Class Diagram Breakdown:

1. **Customer**
    - Attributes:
        - id: int
        - name: String
        - email: String
        - address: String
    - Methods:
        - addToCart(item: Item): void
        - removeFromCart(item: Item): void
        - checkout(): Order
2. **Cart**
    - Attributes:
        - idCart: int
        - idCustomer: int (foreign key to Customer)
        - items: List<Item> (composite relationship with `Item`)
    - Methods:
        - addItem(item: Item): void
        - removeItem(item: Item): void
        - calculateTotal(): double
3. **Item (Abstract Class)**
    - Attributes:
        - idItem: int
        - name: String
        - price: double
    - Methods:
        - getPrice(): double
        - getName(): String
4. **Book (Extends Item)**
    - Attributes:
        - author: String
        - ISBN: String
        - genre: String
    - Methods:
        - getDetails(): String
5. **Clothes (Extends Item)**
    - Attributes:
        - size: String
        - color: String
    - Methods:
        - getDetails(): String
6. **Order**
    - Attributes:
        - idOrder: int
        - customer: Customer (association with `Customer`)

        - cart: Cart (association with `Cart`)
        - shipment: Shipment
        - payment: Payment
    - Methods:
        - placeOrder(): void
        - cancelOrder(): void
        - getOrderDetails(): String
7. **Shipment**
    - Attributes:
        - idShipment: int
        - address: String
        - status: String
    - Methods:
        - updateStatus(status: String): void
        - getShipmentDetails(): String
8. **Payment**
    - Attributes:
        - idPayment: int
        - amount: double
        - method: String
    - Methods:
        - processPayment(): boolean
        - refundPayment(): boolean
9. **Comment**
    - Attributes:
        - idComment: int
        - text: String
        - customer: Customer (association with `Customer`)
        - item: Item (association with `Item`)
    - Methods:
        - addComment(): void
        - editComment(): void
        - deleteComment(): void
10. **Rating**
    - Attributes:
        - idRating: int
        - stars: int
        - customer: Customer (association with `Customer`)
        - item: Item (association with `Item`)
    - Methods:
        - addRating(): void
        - updateRating(): void
        - getRating(): int

Relationships:

- **Customer** ↔ **Cart**: A Customer can have one Cart, and a Cart is associated with one Customer.
- **Cart** ↔ **Item**: A Cart can contain multiple Items (Book, Clothes), and each Item can belong to many Carts.
- **Order** ↔ **Customer**: An Order contains Customer information.
- **Order** ↔ **Cart**: An Order contains the Cart.
- **Order** ↔ **Shipment**: An Order is associated with Shipment details.
- **Order** ↔ **Payment**: An Order is associated with Payment details.
- **Comment** ↔ **Customer** ↔ **Item**: A Customer can comment on Items.
- **Rating** ↔ **Customer** ↔ **Item**: A Customer can rate Items.

1. Customer ↔ Cart

- **Type**: Association
- **Cardinality**: 1 (Customer) to 1 (Cart)
- **Description**: A customer has one cart, and the cart belongs to one customer.

2. Cart ↔ Item (abstract)

- **Type**: Composition
- **Cardinality**: 1 (Cart) to many (Items)
- **Description**: A cart contains multiple items, and if the cart is destroyed, the items in that cart are removed from it.

3. Item (abstract) ↔ Book/Clothes

- **Type**: Generalization (Inheritance)
- **Description**: `Book` and `Clothes` are specialized classes that inherit from the abstract `Item` class.

4. Order ↔ Customer

- **Type**: Association
- **Cardinality**: 1 (Order) to 1 (Customer)
- **Description**: An order contains customer information, but a customer can place multiple orders.

5. Order ↔ Cart

- **Type**: Aggregation
- **Cardinality**: 1 (Order) to 1 (Cart)
- **Description**: An order contains a cart, but the cart can exist independently from an order.

6. Order ↔ Shipment

- **Type**: Association
- **Cardinality**: 1 (Order) to 1 (Shipment)

- **Description**: An order is associated with one shipment.

7. Order ↔ Payment

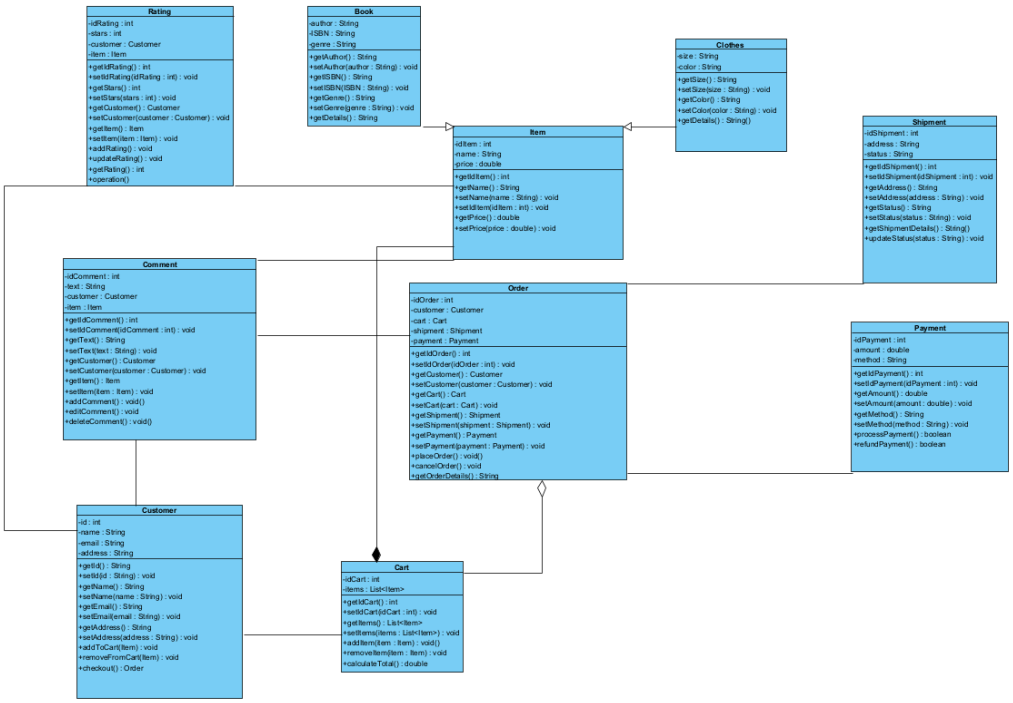
- **Type**: Dependency
- **Cardinality**: 1 (Order) to 1 (Payment)
- **Description**: An order depends on payment to be processed, but payment can exist independently.

