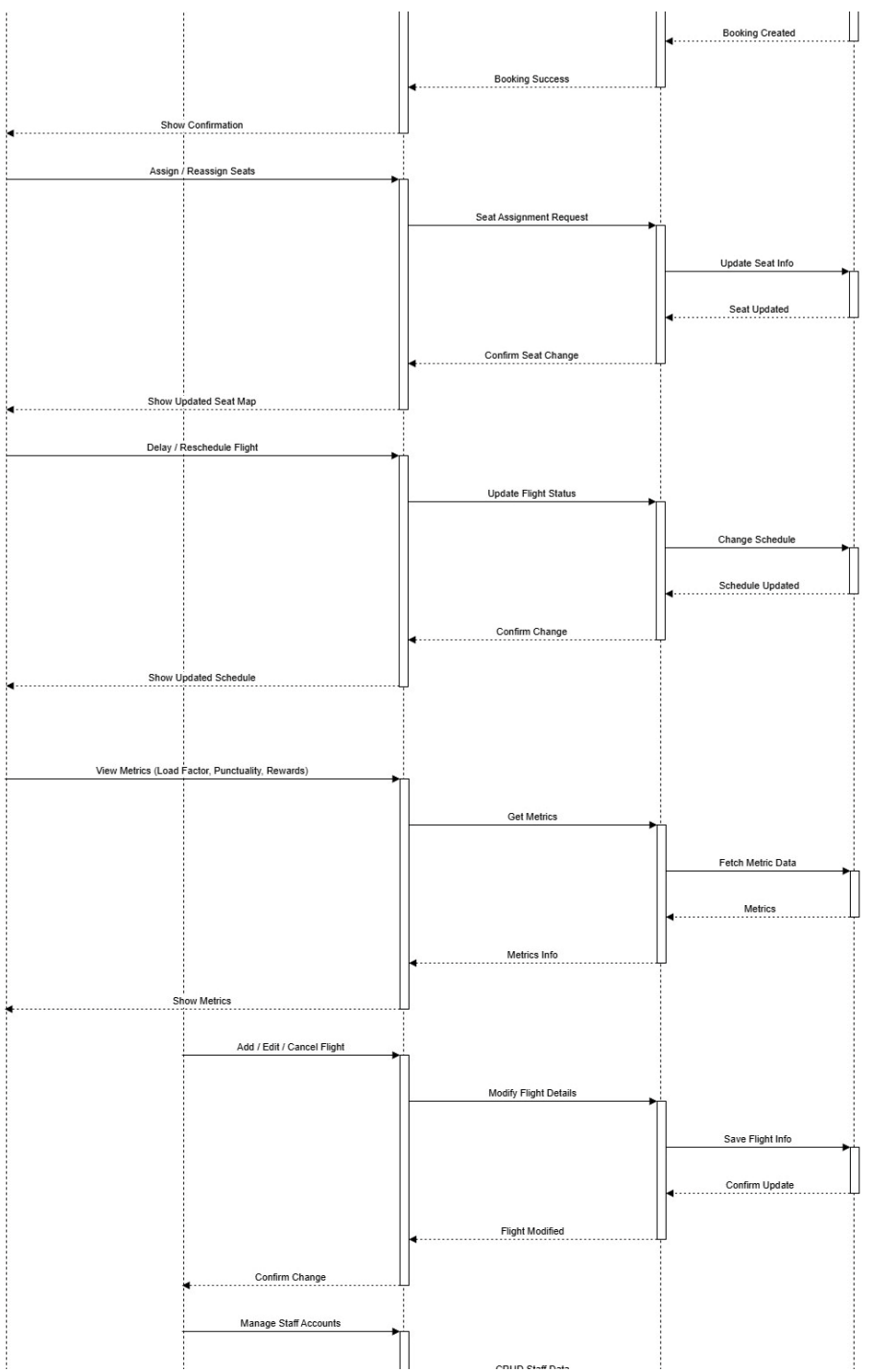
4.1. Passenger Sequence Diagram:



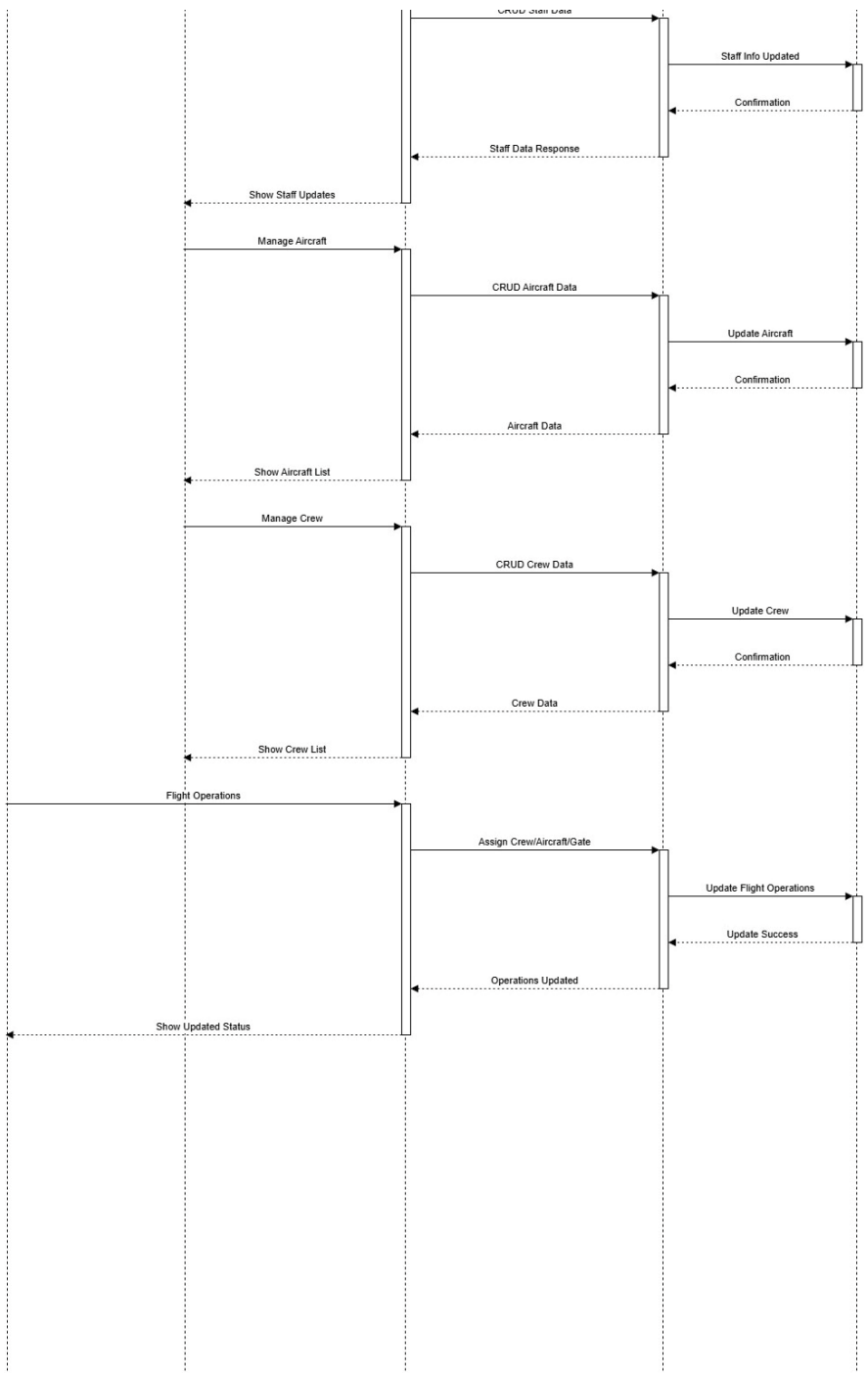
4.2. Dashboard Sequence Diagram:



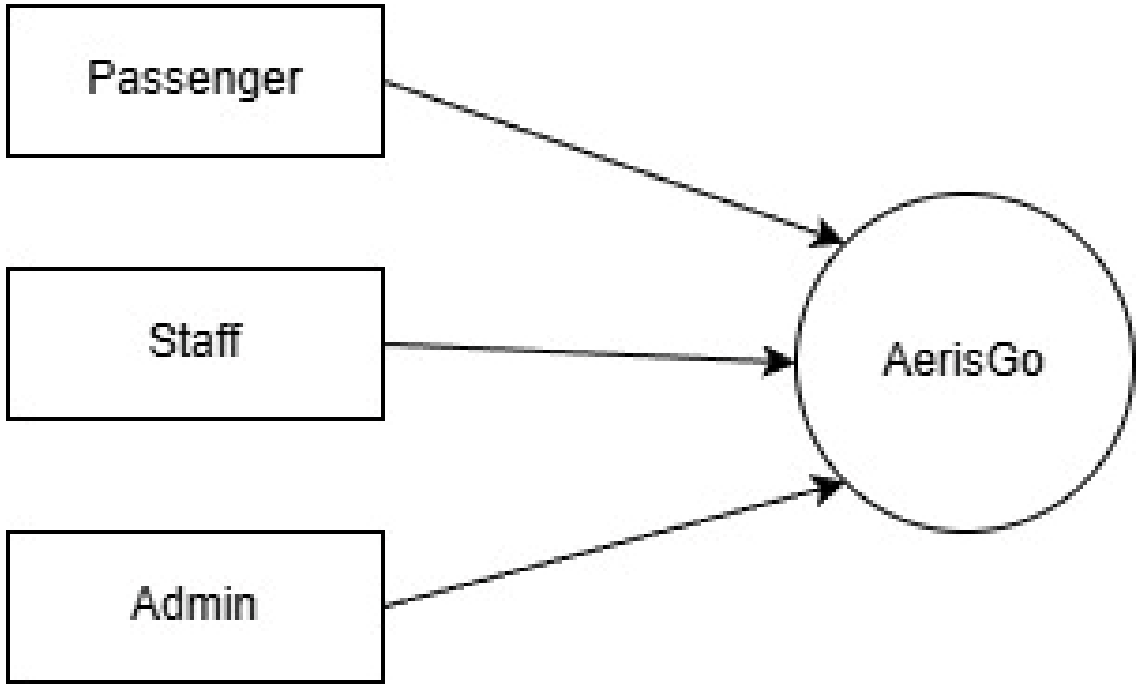
4.2. Dashboard Sequence Diagram:



