FlyNext Database Design Documentation

CSC309 Programming on the Web - Part 1 Submission
March 9, 2025

Introduction

FlyNext is a travel search and booking platform designed to simplify flight and hotel reservations. It integrates real-time flight data from the Advanced Flights System (AFS) via REST APIs while storing hotel, user, and booking data locally in an SQLite database. This document outlines the database schema, comprising 12 models that support user accounts, bookings, hotel management, and AFS-seeded city/airport data. The design ensures efficient search, booking, and notification features for visitors, users, and hotel owners, with explicit deletion behaviors defined via onDelete constraints.

Model Design

Overview

The FlyNext database consists of 12 interconnected models built with Prisma and SQLite. These models manage user profiles, travel itineraries, hotel details, and supporting data like cities and airports. Enums ensure consistent states (e.g., booking status), and relationships link entities with explicit deletion rules (Cascade or SetNull) for data integrity.

Key Models and Fields

1. **User**
    - **Purpose**: Stores user profiles, including authentication and role data.
    - **Fields**: id (PK), firstName, lastName, userName (unique), email (unique), role (UserRole: USER or OWNER), password, profilePicture, phoneNumber, createdAt, updatedAt.
    - **Notes**: role distinguishes regular users from hotel owners.
2. **Booking**
    - **Purpose**: Tracks user itineraries, combining flights and/or hotel stays.
    - **Fields**: id (PK), userId (FK to User), status (BookingStatus: PENDING, CONFIRMED, CANCELLED), totalPrice, paymentDetails, createdAt, updatedAt.
    - **Notes**: totalPrice and paymentDetails are nullable until booking confirmation.

3. **BookingItem**
    - **Purpose**: Represents individual components (flight or hotel) within a booking.
    - **Fields**: id (PK), bookingId (FK to Booking), type (String: "FLIGHT" or "HOTEL"), hotelId (FK to Hotel), roomTypeId (FK to RoomType), referenceId (AFS flight reference), startDate, endDate, price, status (ItemStatus).
    - **Notes**: Nullable fields like referenceId accommodate flight bookings via AFS.
4. **Hotel**
    - **Purpose**: Manages hotel data owned by users.
    - **Fields**: id (PK), cityId (FK to City), ownerId (FK to User), name, logo, address, latitude, longitude, starRating, amenities.
    - **Notes**: amenities is stored as a string (e.g., JSON or comma-separated).
5. **RoomType**
    - **Purpose**: Defines room options within a hotel.
    - **Fields**: id (PK), hotelId (FK to Hotel), name, pricePerNight, amenities.
    - **Notes**: Linked to availability and booking items.
6. **City and Airport**
    - **Purpose**: Stores AFS-seeded data for flight and hotel searches.
    - **Fields (City)**: id (PK), name, country.
    - **Fields (Airport)**: id (PK, String from AFS), cityId (FK to City), code (unique), name, country.