8. Customer ↔ Comment ↔ Item

- **Type**: Association
- **Cardinality**: 1 (Customer) to many (Comments), 1 (Item) to many (Comments)
- **Description**: A customer can comment on multiple items, and an item can have multiple comments.

9. Customer ↔ Rating ↔ Item

- **Type**: Association
- **Cardinality**: 1 (Customer) to many (Ratings), 1 (Item) to many (Ratings)
- **Description**: A customer can rate multiple items, and an item can receive ratings from many customers.



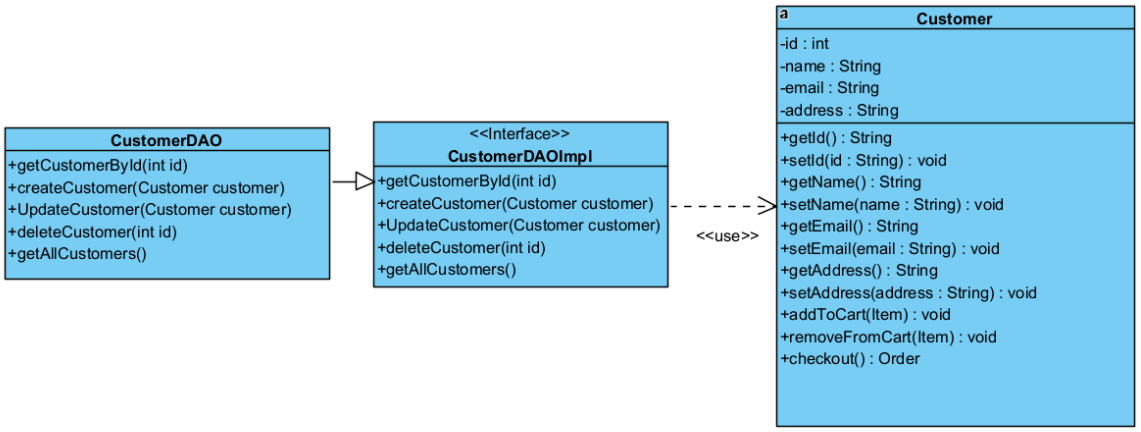
DAO Design Breakdown

---

1. CustomerDAO (for `Customer`)

The **CustomerDAO** is responsible for interacting with the database for customer-related operations.

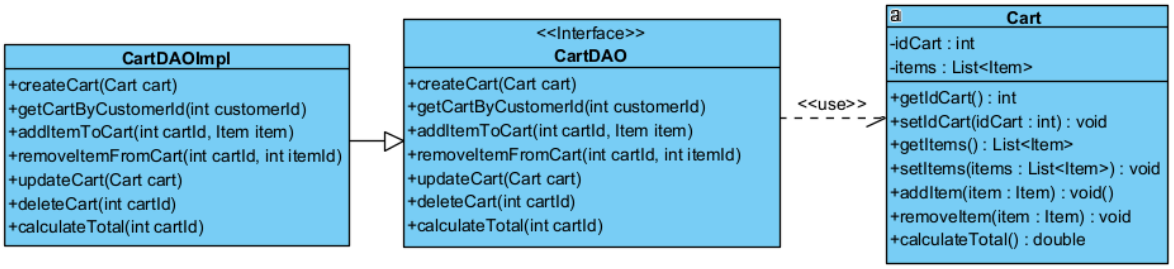
- **DAO Methods**:
  - `createCustomer(Customer customer)`: Adds a new customer to the database.
  - `getCustomerById(int id)`: Retrieves customer details based on their ID.
  - `updateCustomer(Customer customer)`: Updates customer details.
  - `deleteCustomer(int id)`: Removes a customer from the database.
  - `getAllCustomers()`: Retrieves a list of all customers.



2. CartDAO (for `Cart`)

The **CartDAO** manages database operations related to the `Cart` entity, including linking it to `Customer` and `Item`.

