Our Database Design

This document outlines our database structure that powers FlyNext, including the key models, their fields, and how they connect to support features like user accounts, hotel management, flight bookings, and itineraries. We also have an accompanying Entity-Relationship (ER) diagram so that the relationships can be better understood visually.

Database Models

Optional

User

The User model represents anyone who interacts with FlyNext, whether as a regular traveler or a hotel owner.

- **Fields:**
  - id: Unique identifier (auto-incremented integer).
  - firstName: User's first name (text).
  - lastName: User's last name (text).
  - email: User's email address (unique text, used for login).
  - password: Hashed password for secure login (text).
  - profilePicture: URL to the user's profile image (text, nullable).
  - phoneNumber: User's contact number (text).
  - role: User's role, either USER or HOTEL_OWNER (defaults to USER).
- **Relationship Fields:**
  - hotels[]: Owns multiple Hotel entries (one-to-many).
  - hotelBookings[]: Books multiple HotelBooking entries (one-to-many).
  - flightBookings[]: Books multiple FlightBooking entries (one-to-many).
  - notifications[]: Receives multiple Notification entries (one-to-many).
  - itineraries[]: Creates multiple Itinerary entries (one-to-many).

Hotel

The Hotel model represents a property listed on FlyNext, managed by a user with the HOTEL_OWNER role.

- **Fields:**

    - id: Unique identifier (auto-incremented integer).
    - name: Hotel name (text).
    - logo: URL to the hotel's logo (text, nullable).
    - address: Full street address (text).
    - location: City and country (text, e.g., "Toronto, Canada").
    - starRating: Hotel rating (integer, e.g., 1 to 5).
    - ownerId: Foreign key linking to the User who owns this hotel.
- **Relationship Fields:**
    - owner: Belongs to one User (many-to-one).
    - roomTypes[]: Has multiple RoomType entries (one-to-many).
    - bookings[]: Has multiple HotelBooking entries (one-to-many).
    - images[]: Has multiple HotelImage entries (one-to-many).

HotelImage

The HotelImage model stores images associated with a hotel.

- **Fields:**
    - id: Unique identifier (auto-incremented integer).
    - url: URL to the image file (text).
    - hotelId: Foreign key linking to the associated Hotel.
- **Relationship Fields:**
    - hotel: Belongs to one Hotel (many-to-one).

RoomType

The RoomType model defines the types of rooms available at a hotel (e.g., Twin, Double).

- **Fields:**
    - id: Unique identifier (auto-incremented integer).
    - name: Room type name (text, e.g., "Twin").
    - totalRooms: Total number of rooms of this type (integer).
    - pricePerNight: Cost per night (floating-point number).
    - hotelId: Foreign key linking to the associated Hotel.
- **Relationship Fields:**
    - hotel: Belongs to one Hotel (many-to-one).
    - amenities[]: Has multiple RoomTypeAmenity entries (one-to-many).
    - images[]: Has multiple RoomTypeImage entries (one-to-many).
    - hotelBookings[]: Linked to multiple HotelBooking entries (one-to-many).

RoomTypeImage

The RoomTypeImage model stores images for each room type.

- **Fields:**
  - id: Unique identifier (auto-incremented integer).
  - url: URL to the image file (text).
  - roomTypeId: Foreign key linking to the associated RoomType.
- **Relationship Fields:**
  - roomType: Belongs to one RoomType (many-to-one).

RoomTypeAmenity

The RoomTypeAmenity model lists amenities available in a room type (e.g., Pool, Parking).

- **Fields:**
  - id: Unique identifier (auto-incremented integer).
  - name: Amenity name (text).
  - roomTypeId: Foreign key linking to the associated RoomType.
- **Relationship Fields:**
  - roomType: Belongs to one RoomType (many-to-one).

Itinerary

The Itinerary model represents a user's travel plan, combining hotel and/or flight bookings.