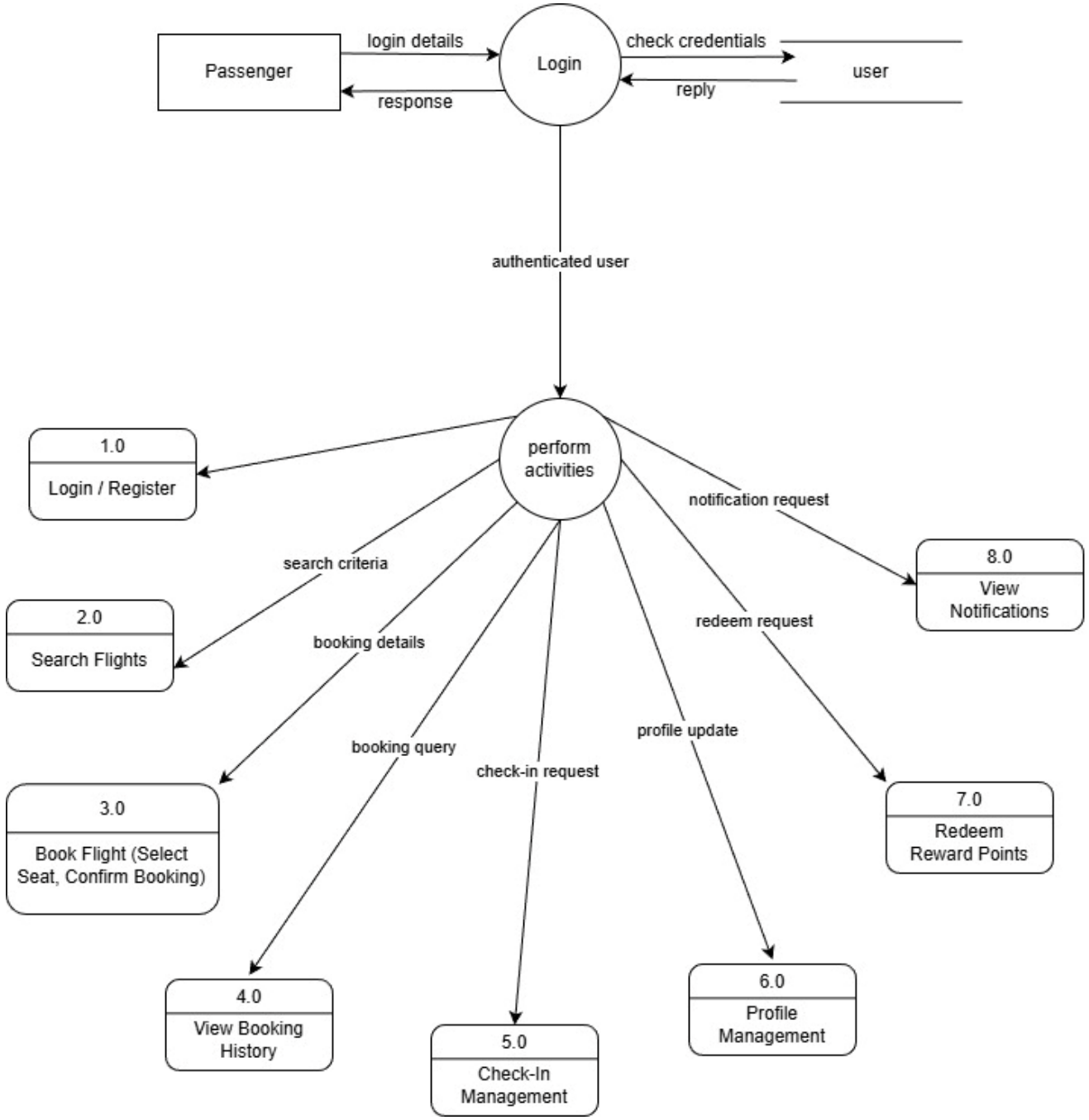
4.2. Dashboard Sequence Diagram:



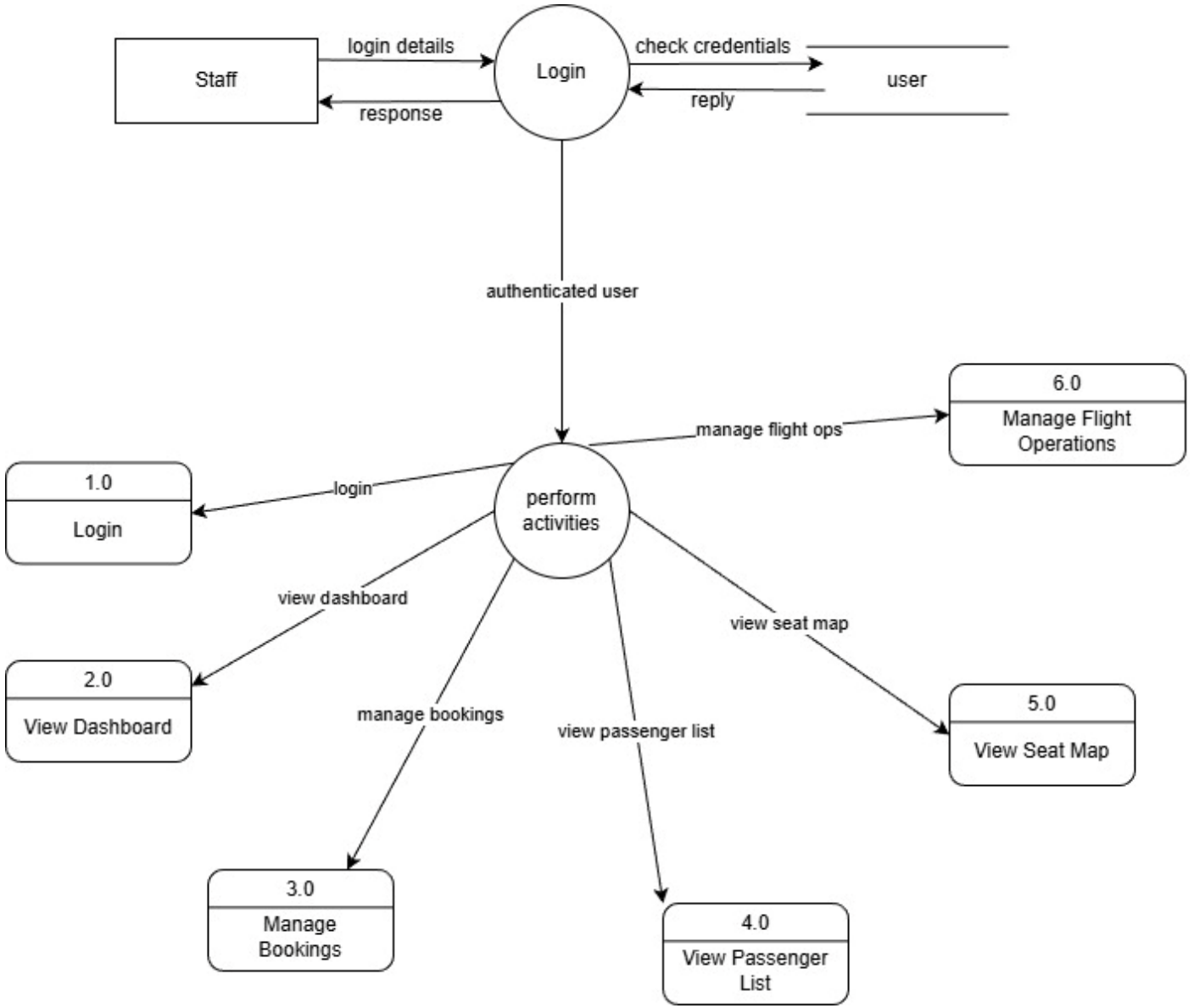
5.1. DFD Diagram(Level 0):



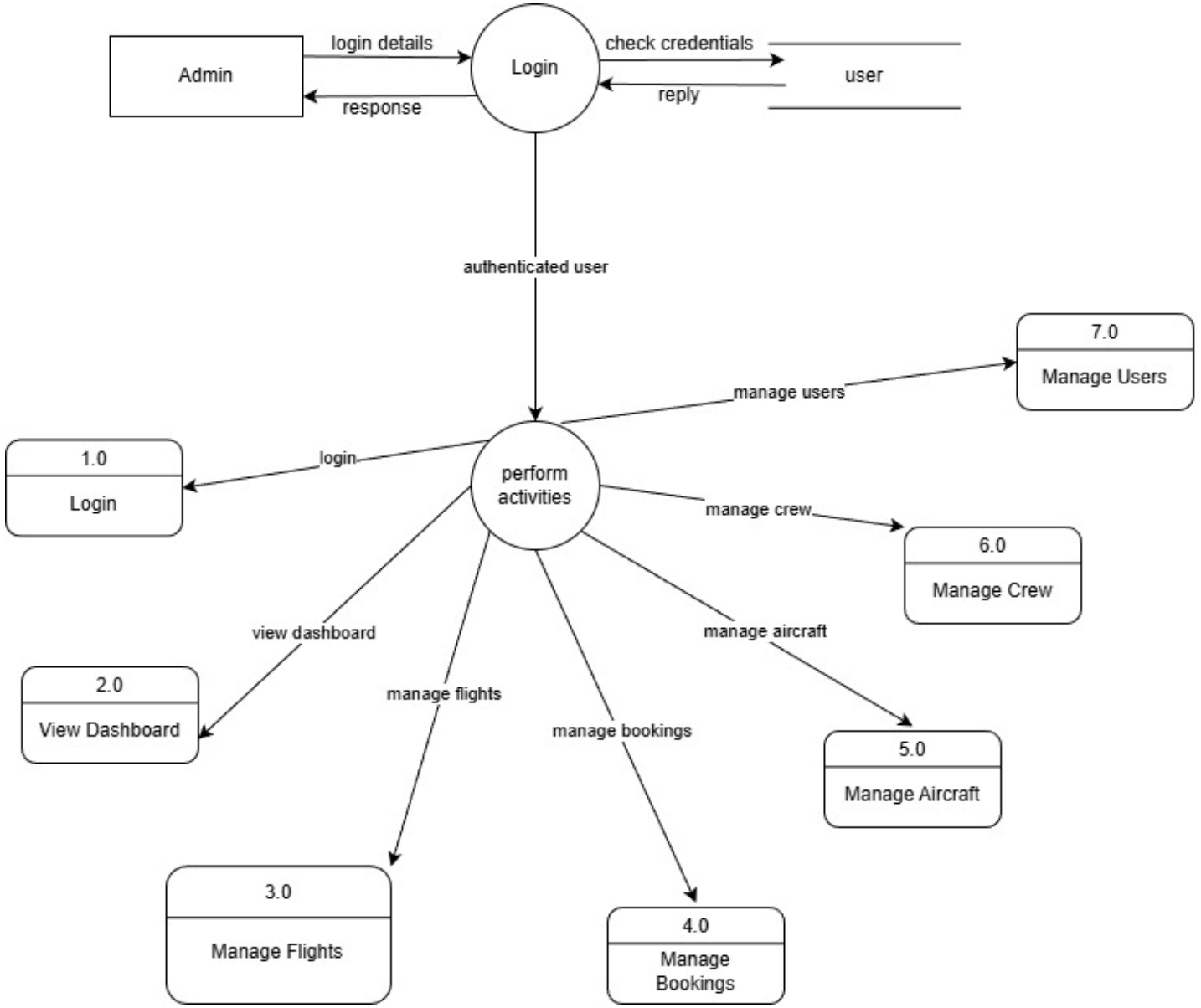
6.1. Passenger DFD Diagram (Level 1):