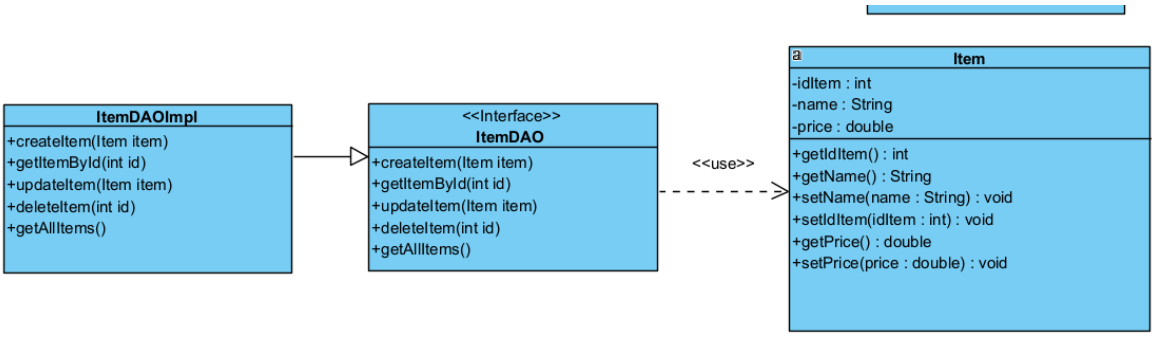
- **DAO Methods**:
  - `createCart(Cart cart)`: Creates a new cart.
  - `getCartByCustomerId(int customerId)`: Retrieves a cart based on the associated customer ID.
  - `addItemToCart(int cartId, Item item)`: Adds an item to the cart.
  - `removeItemFromCart(int cartId, int itemId)`: Removes an item from the cart.
  - `updateCart(Cart cart)`: Updates the cart details.
  - `deleteCart(int cartId)`: Deletes a cart.
  - `calculateTotal(int cartId)`: Calculates the total value of the items in the cart.



3. ItemDAO (for `Item`, `Book`, `Clothes`)

The **ItemDAO** handles the persistence of items. Since `Item` is abstract, the DAO will manage its concrete subclasses (`Book` and `Clothes`).

- **DAO Methods**:
  - `createItem(Item item)`: Adds an item to the database (can apply to `Book` or `Clothes`).
  - `getItemById(int id)`: Retrieves an item by ID.
  - `updateItem(Item item)`: Updates item details.
  - `deleteItem(int id)`: Deletes an item.
  - `getAllItems()`: Retrieves a list of all items (can filter by `Book` or `Clothes`).

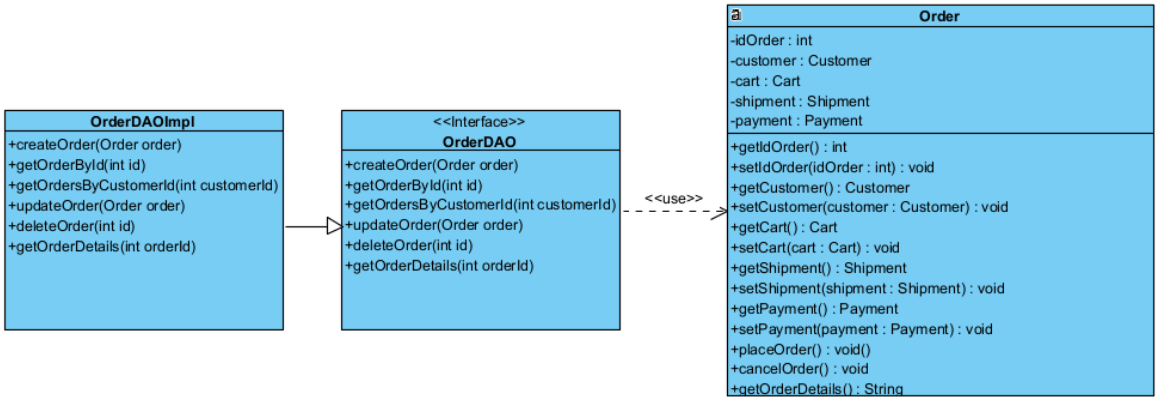


4. OrderDAO (for `Order`)

The **OrderDAO** handles database operations related to `Order`, including linking to `Customer`, `Cart`, `Shipment`, and `Payment`.

- **DAO Methods**:
  - `createOrder(Order order)`: Creates a new order.
  - `getOrderById(int id)`: Retrieves an order by ID.
  - `getOrdersByCustomerId(int customerId)`: Retrieves all orders associated with a customer.

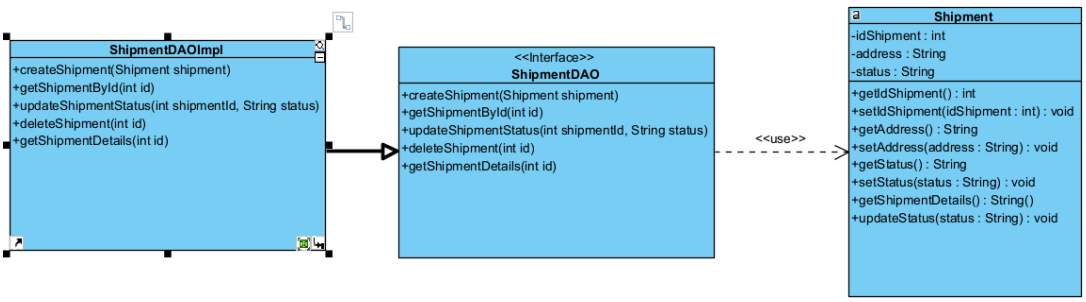
- `updateOrder(Order order)`: Updates order details (e.g., change cart, shipment status, etc.).
- `deleteOrder(int id)`: Deletes an order.
- `getOrderDetails(int orderId)`: Retrieves the detailed information of an order.



