**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**

*Đăng nhập*

Muốn một ứng dụng khi mở lên là trang chính hiển thị các món ăn (Food), chưa cần đăng nhập, nếu đăng nhập rồi thì lưu đăng nhập cho người dùng (User), có thể đăng xuất. Mỗi người dùng sẽ có 1 giỏ hàng (Cart) cho riêng họ. Khách có thể chỉnh sửa thông tin cá nhân như sdt, địa chỉ.

*Tìm kiếm và thêm giỏ hàng*

Từ trang chính, có thể tìm kiếm món ăn theo tên hoặc lọc/xếp theo giá, mỗi mục sản phẩm có thể chọn để đến trang chi tiết và cũng có nút thêm vào giỏ hàng, thao tác thêm giỏ hàng phải **nhanh gọn** **tiện** bằng ít nhất là 1-2 click, trong khi thêm món ăn vào giỏ hàng sẽ đi kèm với size và số lượng của món đó và tất nhiên khách hàng có thể sửa nếu muốn. Món ăn kèm với size và số lượng của nó trong giỏ hàng như vậy gọi là một mục (CartItem).

*Quản lý giỏ hàng*

Vào giỏ hàng để xem các mục trong giỏ, có thể xóa mục, sửa size, số lượng của mục (các giá cố định sẵn, hiển thị thêm giá được giảm nếu có, chọn size khác thì giá khác), có thể nhấn vào mục để đến trang chi tiết xem sản phẩm đó giống lúc thêm vào giỏ đã nói. Chọn các mục và nhấn tạo đơn hàng (OrderItem), nó sẽ tự mở ra trang có tự nhập sẵn sdt và địa chỉ từ profile người dùng và cũng cho phép chỉnh sửa lại sdt và địa chỉ riêng cho đơn hàng đó; áp dụng mã giảm giá (mã của sản phẩm, của khách hàng, nếu có), bên dưới là nút *‘Tạo đơn hàng và thanh toán’*. Đơn hàng sau khi tạo phải được thanh toán ngay để bắt đầu giao hàng, đơn hàng sau khi thanh toán không thể hủy. Nếu không thanh toán sau khi nhấn tạo đơn hàng, đơn hàng sẽ bị hủy tự động và mọi thứ trong giỏ hàng sẽ quay về như lúc chưa nhấn tạo đơn hàng. Các giai đoạn tạo CartItem, OrderItem, và thanh toán đều phải kiểm tra hàng trong kho có đủ hay không. Một trick để đảm bảo điều kiện trên là *Chỉ tạo đơn hàng sau khi đã thanh toán,* hay nói cách khác là không có đơn hàng nào chưa thanh toán được tồn tại, và *FbfOrder cũng đóng vai trò như là hóa đơn luôn*.(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? YES

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 thông tin không có khóa**

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

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

FoodSize(id, size, price)

Category(id, name)

Cart(id, fbf\_user\_id)

CartItem(id, price, quantity)

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

OrderItem(id, price, quantity)

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

// Payment(id, amount)

*Thêm khóa*

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

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

FoodSize(id, size(S, M, L, XL), price, food\_id)

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, fbf\_user\_id)

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

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

// Payment(id, amount, fbf\_user\_id, fbf\_order\_id)

**Spring boot code**

import jakarta.persistence.\*;

@Entity

public class FbfUser {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String username;

private String password;

private String email;

private String name;

private String phoneNumber;

private String address;

@OneToOne(mappedBy = "user")

private Cart cart;

@OneToMany(mappedBy = "user")

private List<FbfOrder> orders;

// Getters and setters

public Long getId() { return id; }

public void setId(Long id) { this.id = id; }

public String getUsername() { return username; }

public void setUsername(String username) { this.username = username; }

public String getPassword() { return password; }

public void setPassword(String password) { this.password = password; }

public String getEmail() { return email; }

public void setEmail(String email) { this.email = email; }

public String getName() { return name; }

public void setName(String name) { this.name = name; }

public String getPhoneNumber() { return phoneNumber; }

public void setPhoneNumber(String phoneNumber) { this.phoneNumber = phoneNumber; }

public String getAddress() { return address; }

public void setAddress(String address) { this.address = address; }

}

import jakarta.persistence.\*;

@Entity

public class Category {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String name;

// Getters and setters

public Long getId() { return id; }

public void setId(Long id) { this.id = id; }

public String getName() { return name; }

public void setName(String name) { this.name = name; }

}

import jakarta.persistence.\*;

@Entity

public class Food {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String name;

private String description;

private String imageUrl;

@ManyToOne

@JoinColumn(name = "category\_id")

private Category category;

@OneToMany(mappedBy = "food")

private List<FoodSize> sizes;

// Getters and setters

public Long getId() { return id; }

public void setId(Long id) { this.id = id; }

public String getName() { return name; }

public void setName(String name) { this.name = name; }

public String getDescription() { return description; }

public void setDescription(String description) { this.description = description; }

public String getImageUrl() { return imageUrl; }

public void setImageUrl(String imageUrl) { this.imageUrl = imageUrl; }

public Category getCategory() { return category; }

public void setCategory(Category category) { this.category = category; }

}

import jakarta.persistence.\*;