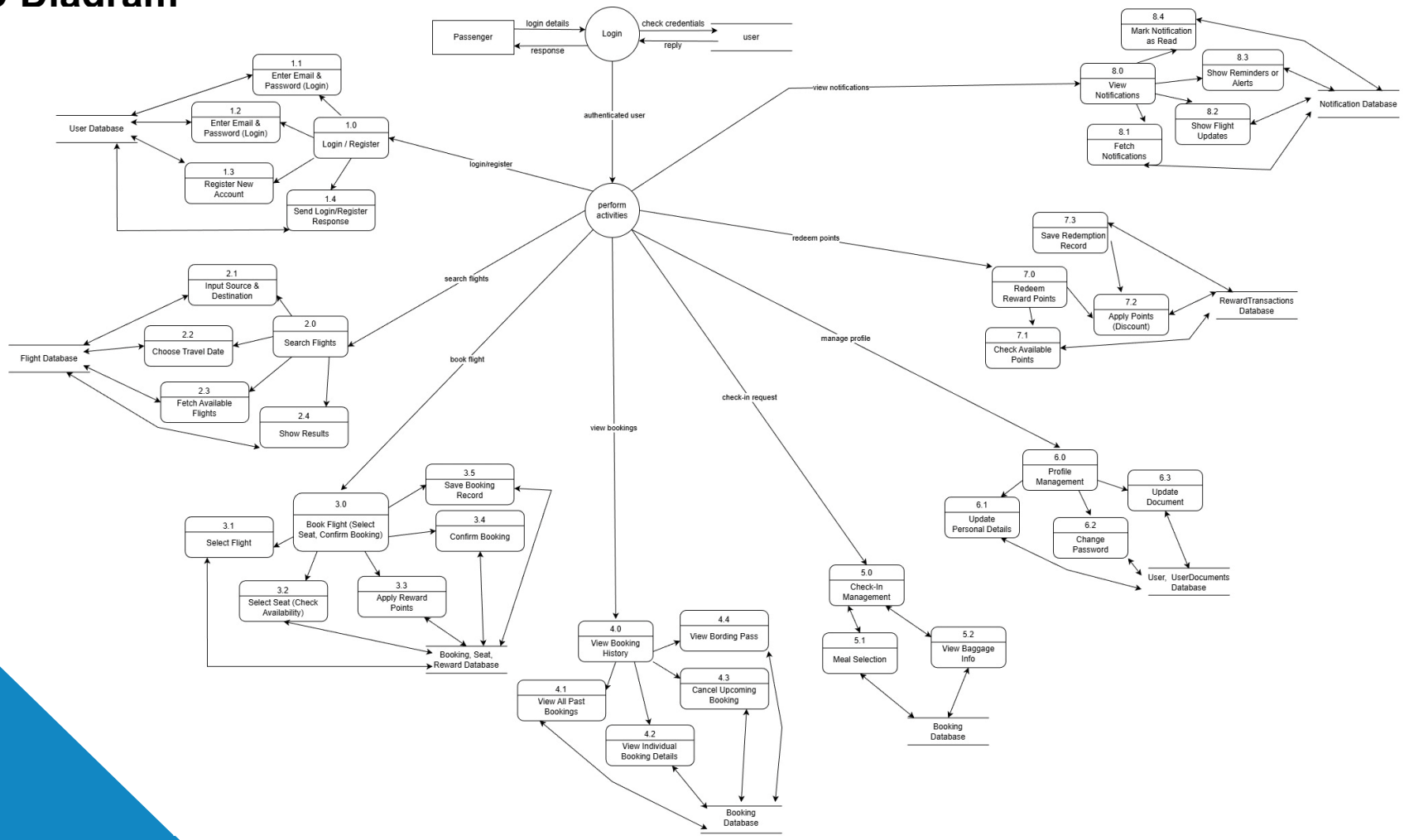
6.2. Staff DFD Diagram (Level 1):