- **Fields:**
  - id: Unique identifier (auto-incremented integer).
  - userId: Foreign key linking to the User who created it.
  - creditCardNumber: Masked credit card number (text, nullable, e.g., "****1234").
  - cardExpiry: Card expiry date (text, nullable, e.g., "12/25").
  - invoiceUrl: URL to a PDF invoice (text, nullable).
  - status: Booking status (PENDING, CONFIRMED, CANCELLED, defaults to PENDING).
- **Relationship Fields:**
  - user: Belongs to one User (many-to-one).
  - hotelBooking?: Has one optional HotelBooking (one-to-one, nullable).
  - flightBooking?: Has one optional FlightBooking (one-to-one, nullable).

HotelBooking

The HotelBooking model tracks a user's hotel reservation.

- **Fields:**
  - id: Unique identifier (auto-incremented integer).
  - hotelId: Foreign key linking to the booked Hotel.
  - roomTypeId: Foreign key linking to the booked RoomType.
  - checkInDate: Start date of stay (date-time).
  - checkOutDate: End date of stay (date-time).
  - status: Booking status (`PENDING`, `CONFIRMED`, `CANCELLED`, defaults to `PENDING`).
  - itineraryId: Foreign key linking to the associated Itinerary (unique).
  - userId: Foreign key linking to the User who booked it.
- **Relationship Fields:**
  - hotel: Belongs to one Hotel (many-to-one).
  - roomType: Belongs to one RoomType (many-to-one).
  - itinerary: Belongs to one Itinerary (one-to-one).
  - user: Belongs to one User (many-to-one).
  - notifications[]: Has multiple Notification entries (one-to-many).

FlightBooking

The FlightBooking model tracks a user's flight reservation, integrated with the Advanced Flights System (AFS).

- **Fields:**
  - id: Unique identifier (auto-incremented integer).
  - flightBookingRef: Reference code from AFS (text).
  - flightTicketNumber: Ticket number from AFS (text, nullable).
  - flightPrice: Total cost (floating-point number).
  - status: Booking status (`PENDING`, `CONFIRMED`, `CANCELLED`, defaults to `PENDING`).
  - itineraryId: Foreign key linking to the associated Itinerary (unique).
  - userId: Foreign key linking to the User who booked it.
- **Relationships:**
  - itinerary: Belongs to one Itinerary (one-to-one).
  - user: Belongs to one User (many-to-one).
  - notifications[]: Has multiple Notification entries (one-to-many).

Notification

The Notification model informs users and hotel owners of booking updates.

- **Fields:**
  - id: Unique identifier (auto-incremented integer).
  - userId: Foreign key linking to the receiving User.
  - message: Notification text (text).
  - isRead: Whether the notification has been read (boolean, defaults to false).
  - hotelBookingId: Foreign key linking to the related HotelBooking (nullable).
  - flightBookingId: Foreign key linking to the related FlightBooking (nullable).
- **Relationships:**
  - user: Belongs to one User (many-to-one).
  - hotelBooking?: Linked to one optional HotelBooking (many-to-one, nullable).
  - flightBooking?: Linked to one optional FlightBooking (many-to-one, nullable).

City

The City model stores city data fetched from AFS for local use.

- **Fields:**
  - id: Unique identifier (auto-incremented integer).
  - name: City name (text).
  - country: Country name (text).
- **Relationship Fields:**
  - airports[]: Has multiple Airport entries (one-to-many).

Airport

The Airport model stores airport data fetched from AFS.

- **Fields:**
  - id: Unique identifier (auto-incremented integer).
  - code: Airport code (text, e.g., "YYZ").
  - name: Airport name (text).
  - cityId: Foreign key linking to the associated City.
- **Relationship Fields:**
  - city: Belongs to one City (many-to-one).

Entity-Relationship Diagram

To better visualize the models above, we have created an entity-relationship diagram that outlines the relationships and fields of each model.

