Food Delivery app project :Schema

Food Delivery App Database Schema

1. Entities Overview

Main Entities:

- **User**: Represents both admin, customers, and delivery boys.
- **Restaurant**: Contains details about the restaurant.
- **FoodItem**: Represents individual food items offered by a restaurant, along with stock management.
- **Order**: Represents an order placed by a user.
- **OrderItem**: Contains details of food items in an order.
- **DeliveryStatus**: Tracks the status of delivery.

---

2. Schema Design

User Table

```
CREATE TABLE User (
    id BIGINT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    email VARCHAR(100) UNIQUE NOT NULL,
    password VARCHAR(255) NOT NULL,
    phone_number VARCHAR(15),
    address TEXT,
    role ENUM('ADMIN', 'CUSTOMER', 'DELIVERY_BOY') NOT NUL
L,
    is_available BOOLEAN DEFAULT TRUE, -- Applicable for de
livery boys
```

```
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

Restaurant Table

```
CREATE TABLE Restaurant (
    id BIGINT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    address TEXT NOT NULL,
    open_time TIME NOT NULL,
    close_time TIME NOT NULL,
    is_open BOOLEAN DEFAULT TRUE,
    created_by BIGINT,
    FOREIGN KEY (created_by) REFERENCES User(id) ON DELETE
SET NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

FoodItem Table

```
CREATE TABLE FoodItem (
    id BIGINT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    description TEXT,
    base_price DECIMAL(10, 2) NOT NULL,
    unit ENUM('KG', 'QUANTITY', 'BOTH') NOT NULL, -- Define
s whether the item is sold by weight, quantity, or both
    variation_name VARCHAR(50), -- Specific variation name
if needed (e.g., 'Small', 'Large')
    weight_kg DECIMAL(5, 2), -- Maximum weight in stock for
items sold by 'KG'
    quantity INT, -- Maximum quantity in stock for items so
ld by 'QUANTITY'
    available_stock DECIMAL(10, 2) NOT NULL, -- Tracks curr
ent stock level for items sold by weight or quantity
    restaurant_id BIGINT,
    FOREIGN KEY (restaurant_id) REFERENCES Restaurant(id) O
```

```
N DELETE CASCADE,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

Order Table

```
CREATE TABLE `Order` (
    id BIGINT AUTO_INCREMENT PRIMARY KEY,
    user_id BIGINT,
    restaurant_id BIGINT,
    total_price DECIMAL(10, 2) NOT NULL,
    order_time TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    status ENUM('PLACED', 'PREPARING', 'OUT_FOR_DELIVERY',
'DELIVERED', 'CANCELLED') DEFAULT 'PLACED',
    FOREIGN KEY (user_id) REFERENCES User(id) ON DELETE CAS
CADE,
    FOREIGN KEY (restaurant_id) REFERENCES Restaurant(id) O
N DELETE CASCADE
);
```

OrderItem Table

```
CREATE TABLE OrderItem (
    id BIGINT AUTO_INCREMENT PRIMARY KEY,
    order_id BIGINT,
    food_item_id BIGINT,
    unit ENUM('KG', 'QUANTITY') NOT NULL, -- Specifies the
type of the item ordered
    quantity DECIMAL(10, 2) NOT NULL, -- Number of items or
weight (in base units)
    price DECIMAL(10, 2) NOT NULL,
    FOREIGN KEY (order_id) REFERENCES `Order`(id) ON DELETE
CASCADE,
    FOREIGN KEY (food_item_id) REFERENCES FoodItem(id) ON D
ELETE CASCADE
);
```

DeliveryStatus Table