5. ShipmentDAO (for `Shipment`)

The **ShipmentDAO** manages the database operations for the `Shipment` entity.

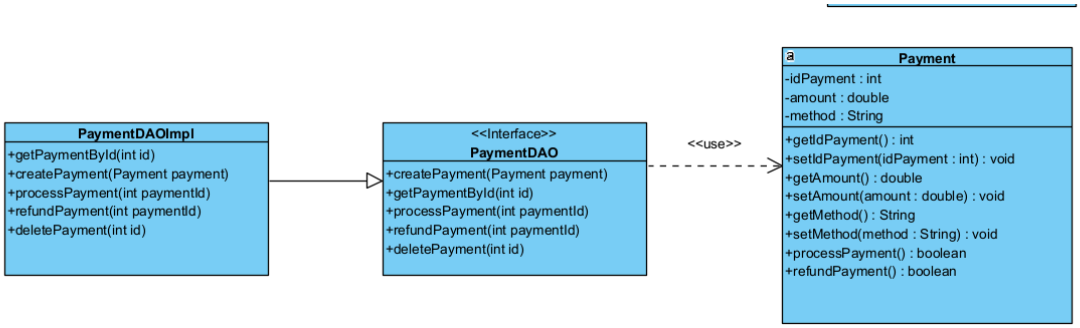
- **DAO Methods**:
  - `createShipment(Shipment shipment)`: Creates a new shipment entry.
  - `getShipmentById(int id)`: Retrieves shipment details by ID.
  - `updateShipmentStatus(int shipmentId, String status)`: Updates the shipment status.
  - `deleteShipment(int id)`: Deletes a shipment.
  - `getShipmentDetails(int id)`: Retrieves the shipment details.



6. PaymentDAO (for `Payment`)

The **PaymentDAO** handles the database interactions related to `Payment` and its association with `Order`.

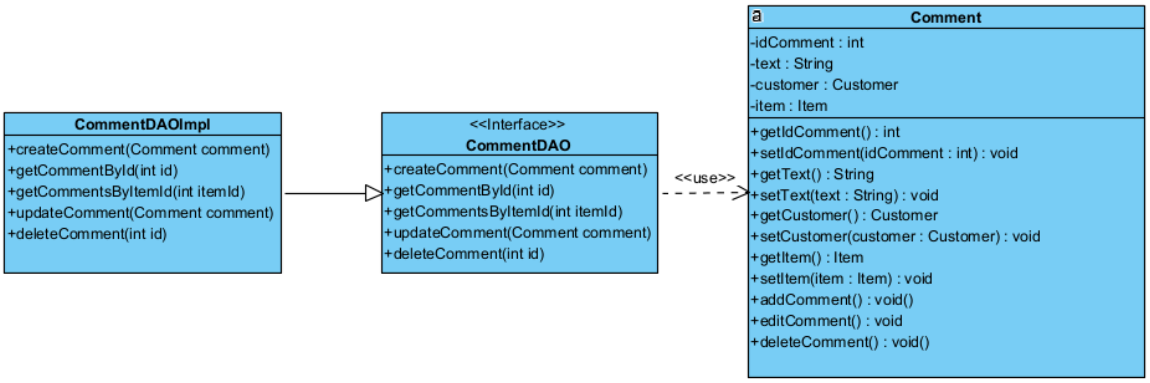
- **Attributes**:
  - idPayment: int
  - amount: double
  - method: String
- **DAO Methods**:
  - `createPayment(Payment payment)`: Adds a new payment.
  - `getPaymentById(int id)`: Retrieves payment details by ID.
  - `processPayment(int paymentId)`: Processes the payment and returns a success/failure status.
  - `refundPayment(int paymentId)`: Refunds a payment.
  - `deletePayment(int id)`: Deletes a payment record.



7. CommentDAO (for `Comment`)

The **CommentDAO** manages the persistence of comments related to `Item` and `Customer`.

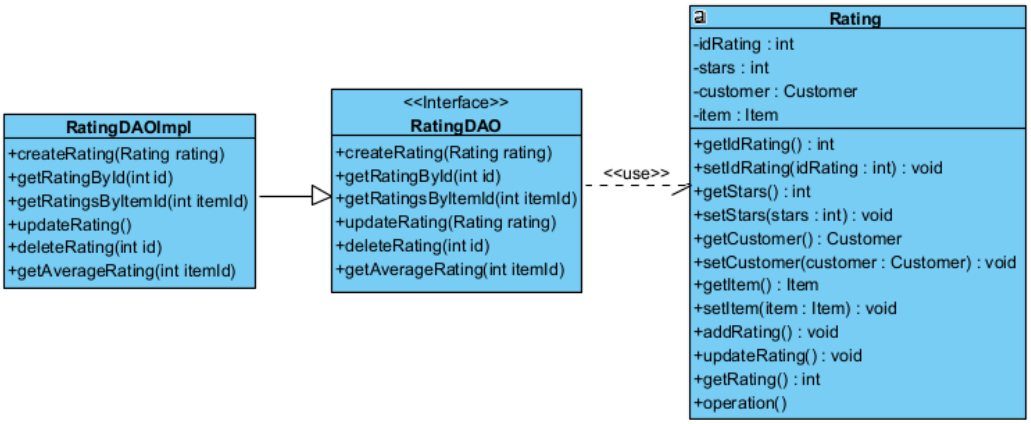
- **DAO Methods**:
  - `createComment(Comment comment)`: Adds a new comment.
  - `getCommentById(int id)`: Retrieves a comment by ID.
  - `getCommentsByItemId(int itemId)`: Retrieves all comments for a specific item.
  - `updateComment(Comment comment)`: Updates the text of an existing comment.
  - `deleteComment(int id)`: Deletes a comment.