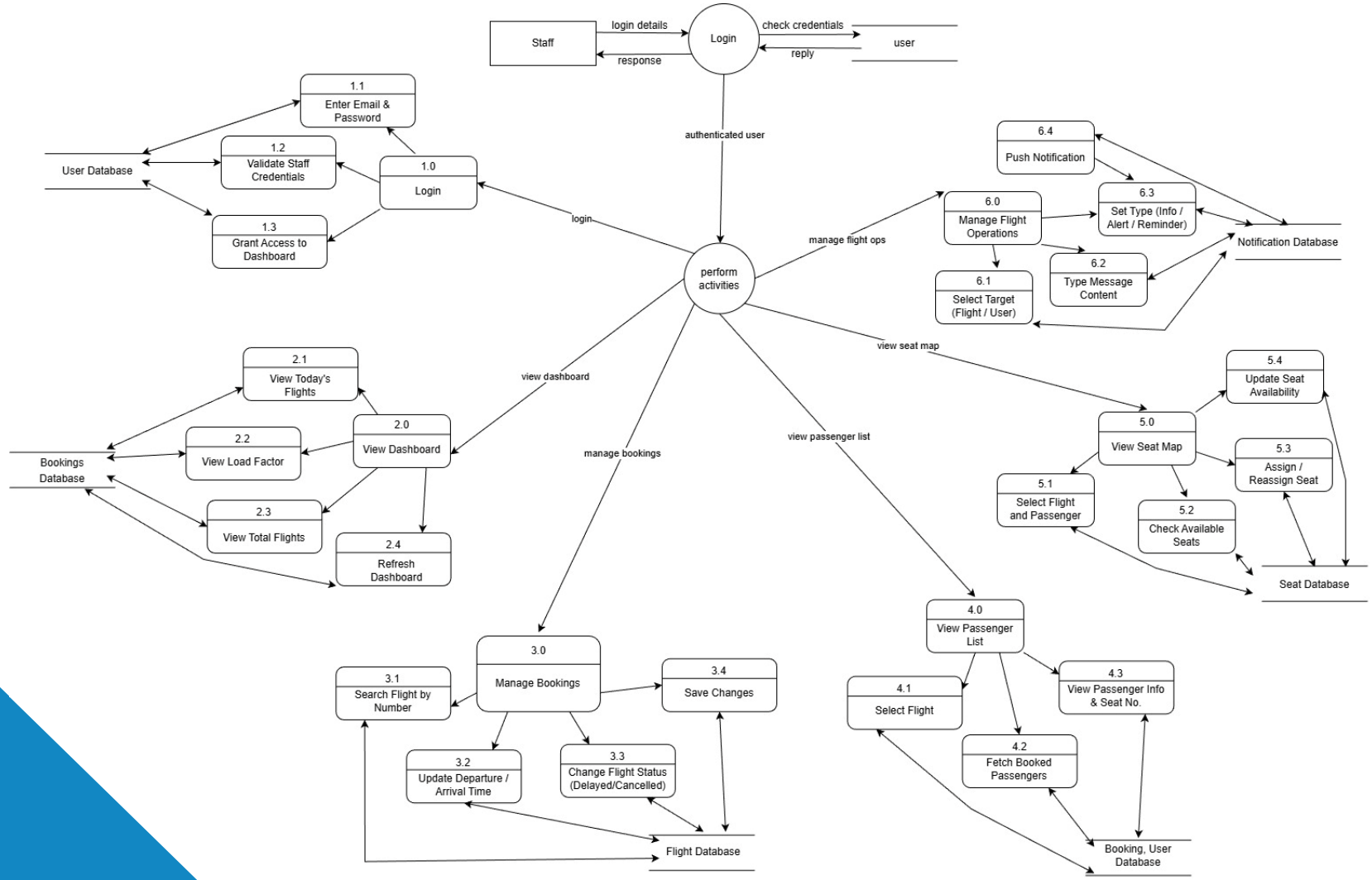
6.3. Admin DFD Diagram (Level 1):