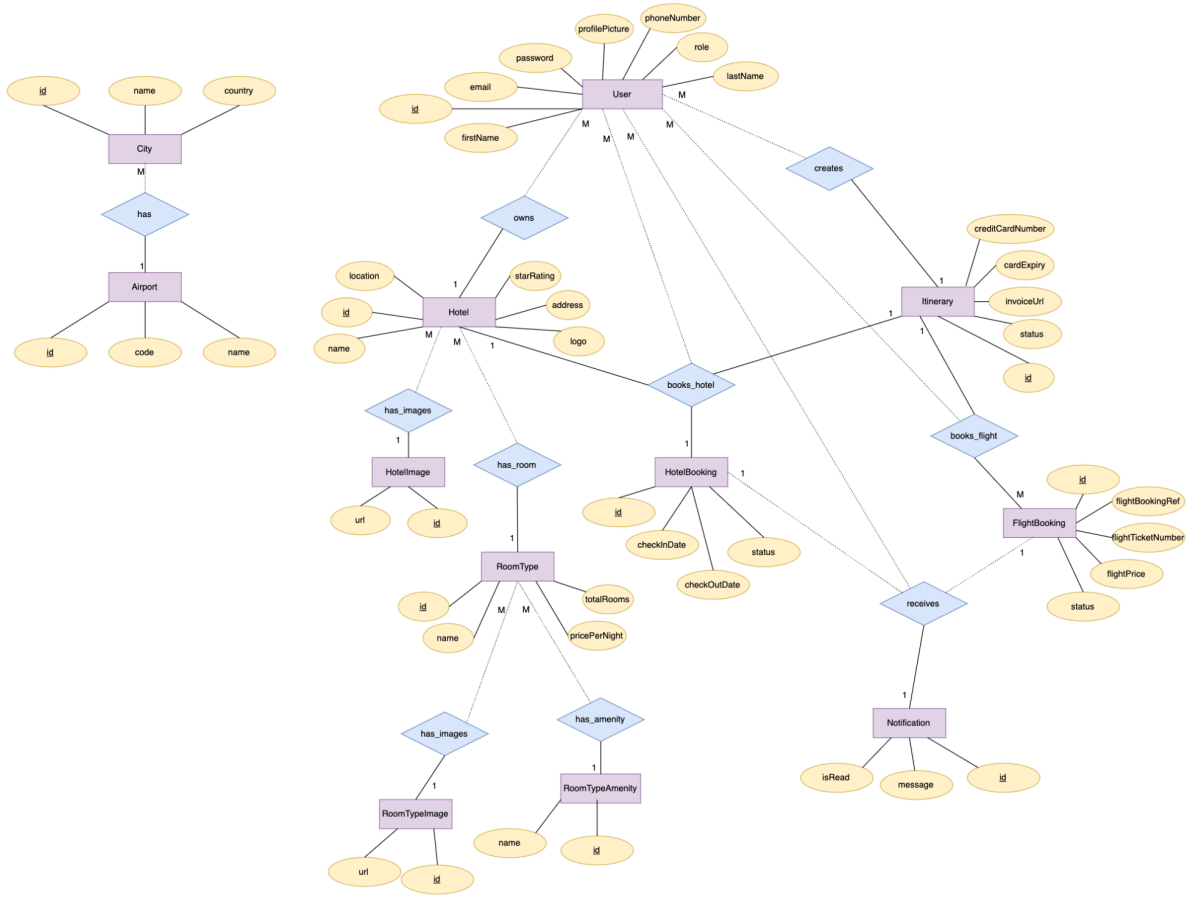

Figure 1: Entity-Relationship diagram of our database models.

Here, purple rectangles represent the model, yellow circles represent fields, and blue diamonds represent the relationships between models. The diagram can also be viewed better here.

Design Notes

- We use Role (USER, HOTEL_OWNER) and BookingStatus (PENDING, CONFIRMED, CANCELLED) to standardize options and ensure consistency.
- Room availability is not stored directly in RoomType (totalRooms is static). Instead, it's calculated by comparing totalRooms against active HotelBooking entries for a given date range.
- Flight data is external (via AFS APIs), but City and Airport data is cached locally for faster searches and to support hotel locations.