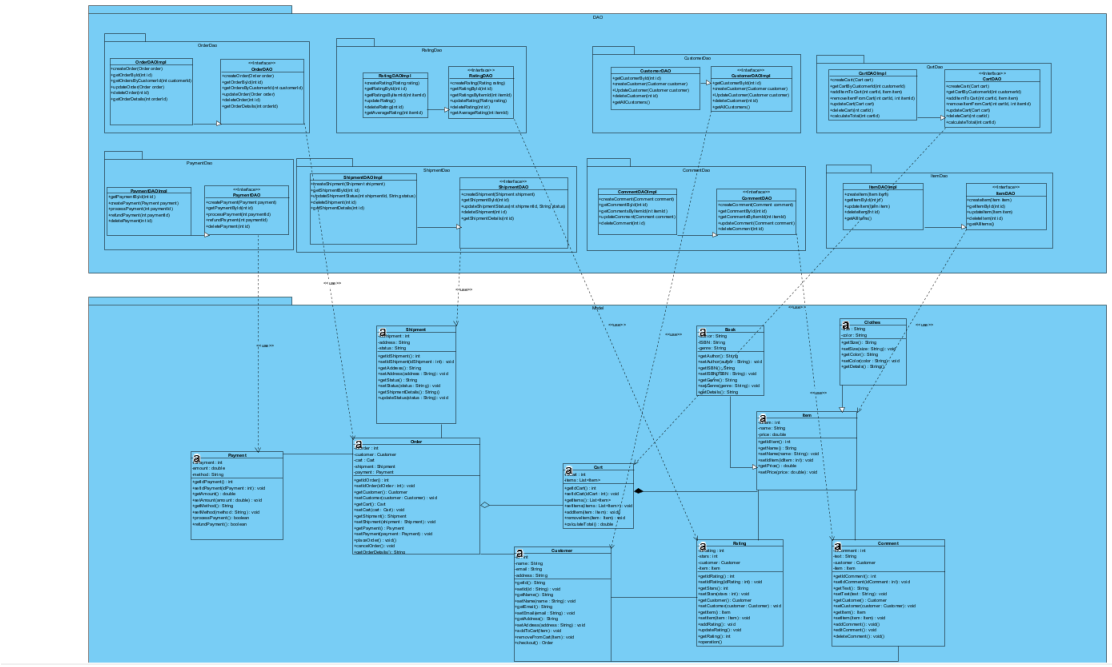


8. RatingDAO (for `Rating`)

The **RatingDAO** handles the database operations related to ratings given by customers on items.

- **DAO Methods**:
  - `createRating(Rating rating)`: Adds a new rating.
  - `getRatingById(int id)`: Retrieves a rating by ID.
  - `getRatingsByItemId(int itemId)`: Retrieves all ratings for a specific item.
  - `updateRating(Rating rating)`: Updates the star rating.
  - `deleteRating(int id)`: Deletes a rating.
  - `getAverageRating(int itemId)`: Calculates the average rating of an item.





Folder Strucure

ecommerce-system/

```
|
├── src/
|   ├── main/
|   |   ├── java/
|   |   |   └── com/
|   |   |       └── ecommerce/
|   |   |           ├── controller/
|   |   |           |   ├── CartController.java
|   |   |           |   ├── ItemController.java
|   |   |           |   ├── PaymentController.java
|   |   |           |   ├── ShipmentController.java
|   |   |           |
|   |   |           ├── model/
|   |   |           |   ├── Cart.java
```

```
│   │   │       │   ├── Customer.java
│   │   │       │   ├── Item.java
│   │   │       │   ├── Book.java
│   │   │       │   ├── Clothes.java
│   │   │       │   ├── Order.java
│   │   │       │   ├── Payment.java
│   │   │       │   ├── Shipment.java
│   │   │       │   ├── Comment.java
│   │   │       │   ├── Rating.java
│   │   │       │
│   │   │       ├── repository/
│   │   │       │   ├── CartRepository.java
│   │   │       │   ├── ItemRepository.java
│   │   │       │   ├── CustomerRepository.java
│   │   │       │   ├── PaymentRepository.java
│   │   │       │   ├── ShipmentRepository.java
│   │   │       │   ├── OrderRepository.java
│   │   │       │
│   │   │       ├── service/
│   │   │       │   ├── CartService.java
│   │   │       │   ├── ItemService.java
│   │   │       │   ├── PaymentService.java
│   │   │       │   ├── ShipmentService.java
│   │   │       │
│   │   │       └── EcommerceSystemApplication.java
│   │   │
```

```
│   │   ├── resources/
│   │   │   ├── templates/
│   │   │   │   ├── cart.html
│   │   │   │   ├── checkout.html
│   │   │   │   ├── itemSearchResults.html
│   │   │   │   ├── orderSuccess.html
│   │   │   │   ├── paymentError.html
│   │   │   │   ├── shipping.html
│   │   │   └── application.properties
│   │
│   └── test/
│       └── java/
│           └── com/
│               └── ecommerce/
│                   └── EcommerceSystemApplicationTests.java
│
├── pom.xml
└── README.md
```