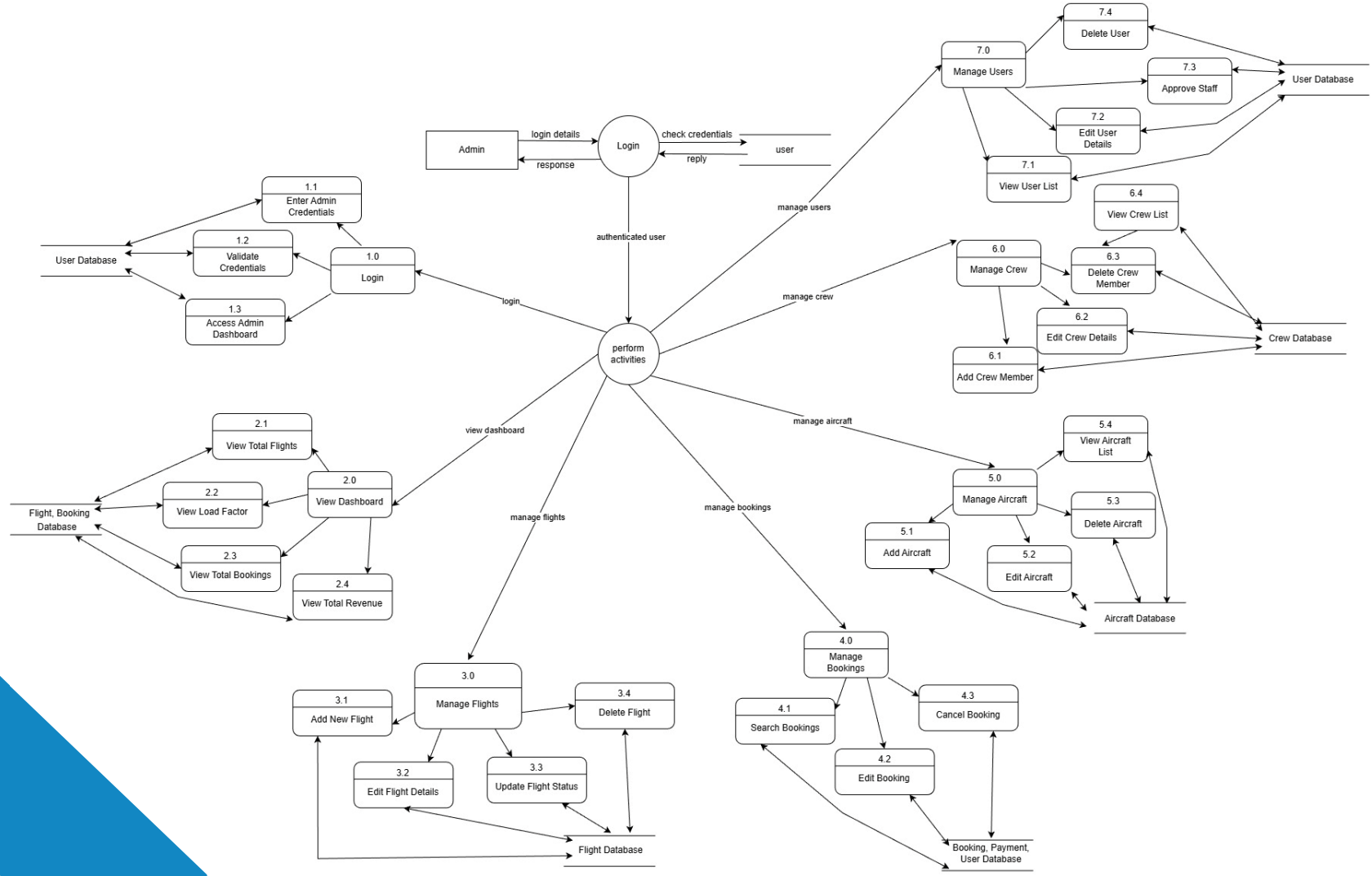
7.1. Passenger DFD Diagram (Level 2):



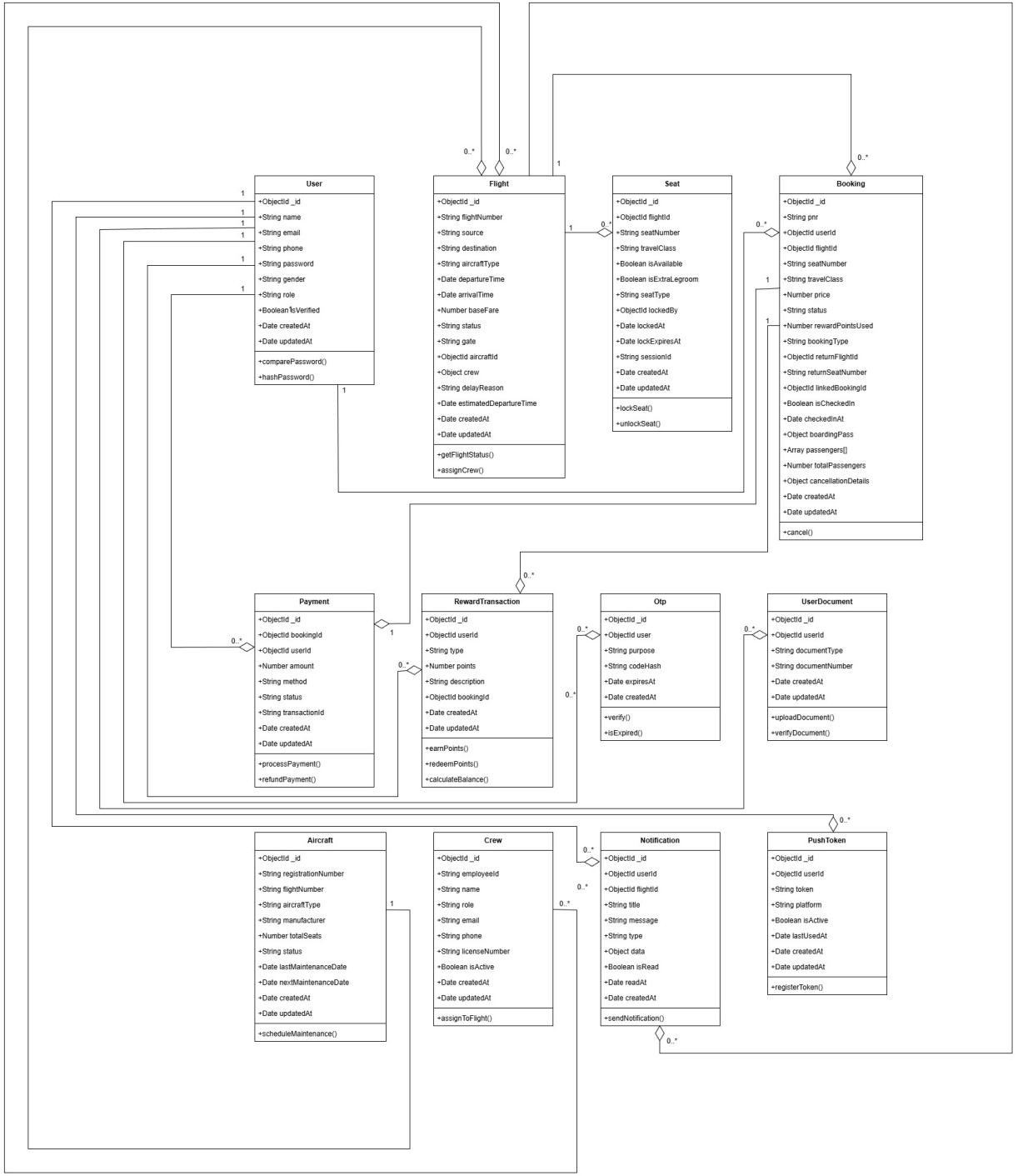
7.2. Staff DFD Diagram (Level 2):