Relationships

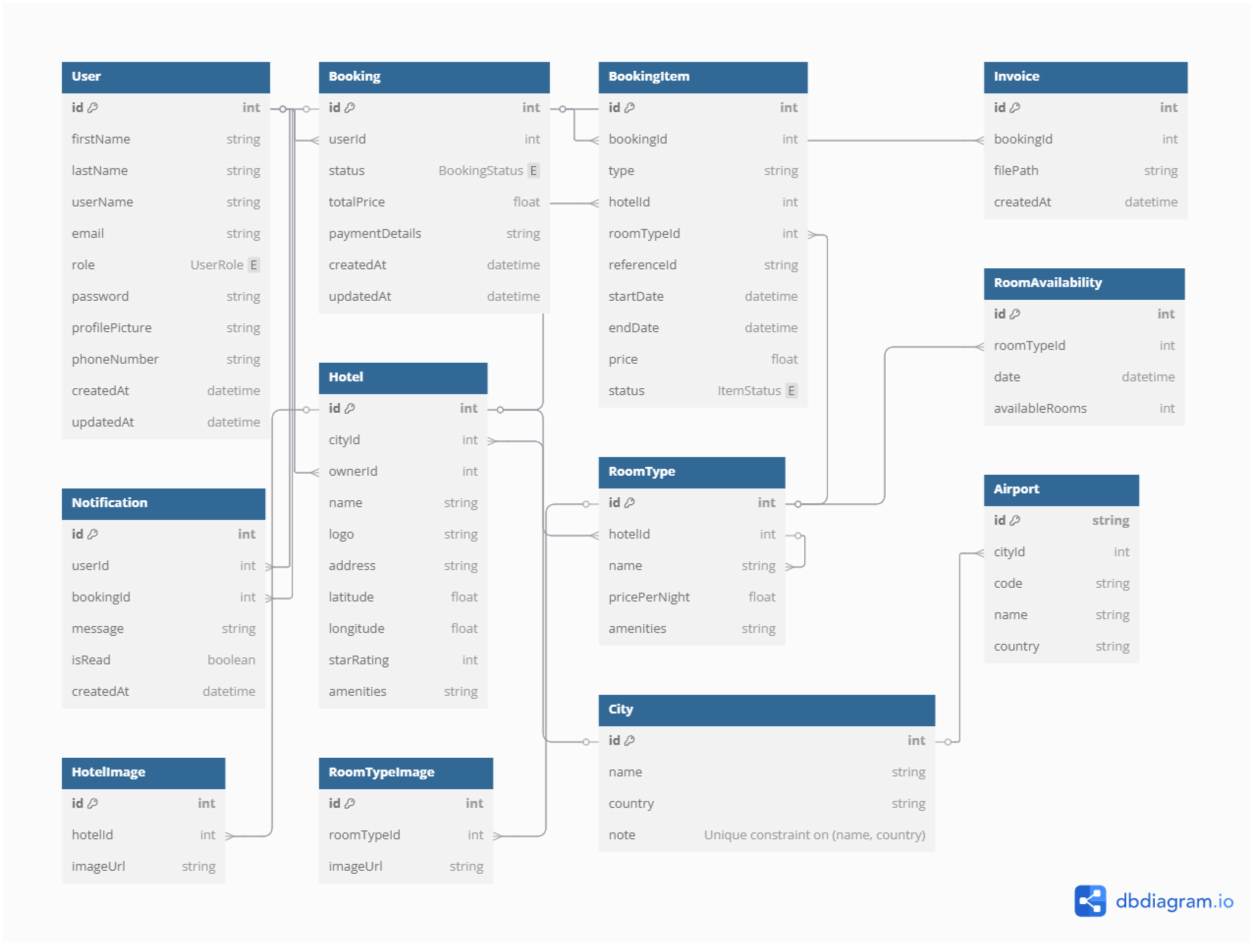
- **User**:
    - One-to-many with Booking (userId, onDelete: Cascade): Deleting a user deletes their bookings.
    - One-to-many with Hotel (ownerId, onDelete: Cascade): Deleting a user deletes their owned hotels.
    - One-to-many with Notification (userId, onDelete: Cascade): Deleting a user deletes their notifications.
- **Booking**:
    - One-to-many with BookingItem (bookingId, onDelete: Cascade): Deleting a booking deletes its items.

    - One-to-one with Invoice (bookingId, onDelete: Cascade): Deleting a booking deletes its invoice.
    - One-to-many with Notification (bookingId, onDelete: SetNull): Deleting a booking sets bookingId to null in notifications.
- **Hotel**:
    - One-to-many with RoomType (hotelId, onDelete: Cascade): Deleting a hotel deletes its room types.
    - One-to-many with HotelImage (hotelId, onDelete: Cascade): Deleting a hotel deletes its images.
    - One-to-many with BookingItem (hotelId, onDelete: SetNull): Deleting a hotel sets hotelId to null in booking items.
    - Many-to-one with City (cityId): No onDelete specified (SQLite default applies).
    - Many-to-one with User (ownerId, onDelete: Cascade).
- **RoomType**:
    - One-to-many with RoomAvailability (roomTypeId, onDelete: Cascade): Deleting a room type deletes its availability records.
    - One-to-many with RoomTypeImage (roomTypeId, onDelete: Cascade): Deleting a room type deletes its images.
    - One-to-many with BookingItem (roomTypeId, onDelete: SetNull): Deleting a room type sets roomTypeId to null in booking items.
- **City**:
    - One-to-many with Hotel (cityId): No onDelete specified.
    - One-to-many with Airport (cityId): No onDelete specified.

Additional Models

- **Invoice**: Stores PDF file paths for bookings (1:1 with Booking, onDelete: Cascade).
- **Notification**: Tracks user alerts (e.g., booking updates).
- **RoomAvailability**: Manages hotel room counts by date.
- **HotelImage** and **RoomTypeImage**: Store image URLs for hotels and rooms.

ER Diagram



The diagram illustrates all 12 entities, their attributes, and relationships. Primary keys are underlined, foreign keys are marked with an asterisk (*), and enums are listed separately. Relationships use crow's foot notation (e.g., one-to-many as ||—o<). The composite unique constraint on City (name, country) is noted in the table description. Deletion behaviors (Cascade or SetNull) are indicated on relationship lines where applicable.

Notes

- **Enums**: BookingStatus, UserRole, ItemStatus, and BookingType ensure data consistency.
- **Nullable Fields**: Fields like referenceId in BookingItem support AFS integration, while latitude/longitude in Hotel are optional for mapping.
- **Unique Constraints**: City has a composite unique constraint on name and country, indicated as a note in the ER diagram to prevent duplicate city entries by country.

- **Deletion Rules**: onDelete: Cascade ensures dependent records are removed (e.g., bookings with users), while onDelete: SetNull preserves records with nullified references (e.g., notifications after booking deletion).

This schema supports FlyNext's core backend features: user management, flight/hotel bookings, and hotel ownership, all integrated with AFS data.

Citation

This document and the accompanying ER diagram were fully generated by Grok, an artificial intelligence developed by xAI, on March 9, 2025, during an interaction via the xAI platform.