Code

Controller

CartController

```java
package com.example.ecommerce.controller;

import com.example.ecommerce.model.Cart;
import com.example.ecommerce.model.Item;
import com.example.ecommerce.service.CartService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;

@RestController
@RequestMapping("/api/cart")
```

```
public class CartController {
    @Autowired
    private CartService cartService;

    @PostMapping("/{customerId}")
    public Cart createCart(@PathVariable int customerId) {
        return cartService.createCart(customerId);
    }

    @GetMapping("/{cartId}")
    public Cart getCart(@PathVariable int cartId) {
        return cartService.getCart(cartId);
    }

    @PostMapping("/{cartId}/addItem")
    public Cart addItemToCart(@PathVariable int cartId, @RequestBody Item
item) {
        return cartService.addItemToCart(cartId, item);
    }

    @DeleteMapping("/{cartId}/removeItem/{itemId}")
    public Cart removeItemFromCart(@PathVariable int cartId, @PathVariable int
itemId) {
        return cartService.removeItemFromCart(cartId, itemId);
    }
}
```

ItemController

```
package com.example.ecommerce.controller;

import com.example.ecommerce.model.Item;
import com.example.ecommerce.service.ItemService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@RestController
@RequestMapping("/api/items")
public class ItemController {
    @Autowired
    private ItemService itemService;

    @GetMapping
    public List<Item> getAllItems() {
        return itemService.getAllItems();
    }

    @GetMapping("/{itemId}")
    public Item getItemById(@PathVariable int itemId) {
        return itemService.getItemById(itemId);
    }
```

```
    @PostMapping
    public Item addItem(@RequestBody Item item) {
        return itemService.addItem(item);
    }

    @PutMapping("/{itemId}")
    public Item updateItem(@PathVariable int itemId, @RequestBody Item item) {
        return itemService.updateItem(itemId, item);
    }

    @DeleteMapping("/{itemId}")
    public void deleteItem(@PathVariable int itemId) {
        itemService.deleteItem(itemId);
    }
}
```

PaymentController

```
package com.example.ecommerce.controller;

import com.example.ecommerce.model.Payment;
import com.example.ecommerce.service.PaymentService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;

@RestController
@RequestMapping("/api/payment")
public class PaymentController {
    @Autowired
    private PaymentService paymentService;

    @PostMapping
    public Payment processPayment(@RequestBody Payment payment) {
        return paymentService.processPayment(payment);
    }

    @GetMapping("/{paymentId}")
    public Payment getPaymentById(@PathVariable int paymentId) {
        return paymentService.getPaymentById(paymentId);
    }
}
```

ShipmentController

```
package com.example.ecommerce.controller;

import com.example.ecommerce.model.Shipment;
import com.example.ecommerce.service.ShipmentService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;
```

```
@RestController
@RequestMapping("/api/shipment")
public class ShipmentController {
    @Autowired
    private ShipmentService shipmentService;

    @PostMapping
    public Shipment createShipment(@RequestBody Shipment shipment) {
        return shipmentService.createShipment(shipment);
    }

    @GetMapping("/{shipmentId}")
    public Shipment getShipmentById(@PathVariable int shipmentId) {
        return shipmentService.getShipmentById(shipmentId);
    }
}
```

Model

Cart

```
package com.example.ecommerce.model;


import jakarta.persistence.*;
import org.springframework.data.relational.core.mapping.Table;

import java.util.ArrayList;
import java.util.List;

@Entity
@Table(name = "cart")
public class Cart {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int idCart;

    @ManyToOne
    @JoinColumn(name = "idCustomer", nullable = false)
    private Customer customer;

    @OneToMany(mappedBy = "cart", cascade = CascadeType.ALL, orphanRemoval =
true)
    private List<Item> items = new ArrayList<>();

    // Getters and Setters

    public int getIdCart() {
        return idCart;
    }
```

```
    public void setIdCart(int idCart) {
        this.idCart = idCart;
    }

    public Customer getCustomer() {
        return customer;
    }

    public void setCustomer(Customer customer) {
        this.customer = customer;
    }

    public List<Item> getItems() {
        return items;
    }

    public void setItems(List<Item> items) {
        this.items = items;
    }

    public void addItem(Item item) {
        items.add(item);
        item.setCart(this);
    }

    public void removeItem(Item item) {
        items.remove(item);
        item.setCart(null);
    }


    public void setIdCustomer(int customerId) {
        this.customer.setId(customerId);
    }
}
```

Customer

```
package com.example.ecommerce.model;

import jakarta.persistence.*;

import java.util.ArrayList;
import java.util.List;

@Entity
@Table(name = "customer")
public class Customer {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;

    private String name;
```

```
    private String email;

    @Column(length = 255)
    private String address;

    @OneToMany(mappedBy = "customer", cascade = CascadeType.ALL, orphanRemoval
= true)
    private List<Cart> carts = new ArrayList<>();

    // Getters and Setters

    public int getId() {
        return id;
    }

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

    public String getName() {
        return name;
    }

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

    public String getEmail() {
        return email;
    }

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

    public String getAddress() {
        return address;
    }

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

    public List<Cart> getCarts() {
        return carts;
    }

    public void setCarts(List<Cart> carts) {
        this.carts = carts;
    }
}
```