```
CREATE TABLE DeliveryStatus (
    id BIGINT AUTO_INCREMENT PRIMARY KEY,
    order_id BIGINT,
    delivery_boy_id BIGINT,
    status ENUM('ASSIGNED', 'PICKED_UP', 'DELIVERED', 'CANC
ELLED') DEFAULT 'ASSIGNED',
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDAT
E CURRENT_TIMESTAMP,
    FOREIGN KEY (order_id) REFERENCES `Order`(id) ON DELETE
CASCADE,
    FOREIGN KEY (delivery_boy_id) REFERENCES User(id) ON DE
LETE SET NULL
);
```

---

3. Relationships

1. **User and Role**: Single `User` table differentiates admin, customers, and delivery boys using the `role` field.
2. **Admin and Restaurant**: Admins manage multiple restaurants.
3. **Restaurant and FoodItem**: Restaurants offer multiple food items.
4. **FoodItem Stock**: Stock is directly managed within the `FoodItem` table, accommodating both `KG` and `QUANTITY` units.
5. **User and Order**: Users place orders.
6. **Order and OrderItem**: Orders consist of multiple items, supporting both weight and quantity.
7. **Order and DeliveryStatus**: Each order tracks delivery progress with a linked delivery boy.

---

4. Features Supported by Schema

- Admin can add restaurants and food items, specifying unit type (`KG`, `QUANTITY`, or `BOTH`).

- Stock levels for food items are directly managed in the `FoodItem` table via the `available_stock`, `weight_kg`, and `quantity` fields.
- Food items can support sales by weight, quantity, or both.
- Customers can place orders for food from open restaurants.
- Stock is updated when an order is placed, either for weight or quantity.
- Delivery boys are assigned orders, and their statuses are tracked.
- A single `User` table simplifies user management by differentiating roles using the `role` field.
- The schema is flexible for future enhancements like promotions, reviews, and payment integrations.

Entities and Their relation:

JPA Entities and Relationships for the Food Delivery App

Here are the JPA entities based on the schema described. Each entity includes annotations to define relationships and constraints.

User Entity

```
@Entity
@Table(name = "User")
public class User {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    private String name;
    private String email;
    private String password;
    private String phoneNumber;
    private String address;
```

```
    @Enumerated(EnumType.STRING)
    private Role role; // Enum: ADMIN, CUSTOMER, DELIVERY_B
OY

    private Boolean isAvailable = true; // Applicable for d
elivery boys

    @CreationTimestamp
    private LocalDateTime createdAt;

    // Getters and Setters
}

public enum Role {
    ADMIN, CUSTOMER, DELIVERY_BOY
}
```

Restaurant Entity

```
@Entity
@Table(name = "Restaurant")
public class Restaurant {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    private String name;
    private String address;

    private LocalTime openTime;
    private LocalTime closeTime;

    private Boolean isOpen = true;
```

```
    @ManyToOne
    @JoinColumn(name = "created_by")
    private User admin; // The admin who created the restau
rant

    @CreationTimestamp
    private LocalDateTime createdAt;

    @OneToMany(mappedBy = "restaurant", cascade = CascadeTy
pe.ALL, orphanRemoval = true)
    private List<FoodItem> foodItems;

    // Getters and Setters
}
```

FoodItem Entity

```
@Entity
@Table(name = "FoodItem")
public class FoodItem {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    private String name;
    private String description;

    private BigDecimal basePrice;

    @Enumerated(EnumType.STRING)
    private UnitType unit; // Enum: KG, QUANTITY, BOTH

    private String variationName; // E.g., "Small", "1kg"

    private BigDecimal weightKg; // For items sold by weigh
```

```
t
    private Integer quantity; // For items sold by quantity

    private BigDecimal availableStock; // Current stock lev
el

    @ManyToOne
    @JoinColumn(name = "restaurant_id")
    private Restaurant restaurant;

    @CreationTimestamp
    private LocalDateTime createdAt;

    // Getters and Setters
}

public enum UnitType {
    KG, QUANTITY, BOTH
}
```

Order Entity

```
@Entity
@Table(name = "Order")
public class Order {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @ManyToOne
    @JoinColumn(name = "user_id")
    private User user; // The customer placing the order

    @ManyToOne
    @JoinColumn(name = "restaurant_id")
```

```
    private Restaurant restaurant;

    private BigDecimal totalPrice;

    @Enumerated(EnumType.STRING)
    private OrderStatus status = OrderStatus.PLACED; // Enu
m: PLACED, PREPARING, OUT_FOR_DELIVERY, DELIVERED, CANCELLE
D

