**LTMOBLIE\_report\_1**

**Ngữ cảnh**

Ứng dụng bán đồ ăn sáng online freeship hỏa tốc FastBreakfast, khách hàng chỉ việc tải ứng dụng, đặt hàng tại nhà, chờ đồ ăn đến, và sau đó thưởng thức.

**Yêu cầu khách hàng**

Muốn một ứng dụng khi mở lên là hiển thị trang các món ăn (trang chính), chưa cần đăng nhập, nếu đăng nhập rồi thì lưu đăng nhập cho người dùng, có thể đăng xuất. Mỗi người dùng sẽ có 1 giỏ hàng cho riêng họ.

Từ trang chính, có thể tìm kiếm món ăn hoặc lọc theo giá, mỗi sản phẩm có thể chọn và thêm vào giỏ hàng, khi thêm món ăn vào giỏ hàng sẽ đi kèm với giá của món đó và số lượng món (cái này do khách hàng chọn), món ăn kèm với giá và số lượng của nó trong giỏ hàng như vậy gọi là một mục.

Vào giỏ hàng để xem các mục trong giỏ, có thể xóa mục, hoặc sửa giá (các giá có sẵn, chọn size khác nên giá đồ ăn khác), sửa thuộc tính số lượng sản phẩm của 1 mục trong giỏ, có thể nhấn vào mục để đến trang xem sản phẩm đó (trang này không khác gì lúc họ xem, tìm kiếm sản phẩm). Chọn các mục để tạo hóa đơn, áp dụng mã giảm giá (của sản phẩm, của khách hàng, nếu có), tự nhập sẵn sdt và địa chỉ từ profile người dùng và cũng cho phép chỉnh sửa mới sdt và địa chỉ riêng cho đơn hàng đó, tạo xong đơn hàng rồi, nếu chưa thanh toán thì có thể xóa đơn hàng, nếu xóa, mọi thông tin trước khi tạo đơn hàng sẽ được hoàn tác, thanh toán để bắt đầu giao hàng lúc này đơn hàng không thể hủy nữa. Các giai đoạn tạo CartItem, OrderItem, và thanh toán đều phải kiểm tra hàng trong kho có đủ hay không. (App không nhận tiền mặt, nếu muốn nhận tiền mặt thì cần cho phép Quản lý sửa trạng thái và số tiền thanh toán của đơn hàng một cách thủ công, điều này là một tính năng khác, cân nhắc nâng cấp).

**Đặc tả tóm tắt (tech)**

Một Food phải thuộc về duy nhất một Category và một FoodSize.

Food (có Category, thêm FoodSize, thêm quantity) -> CartItem (thêm DiscountCode hệ thống) -> OrderItem, OrderItem chỉ khác CartItem về giá

Hay là FoodItem có thuộc tính size luôn cho đơn giản?

Một FbfUser phải có !1 Cart và 1 Cart chỉ thuộc về !1 FbfUser.

Cart chứa 0, 1 hoặc n CartItem, và 1 CartItem phải thuộc về !1 Cart.

FbfUser có thể thêm 1 hoặc n CartItem vào trong Cart của họ. (=1+2?)

Một FbfOrder phải chứa 1 hoặc n OrderItem.

FbfUser có thể tạo 1 hoặc n FbfOrder.

FbfUser có thể thanh toán 1 hoặc n FbfOrder.

1 DiscountCode có thể được áp dụng lên 1 OrderItem sau khi được tạo ra từ CartItem (không áp dụng lên CartItem, nhưng trong Cart vẫn hiển thị cho user giá đã giảm khi xem giỏ hàng)

DiscountCode có thể được áp dụng lên FbfOrder bởi FbfUser.

DiscountCode áp dụng lên FbfOrder bởi FbfUser được tặng cho FbfUser tự động bởi hệ thống dựa trên lịch sử mua hàng.