Item

```java
package com.example.ecommerce.model;
import jakarta.persistence.*;

@Entity
@Inheritance(strategy = InheritanceType.JOINED)
@Table(name = "item")
public abstract class Item {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;

    private String name;
    private double price;

    @ManyToOne
    @JoinColumn(name = "idCart")
    private Cart cart;

    // Getters and Setters

    public int getId() {
        return id;
    }

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

    public String getName() {
        return name;
    }

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

    public double getPrice() {
        return price;
    }

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

    public Cart getCart() {
        return cart;
    }

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

Order

```
package com.example.ecommerce.model;

import jakarta.persistence.*;

@Entity
@Table(name = "order")
public class Order {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int idOrder;

    @ManyToOne
    @JoinColumn(name = "idCustomer", nullable = false)
    private Customer customer;

    @ManyToOne
    @JoinColumn(name = "idCart", nullable = false)
    private Cart cart;

    @OneToOne(mappedBy = "order", cascade = CascadeType.ALL, orphanRemoval =
true)
    private Payment payment;

    @OneToOne(mappedBy = "order", cascade = CascadeType.ALL, orphanRemoval =
true)
    private Shipment shipment;

    // Getters and Setters

    public int getIdOrder() {
        return idOrder;
    }

    public void setIdOrder(int idOrder) {
        this.idOrder = idOrder;
    }

    public Customer getCustomer() {
        return customer;
    }

    public void setCustomer(Customer customer) {
        this.customer = customer;
    }

    public Cart getCart() {
        return cart;
    }

    public void setCart(Cart cart) {
```

```
        this.cart = cart;
    }

    public Payment getPayment() {
        return payment;
    }

    public void setPayment(Payment payment) {
        this.payment = payment;
    }

    public Shipment getShipment() {
        return shipment;
    }

    public void setShipment(Shipment shipment) {
        this.shipment = shipment;
    }
}
```

Payment

```
package com.example.ecommerce.model;

import jakarta.persistence.*;

@Entity
@Table(name = "payment")
public class Payment {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int idPayment;

    private double amount;
    private String method;

    @OneToOne
    @JoinColumn(name = "idOrder", nullable = false)
    private Order order;

    // Getters and Setters

    public int getIdPayment() {
        return idPayment;
    }

    public void setIdPayment(int idPayment) {
        this.idPayment = idPayment;
    }

    public double getAmount() {
        return amount;
    }
```

```
    public void setAmount(double amount) {
        this.amount = amount;
    }

    public String getMethod() {
        return method;
    }

    public void setMethod(String method) {
        this.method = method;
    }

    public Order getOrder() {
        return order;
    }

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

Rating

```
package com.example.ecommerce.model;

import jakarta.persistence.*;

@Entity
@Table(name = "rating")
public class Rating {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int idRating;

    private int stars;

    @ManyToOne
    @JoinColumn(name = "idCustomer", nullable = false)
    private Customer customer;

    @ManyToOne
    @JoinColumn(name = "idItem", nullable = false)
    private Item item;

    // Getters and Setters

    public int getIdRating() {
        return idRating;
    }

    public void setIdRating(int idRating) {
        this.idRating = idRating;
```

```
    }

    public int getStars() {
        return stars;
    }

    public void setStars(int stars) {
        this.stars = stars;
    }

    public Customer getCustomer() {
        return customer;
    }

    public void setCustomer(Customer customer) {
        this.customer = customer;
    }

    public Item getItem() {
        return item;
    }

    public void setItem(Item item) {
        this.item = item;
    }
}
```

Shipment

```
package com.example.ecommerce.model;

import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import jakarta.persistence.JoinColumn;
import jakarta.persistence.OneToOne;
import jakarta.persistence.Table;

@Entity
@Table(name = "shipment")
public class Shipment {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int idShipment;

    private String address;
    private String status;

    @OneToOne
    @JoinColumn(name = "idOrder", nullable = false)
    private Order order;
```

```
    // Getters and Setters

    public int getIdShipment() {
        return idShipment;
    }

    public void setIdShipment(int idShipment) {
        this.idShipment = idShipment;
    }

    public String getAddress() {
        return address;
    }

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

    public String getStatus() {
        return status;
    }

    public void setStatus(String status) {
        this.status = status;
    }

    public Order getOrder() {
        return order;
    }

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

Comment

```
package com.example.ecommerce.model;

import jakarta.persistence.*;

@Entity
@Table(name = "comment")
public class Comment {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int idComment;

    @Column(length = 500)
    private String text;

    @ManyToOne
```

```
    @JoinColumn(name = "idCustomer", nullable = false)
    private Customer customer;

    @ManyToOne
    @JoinColumn(name = "idItem", nullable = false)
    private Item item;

    // Getters and Setters

    public int getIdComment() {
        return idComment;
    }

    public void setIdComment(int idComment) {
        this.idComment = idComment;
    }

    public String getText() {
        return text;
    }

    public void setText(String text) {
        this.text = text;
    }

    public Customer getCustomer() {
        return customer;
    }

    public void setCustomer(Customer customer) {
        this.customer = customer;
    }

    public Item getItem() {
        return item;
    }

    public void setItem(Item item) {
        this.item = item;
    }
}
```

Clothes

```
package com.example.ecommerce.model;

import jakarta.persistence.Entity;
import jakarta.persistence.Table;
@Entity
@Table(name = "clothes")
public class Clothes extends Item {
    private String size;
    private String color;
```

```
    // Getters and Setters

    public String getSize() {
        return size;
    }

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

    public String getColor() {
        return color;
    }

    public void setColor(String color) {
        this.color = color;
    }
}
```

Book

```
package com.example.ecommerce.model;

import jakarta.persistence.Entity;
import jakarta.persistence.Table;

