Database Model Design for FlyNext

This document describes the database models, their fields, and relationships, supplemented with class diagram.

1. 1 Database Models

User Model

Represents registered users of the system, including regular users and hotel owners.

- **Fields**: *id, firstName, lastName, email, password, phoneNumber, profilePicture, role, createdAt, updatedAt.*
- **Relationships**:
    - One-to-many with Booking (a user can have multiple bookings).
    - One-to-many with Notification (a user can receive multiple notifications).

Hotel Model

Represents hotels listed in the system.

- **Fields**: *id, name, logo, address, city, starRating, images, ownerId, createdAt, updatedAt.*
- **Relationships**:
    - One-to-many with Room (a hotel contains multiple rooms).
    - One-to-many with Booking (a hotel can have multiple bookings).

Room Model

Represents individual rooms available in a hotel.

- **Fields**: *id, type, amenities, pricePerNight, images, hotelId, createdAt, updatedAt.*
- **Relationships**:
    - Many-to-one with Hotel (each room belongs to a hotel).
    - One-to-many with Booking (a room can have multiple bookings over time).

Booking Model

Tracks flight and hotel reservations.

- **Fields**: *id, userId, itinerary, hotelCost, checkIn, checkOut, roomId, bookRef, ticketNum, status, createdAt, updatedAt.*
- **Relationships**:
    - Many-to-one with User (a booking is made by a user).
    - Many-to-one with Room (a booking can be linked to a specific hotel room).
    - Many-to-many with Flight via FlightBooking (a booking can include multiple flights).

    - One-to-one with Invoice (a booking can only have one invoice)

Flight Model

Stores flight details.

- **Fields**: *flightId, flightNum, departureTime, arrivalTime, flightCost, origin, destination, airline.*
- **Relationships**:
    - Many-to-many with Booking (a flight can be part of multiple bookings).

Notification Model

Tracks notifications for users regarding their bookings.

- **Fields**: *id, userId, message, isRead, isUpdate.*
- **Relationships**:
    - Many-to-one with User (notifications belong to a user).

Invoice Model

Tracks the transaction of each completed or cancelled booking.

- **Fields**: *id, userId, bookingId, hotelCost, flightCost, refundAmount, currency, status, createdAt, updatedAt.*
- **Relationships**:
    - One-to-one with Booking (an invoice can only belong to one booking).

City Model

Store the city and country pair fetched from AFS api.

- **Fields**: *city, country.*
- When creating a hotel, we check if the hotel's city is already in the list of cities stored in the City model. However, there is no relationship between the City and Hotel models.

Airport Model

Store the airport data fetched from AFS api.

- **Fields**: *id, code, name, city, country.*
- The Airport model has no direct relationship with the City model in the schema. However, whenever we create an airport, we check if the city's name provided for the airport exists in the list of cities. This ensures that each airport is correctly associated with an existing city, but there is no explicit relationship defined between the Airport and City models in the schema.

1. 2 Class Diagram

The following class diagram represents the models' connections

```
class User:
    # A User can have multiple Bookings and Notifications
    bookings: List["Booking"]
    Notification: List["Notification"]


class Booking:
    # A Booking belongs to a User and can have multiple Flights or a Room (for hotels)
    user: User
    flights: List["Flight"] = None
    room: Optional["Room"] = None
    invoice: Optional["Invoice"] = None

class Hotel:
    # A Hotel is owned by a User (Hotel Owner) and can have multiple Rooms
    ownerId: int
    rooms: List["Room"]

class Room:
    # A Room belongs to a Hotel and can have multiple Bookings
    hotel: Hotel
    bookings: List["Booking"]
```

```
class Flight:

   # A Flight can have multiple Bookings

   bookings: List["Booking"]

   origin: str  # Represents a location (origin airport)

   destination: str  # Represents a location (destination airport)


class Notification:

   # A Notification belongs to a User

   user: User


class Invoice:

   # A Invoice belongs to one completed or cancelled booking

   booking: Booking
```

This database design ensures efficient data storage, retrieval, and management for flight and hotel reservations within FlyNext.