**Bảng sơ**

FbfUser(id, username, password, email, name, phone\_number, address)

Food(id, name, price, description, image\_url)

FoodSize(id, size\_name(Small, Medium, Large, Extra Large))

Category(id, name)

Cart(id, fbf\_user\_id)

CartItem(id, price, quantity)

FbfOrder(id, total\_price, phone\_number, address, created\_at , status(created, paid))

OrderItem(id, price, quantity)

DiscountCode(id, code, discount\_percentage, expiration\_date, expired)

*Thêm khóa*

FbfUser(id, username, password, email, name, phone\_number, address)

Food(id, name, price, description, image\_url, food\_size\_id, category\_id)

FoodSize(id, size\_name(Small, Medium, Large, Extra Large))

Category(id, name)

Cart(id, fbf\_user\_id)

CartItem(id, price, quantity, food\_id, cart\_id)

FbfOrder(id, total\_price, phone\_number, address, created\_at , status(created, paid), fbf\_user\_id, …)

OrderItem(id, price, quantity, fbf\_order\_id, cart\_item\_id)

DiscountCode(id, code, discount\_percentage, expiration\_date, expired, order\_item\_id(nullable), fbf\_order\_id(nullable))

**Spring boot code**

@Entity

@Table(name = "fbf\_user")

public class FbfUser {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

@Column(nullable = false, unique = true)

private String username;

@Column(nullable = false)

private String password;

@Column(nullable = false, unique = true)

private String email;

private String name;

private String phoneNumber;

private String address;

}

@Entity

@Table(name = "food")

public class Food {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

@Column(nullable = false)

private String name;

@Column(nullable = false)

private Double price;

private String description;

private String imageUrl;

@ManyToOne

@JoinColumn(name = "food\_size\_id", nullable = false)

private FoodSize foodSize;

@ManyToOne

@JoinColumn(name = "category\_id", nullable = false)

private Category category;

}

@Entity

@Table(name = "food\_size")

public class FoodSize {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

@Column(nullable = false, unique = true)

private String sizeName; // Small, Medium, Large, Extra Large

}

@Entity

@Table(name = "category")

public class Category {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

@Column(nullable = false, unique = true)

private String name;

}

@Entity

@Table(name = "cart")

public class Cart {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

@OneToOne

@JoinColumn(name = "fbf\_user\_id", nullable = false, unique = true)

private FbfUser user;

}

@Entity

@Table(name = "cart\_item")

public class CartItem {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

@Column(nullable = false)

private Double price;

@Column(nullable = false)

private Integer quantity;

@ManyToOne

@JoinColumn(name = "food\_id", nullable = false)

private Food food;

@ManyToOne

@JoinColumn(name = "cart\_id", nullable = false)

private Cart cart;

}

@Entity

@Table(name = "fbf\_order")

public class FbfOrder {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

@Column(nullable = false)

private Double totalPrice;

@Column(nullable = false)

private String phoneNumber;

@Column(nullable = false)

private String address;

@Column(nullable = false)

@Temporal(TemporalType.TIMESTAMP)

private Date createdAt;

@Enumerated(EnumType.STRING)

@Column(nullable = false)

private OrderStatus status; // CREATED, PAID

@ManyToOne

@JoinColumn(name = "fbf\_user\_id", nullable = false)

private FbfUser user;

}

public enum OrderStatus {

CREATED, PAID

}

@Entity

@Table(name = "order\_item")

public class OrderItem {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

@Column(nullable = false)

private Double price;

@Column(nullable = false)

private Integer quantity;

@ManyToOne

@JoinColumn(name = "fbf\_order\_id", nullable = false)

private FbfOrder order;

@ManyToOne

@JoinColumn(name = "cart\_item\_id", nullable = false)

private CartItem cartItem;

}

@Entity

@Table(name = "discount\_code")

public class DiscountCode {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

@Column(nullable = false, unique = true)

private String code;

@Column(nullable = false)

private Double discountPercentage;

@Temporal(TemporalType.TIMESTAMP)

private Date expirationDate;

@Column(nullable = false)

private Boolean expired;

@ManyToOne

@JoinColumn(name = "order\_item\_id", nullable = true)

private OrderItem orderItem;

@ManyToOne

@JoinColumn(name = "fbf\_order\_id", nullable = true)

private FbfOrder order;

}