Database Model Design for FlyNext

1. Introduction

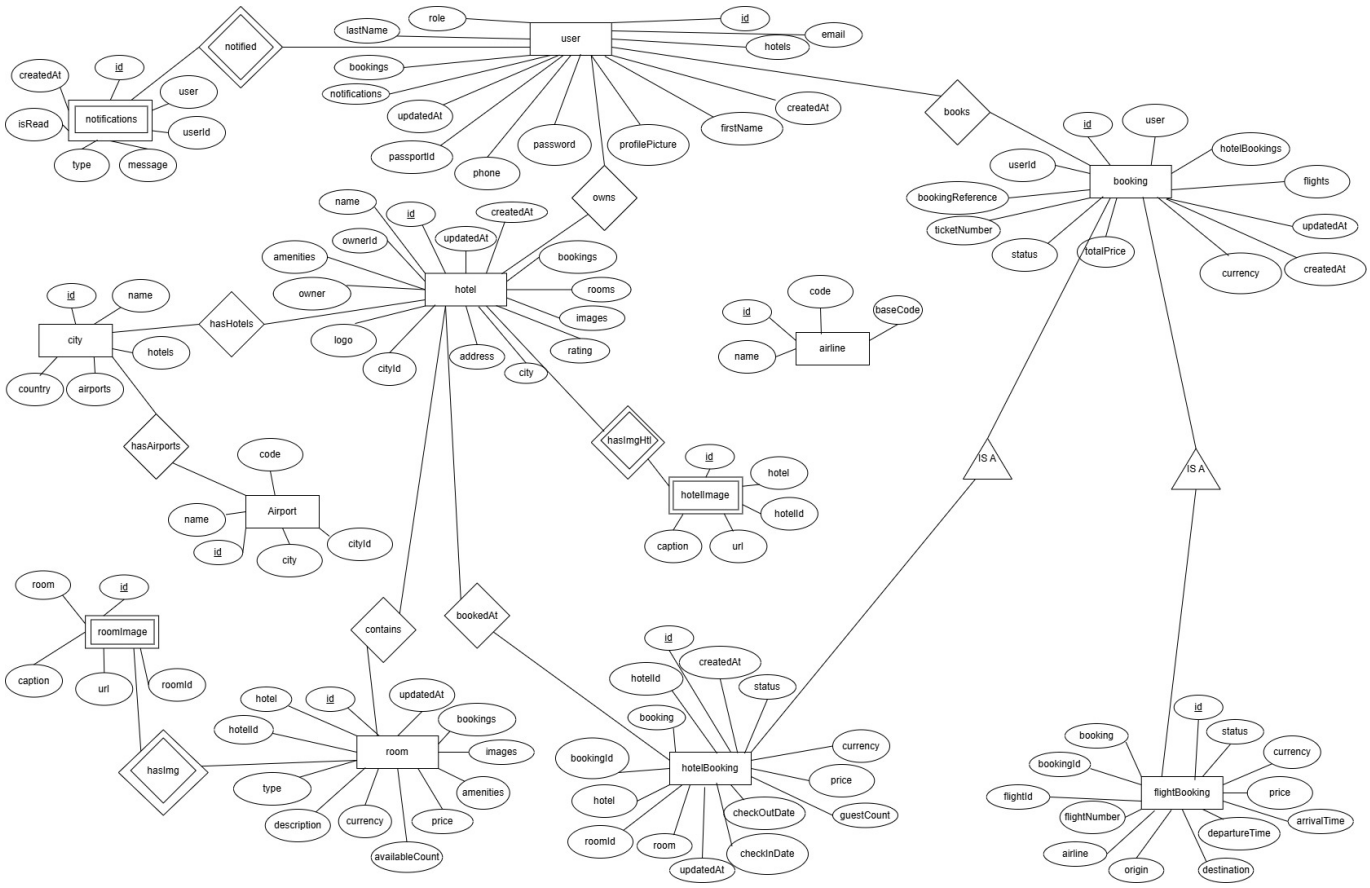
This document describes the database design for the FlyNext travel booking system. The database is implemented using Prisma and consists of multiple relational tables that support authentication, flights, hotels, bookings, and notifications.

The database is designed to:

- Manage **user authentication** and roles.
- Store **flight and hotel data** efficiently.
- Allow **users to book flights and hotels**.
- Handle **notifications** for booking confirmations, cancellations, and updates.

---

2. Entity-Relationship Diagram (ERD)



3. Database Models Overview

Below is a detailed breakdown of the database models, including their **fields, relationships, and purpose** in the FlyNext system.

◇ User Model (`User`)

Purpose: Stores user accounts, including **travelers, hotel owners, and admins**.