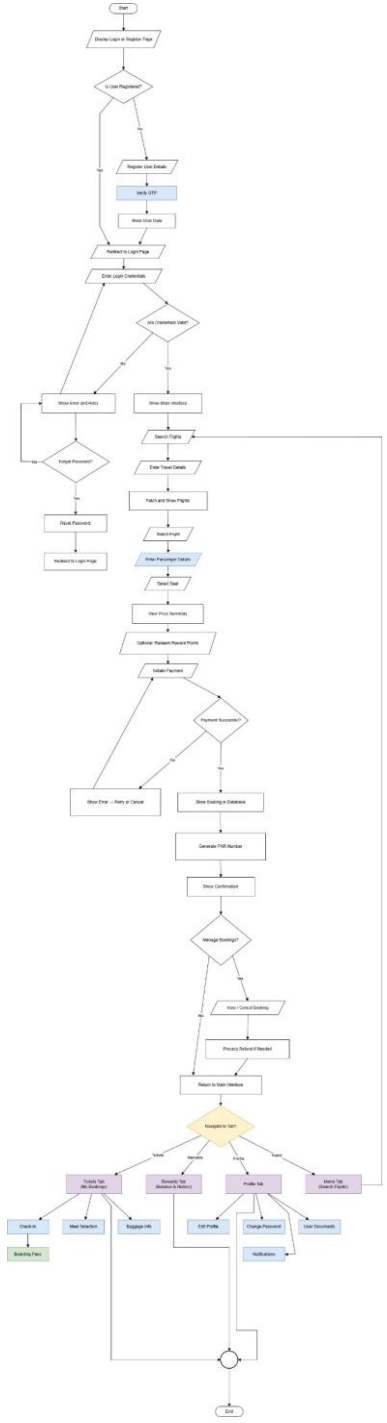
7.3. Admin DFD Diagram (Level 2):



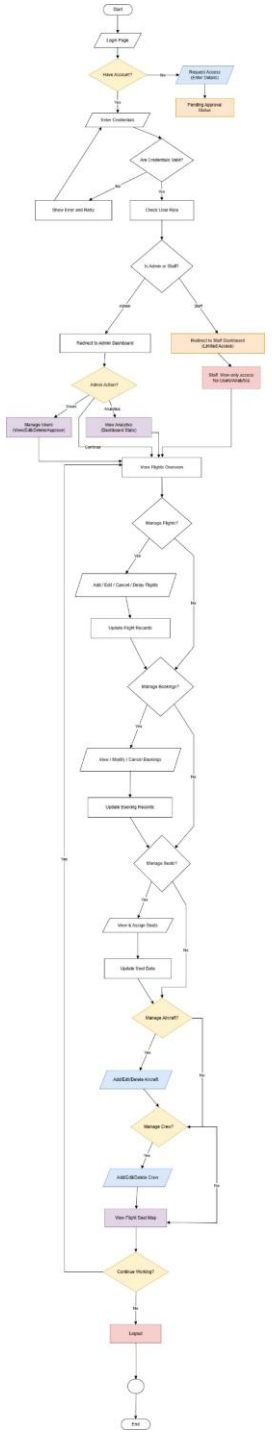
8.1. Class Diagram:



9.1. Passenger Flowchart:



9.2. Dashboard Flowchart:



Reference and Sources

This document has been prepared based on the planning and design discussions done by our project team. To complete the SRS smoothly, we referred to a few trusted sources and tools that guided us technically and helped us structure the document properly. The following references were used during the preparation of this document:

- **React Native documentation** – https://reactnative.dev/docs/getting-started
- **React documentation** – https://react.dev/learn
- **MongoDB documentation** – https://www.mongodb.com/docs/
- **Express.js official guide** – https://expressjs.com/
- **Node.js documentation** – https://nodejs.org/docs/latest/api/
- **Expo documentation** – https://docs.expo.dev/

Summary

- AerisGo is a newly built system that connects passengers and airline operations on a single modern platform.
- It provides three interfaces: a passenger mobile app, a passenger web app, and an operations dashboard for staff and admins.
- The passenger apps let users search flights, book tickets, select seats, manage bookings, check in, view boarding passes, and use rewards.
- The operations dashboard helps staff manage flights, crew, bookings, delays, and passenger lists with clear analytics.
- A key feature is real-time seat selection, showing live availability and preventing seat conflicts.
- All interfaces use a shared backend built with Node.js, Express.js, and MongoDB, ensuring data consistency and easy maintenance.

Thank You