@Entity

public class FoodSize {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String size; // e.g., "S", "M", "L", "XL"

private Double price;

private Double discountPercentage; // e.g., 10.0 for 10%, null if no discount

@ManyToOne

@JoinColumn(name = "food\_id")

private Food food;

// Getters and setters

public Long getId() { return id; }

public void setId(Long id) { this.id = id; }

public String getSize() { return size; }

public void setSize(String size) { this.size = size; }

public Double getPrice() { return price; }

public void setPrice(Double price) { this.price = price; }

public Double getDiscountPercentage() { return discountPercentage; }

public void setDiscountPercentage(Double discountPercentage) { this.discountPercentage = discountPercentage; }

public Food getFood() { return food; }

public void setFood(Food food) { this.food = food; }

}

import jakarta.persistence.\*;

@Entity

public class Cart {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

@OneToOne

@JoinColumn(name = "fbf\_user\_id")

private FbfUser user;

@OneToMany(mappedBy = "cart")

private List<CartItem> items;

// Getters and setters

public Long getId() { return id; }

public void setId(Long id) { this.id = id; }

public FbfUser getUser() { return user; }

public void setUser(FbfUser user) { this.user = user; }

}

import jakarta.persistence.\*;

@Entity

public class CartItem {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

@ManyToOne

@JoinColumn(name = "food\_size\_id")

private FoodSize foodSize;

private Double price; // Price at the time of adding

private Double discountPercentage; // Discount at the time of adding, null if none

private Integer quantity;

@ManyToOne

@JoinColumn(name = "cart\_id")

private Cart cart;

// Getters and setters

public Long getId() { return id; }

public void setId(Long id) { this.id = id; }

public FoodSize getFoodSize() { return foodSize; }

public void setFoodSize(FoodSize foodSize) { this.foodSize = foodSize; }

public Double getPrice() { return price; }

public void setPrice(Double price) { this.price = price; }

public Double getDiscountPercentage() { return discountPercentage; }

public void setDiscountPercentage(Double discountPercentage) { this.discountPercentage = discountPercentage; }

public Integer getQuantity() { return quantity; }

public void setQuantity(Integer quantity) { this.quantity = quantity; }

public Cart getCart() { return cart; }

public void setCart(Cart cart) { this.cart = cart; }

}

import jakarta.persistence.\*;

import java.time.LocalDateTime;

@Entity

public class FbfOrder {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private Double totalPrice;

private String phoneNumber;

private String address;

private LocalDateTime createdAt;

@ManyToOne

@JoinColumn(name = "fbf\_user\_id")

private FbfUser user;

@ManyToOne

@JoinColumn(name = "discount\_code\_id")

private DiscountCode discountCode;

@OneToMany(mappedBy = "order")

private List<OrderItem> items;

// Getters and setters

public Long getId() { return id; }

public void setId(Long id) { this.id = id; }

public Double getTotalPrice() { return totalPrice; }

public void setTotalPrice(Double totalPrice) { this.totalPrice = totalPrice; }

public String getPhoneNumber() { return phoneNumber; }

public void setPhoneNumber(String phoneNumber) { this.phoneNumber = phoneNumber; }

public String getAddress() { return address; }

public void setAddress(String address) { this.address = address; }

public LocalDateTime getCreatedAt() { return createdAt; }

public void setCreatedAt(LocalDateTime createdAt) { this.createdAt = createdAt; }

public FbfUser getUser() { return user; }

public void setUser(FbfUser user) { this.user = user; }

public DiscountCode getDiscountCode() { return discountCode; }

public void setDiscountCode(DiscountCode discountCode) { this.discountCode = discountCode; }

}

import jakarta.persistence.\*;

@Entity

public class OrderItem {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

@ManyToOne

@JoinColumn(name = "food\_size\_id")

private FoodSize foodSize;

private Double price;

private Double discountPercentage; // Copied from CartItem

private Integer quantity;

@ManyToOne

@JoinColumn(name = "fbf\_order\_id")

private FbfOrder order;

// Getters and setters

public Long getId() { return id; }

public void setId(Long id) { this.id = id; }

public FoodSize getFoodSize() { return foodSize; }

public void setFoodSize(FoodSize foodSize) { this.foodSize = foodSize; }

public Double getPrice() { return price; }

public void setPrice(Double price) { this.price = price; }

public Double getDiscountPercentage() { return discountPercentage; }

public void setDiscountPercentage(Double discountPercentage) { this.discountPercentage = discountPercentage; }

public Integer getQuantity() { return quantity; }

public void setQuantity(Integer quantity) { this.quantity = quantity; }

public FbfOrder getOrder() { return order; }

public void setOrder(FbfOrder order) { this.order = order; }

}

import jakarta.persistence.\*;

import java.time.LocalDateTime;

@Entity

public class DiscountCode {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String code;

private Double discountPercentage;

private LocalDateTime expirationDate; // Null if it never expires

// Getters and setters

public Long getId() { return id; }

public void setId(Long id) { this.id = id; }

public String getCode() { return code; }

public void setCode(String code) { this.code = code; }

public Double getDiscountPercentage() { return discountPercentage; }

public void setDiscountPercentage(Double discountPercentage) { this.discountPercentage = discountPercentage; }

public LocalDateTime getExpirationDate() { return expirationDate; }

public void setExpirationDate(LocalDateTime expirationDate) { this.expirationDate = expirationDate; }

}