Fields:

- `id` (Primary Key, UUID) – Unique identifier for each user.
- `firstName`, `lastName` (String) – The user's real name.
- `email` (String, Unique) – Used for login and communication.
- `passwordHash` (String) – Securely stored password.
- `role` (`USER`, `HOTEL_OWNER`, `ADMIN`) – Defines user permissions.
- `createdAt`, `updatedAt` (Timestamp) – Track account creation and updates.

Relationships:

- A `User` can create **multiple Bookings** (`1:N`).
- A `User` can own **multiple Hotels** (`1:N`) if they are a `HOTEL_OWNER`.
- A `User` receives **Notifications** related to bookings (`1:N`).

✈️ Flight Model (`Flight`)

Purpose: Stores **available flights** for searching and booking.

Fields:

- `id` (Primary Key, UUID) – Unique flight identifier.
- `flightNumber` (String, Unique) – The airline's flight code (e.g., "AC101").
- `airlineId` (Foreign Key → `Airline`) – Links to the airline operating the flight.
- `departureAirportId`, `arrivalAirportId` (Foreign Keys → `Airport`) – Departure and arrival locations.
- `departureTime`, `arrivalTime` (Timestamp) – Scheduled flight times.
- `price` (Decimal) – Cost of the flight.

Relationships:

- A `Flight` belongs to **one Airline** (`N:1`).
- A `Flight` can be booked **by multiple users** (`N:M` via `Booking`).

🏨 Hotel Model (`Hotel`)

Purpose: Stores information about **hotels available for booking**.

Fields:

- `id` (Primary Key, UUID) – Unique hotel identifier.
- `name` (String) – Name of the hotel.
- `address` (String) – Street address.
- `cityId` (Foreign Key → `City`) – Location of the hotel.
- `ownerId` (Foreign Key → `User`) – The hotel owner.

Relationships:

- A `Hotel` has **many Rooms** (`1:N`).
- A `Hotel` is **owned by a User** (`N:1`).

🏠 Room Model (`Room`)

Purpose: Represents **individual rooms in a hotel** that users can book.

Fields:

- `id` (Primary Key, UUID) – Unique room identifier.
- `hotelId` (Foreign Key → `Hotel`) – Belongs to a hotel.
- `roomType` (String) – Type of room (e.g., "Single", "Double").
- `pricePerNight` (Decimal) – Cost per night.
- `available` (Boolean) – Indicates if the room is currently available.

Relationships:

- A `Room` belongs to a **Hotel** (`N:1`).
- A `Room` can be **booked multiple times** (`N:M` via `Booking`).

📌 Booking Model (`Booking`)

Purpose: Stores **user reservations** for flights and hotel rooms.

Fields:

- `id` (Primary Key, UUID) – Unique booking identifier.
- `userId` (Foreign Key → `User`) – The traveler making the booking.
- `flightId` (Foreign Key → `Flight`, Optional) – The flight booked.
- `roomId` (Foreign Key → `Room`, Optional) – The hotel room booked.
- `status` (`CONFIRMED`, `CANCELLED`) – Current booking status.
- `createdAt`, `updatedAt` (Timestamp) – Track when the booking was made.

Relationships:

- A `Booking` is made by **one User** (`N:1`).
- A `Booking` can include **both a Flight and a Room** (Optional fields).

🔔 Notification Model (`Notification`)

Purpose: Stores **alerts for users** about bookings, cancellations, and other events.

Fields:

- `id` (Primary Key, UUID) – Unique notification identifier.
- `userId` (Foreign Key → `User`) – The recipient of the notification.
- `message` (String) – The notification text.
- `type` (`BOOKING_CONFIRMATION`, `BOOKING_CANCELLATION`) – Defines the notification purpose.
- `createdAt` (Timestamp) – Time when the notification was generated.

Relationships:

- A `Notification` belongs to **one User** (`N:1`).

🌍 City Model (`City`)

Purpose: Stores **city information** used for hotels and flights.

Fields:

- `id` (Primary Key, UUID) – Unique city identifier.
- `name` (String) – City name.
- `country` (String) – Country where the city is located.

Relationships:

- A `City` can have **multiple Hotels** (`1:N`).
- A `City` is referenced in **Flight departure and arrival locations**.

---

4. Relationships Between Tables

- **A `User` can make multiple `Bookings`** (`1:N`).
- **A `Flight` belongs to an `Airline` but has many `Bookings`** (`N:1`).
- **A `Hotel` has multiple `Rooms`, and `Rooms` are booked by users** (`1:N`).
- **A `Notification` belongs to a `User`** (`N:1`).
- **A `City` is linked to both `Hotels` and `Flights`** (`1:N`).

---

5. Summary

This document provides a structured overview of the **FlyNext database schema**. The schema is designed to:

- Efficiently **store and retrieve** flight and hotel data.
- Allow **users to search, book, and manage reservations**.
- Ensure **role-based access** (regular users, hotel owners, admins).
- Enable **notifications** for booking updates.

By maintaining **clear entity relationships**, the system provides an efficient and scalable structure for handling travel-related services.