    @CreationTimestamp
    private LocalDateTime orderTime;

    @OneToMany(mappedBy = "order", cascade = CascadeType.AL
L, orphanRemoval = true)
    private List<OrderItem> orderItems;

    // Getters and Setters
}

public enum OrderStatus {
    PLACED, PREPARING, OUT_FOR_DELIVERY, DELIVERED, CANCELL
ED
}
```

OrderItem Entity

```
@Entity
@Table(name = "OrderItem")
public class OrderItem {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @ManyToOne
    @JoinColumn(name = "order_id")
```

```
    private Order order;

    @ManyToOne
    @JoinColumn(name = "food_item_id")
    private FoodItem foodItem;

    @Enumerated(EnumType.STRING)
    private UnitType unit; // Enum: KG, QUANTITY

    private BigDecimal quantity; // Weight or quantity of t
he food item

    private BigDecimal price; // Total price for this item

    // Getters and Setters
}
```

DeliveryStatus Entity

```
@Entity
@Table(name = "DeliveryStatus")
public class DeliveryStatus {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @ManyToOne
    @JoinColumn(name = "order_id")
    private Order order;

    @ManyToOne
    @JoinColumn(name = "delivery_boy_id")
    private User deliveryBoy; // Delivery boy assigned to t
he order
```

```
    @Enumerated(EnumType.STRING)
    private DeliveryStatusEnum status = DeliveryStatusEnum.
ASSIGNED; // Enum: ASSIGNED, PICKED_UP, DELIVERED, CANCELLE
D

    @UpdateTimestamp
    private LocalDateTime updatedAt;

    // Getters and Setters
}

public enum DeliveryStatusEnum {
    ASSIGNED, PICKED_UP, DELIVERED, CANCELLED
}
```

Relationships Summary

1. **User**:
   - Differentiates admin, customers, and delivery boys using the `role` field.
   - Delivery boys' availability is tracked with the `isAvailable` field.
2. **Restaurant**:
   - Linked to the admin who created it.
   - Maintains a list of food items it offers.
3. **FoodItem**:
   - Tracks whether items are sold by weight, quantity, or both.
   - Belongs to a specific restaurant.
4. **Order and OrderItem**:
   - `Order` tracks the overall order details.
   - `OrderItem` specifies the food items in the order, including their unit type and quantity/weight.
5. **DeliveryStatus**:
   - Tracks the status of delivery, linked to an order and a delivery boy.

Role Enum

Used in the `User` entity to differentiate between roles.

```
java
Copy code
public enum Role {
    ADMIN,        // Represents an administrator who manage
s restaurants and food items
    CUSTOMER,     // Represents a customer who places order
s
    DELIVERY_BOY  // Represents a delivery boy responsible
for delivering orders
}
```

UnitType Enum

Used in the `FoodItem` and `OrderItem` entities to specify the unit of sale.

```
public enum UnitType {
    KG,           // Represents items sold by weight (in ki
lograms)
    QUANTITY,     // Represents items sold by quantity
    BOTH          // Represents items sold by both weight a
nd quantity
}
```

OrderStatus Enum

Used in the `Order` entity to track the status of an order.

```
public enum OrderStatus {
```

```
    PLACED,           // Order has been placed by the custo
mer
    PREPARING,        // Order is being prepared by the res
taurant
    OUT_FOR_DELIVERY, // Order has been dispatched for deli
very
    DELIVERED,        // Order has been delivered to the cu
stomer
    CANCELLED         // Order has been cancelled
}
```

DeliveryStatusEnum

Used in the `DeliveryStatus` entity to track the status of a delivery.

```
java
Copy code
public enum DeliveryStatusEnum {
    ASSIGNED,    // Delivery boy has been assigned to the o
rder
    PICKED_UP,   // Delivery boy has picked up the order fr
om the restaurant
    DELIVERED,   // Order has been delivered to the custome
r
    CANCELLED    // Delivery has been cancelled
}
```