@Entity
@Table(name = "book")
public class Book extends Item {
    private String author;
    private String ISBN;
    private String genre;

    // Getters and Setters

    public String getAuthor() {
        return author;
    }

    public void setAuthor(String author) {
        this.author = author;
    }

    public String getISBN() {
        return ISBN;
    }

    public void setISBN(String ISBN) {
        this.ISBN = ISBN;
    }

    public String getGenre() {
```

```
        return genre;
    }

    public void setGenre(String genre) {
        this.genre = genre;
    }
}
```

Repository (DAO)

CartRepository

```
package com.example.ecommerce.repository;

import com.example.ecommerce.model.Cart;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

@Repository
public interface CartRepository extends JpaRepository<Cart, Integer> {
    Cart findByIdCustomer(int customerId);
}
```

CustomerRepository

```
package com.example.ecommerce.repository;

import com.example.ecommerce.model.Customer;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

@Repository
public interface CustomerRepository extends JpaRepository<Customer, Integer>
{
    // Tìm kiếm customer theo email
    Customer findByEmail(String email);
}
```

ItemRepository

```
package com.example.ecommerce.repository;

import com.example.ecommerce.model.Item;

import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

@Repository
public interface ItemRepository extends JpaRepository<Item, Integer> {
```

```
    // Tìm kiếm item theo tên
    Item findByName(String name);
}
```

OrderRepository

```
package com.example.ecommerce.repository;

import com.example.ecommerce.model.Order;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

import java.util.List;

@Repository
public interface OrderRepository extends JpaRepository<Order, Integer> {
    List<Order> findByIdCustomer(int customerId);
}
```

PaymentRepository

```
package com.example.ecommerce.repository;



import com.example.ecommerce.model.Payment;

import org.springframework.data.jpa.repository.JpaRepository;

import org.springframework.stereotype.Repository;



@Repository

public interface PaymentRepository extends JpaRepository<Payment, Integer> {

    // Tìm kiếm payment theo order

    Payment findByIdOrder(int orderId);

}
```

ShipmentRepository

```
package com.example.ecommerce.repository;

import com.example.ecommerce.model.Shipment;
```

```java
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

@Repository
public interface ShipmentRepository extends JpaRepository<Shipment, Integer>
{
   // Tìm kiếm shipment theo order
   Shipment findByIdOrder(int orderId);
}
```

Service

CartService

```java
package com.example.ecommerce.service;

import com.example.ecommerce.model.Cart;
import com.example.ecommerce.model.Item;
import com.example.ecommerce.repository.CartRepository;
import com.example.ecommerce.repository.ItemRepository;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

@Service
public class CartService {
   @Autowired
   private CartRepository cartRepository;

   @Autowired
   private ItemRepository itemRepository;

   public Cart createCart(int customerId) {
       Cart cart = new Cart();
       cart.setIdCustomer(customerId);
       return cartRepository.save(cart);
   }

   public Cart getCart(int cartId) {
       return cartRepository.findById(cartId).orElse(null);
   }

   public Cart addItemToCart(int cartId, Item item) {
       Cart cart = getCart(cartId);
       if (cart != null) {
           cart.addItem(item);
           cartRepository.save(cart);
       }
       return cart;
   }
```

```
    public Cart removeItemFromCart(int cartId, int itemId) {
        Cart cart = getCart(cartId);
        if (cart != null) {
            Item item = itemRepository.findById(itemId).orElse(null);
            if (item != null) {
                cart.removeItem(item);
                cartRepository.save(cart);
            }
        }
        return cart;

    }
}
```

ItemService

```
package com.example.ecommerce.service;

import com.example.ecommerce.model.Item;
import com.example.ecommerce.repository.ItemRepository;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.List;

@Service
public class ItemService {
    @Autowired
    private ItemRepository itemRepository;

    public List<Item> getAllItems() {
        return itemRepository.findAll();
    }

    public Item getItemById(int itemId) {
        return itemRepository.findById(itemId).orElse(null);
    }

    public Item addItem(Item item) {
        return itemRepository.save(item);
    }

    public Item updateItem(int itemId, Item item) {
        if (itemRepository.existsById(itemId)) {
            item.setId(itemId);
            return itemRepository.save(item);
        }
        return null;
    }

    public void deleteItem(int itemId) {
        itemRepository.deleteById(itemId);
```

```
    }
}
```

PaymentService

```
package com.example.ecommerce.service;

import com.example.ecommerce.model.Payment;
import com.example.ecommerce.repository.PaymentRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

@Service
public class PaymentService {
    @Autowired
    private PaymentRepository paymentRepository;

    public Payment processPayment(Payment payment) {
        // Logic for processing payment
        // For example, validating payment method and amount
        return paymentRepository.save(payment);
    }

    public Payment getPaymentById(int paymentId) {
        return paymentRepository.findById(paymentId).orElse(null);
    }
}
```

ShipmentService

```
package com.example.ecommerce.service;

import com.example.ecommerce.model.Shipment;
import com.example.ecommerce.repository.ShipmentRepository;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

@Service
public class ShipmentService {
    @Autowired
    private ShipmentRepository shipmentRepository;

    public Shipment createShipment(Shipment shipment) {
        // Logic for creating shipment (e.g., determining shipment status)
        return shipmentRepository.save(shipment);
    }

    public Shipment getShipmentById(int shipmentId) {
        return shipmentRepository.findById(shipmentId).orElse(null);
```

```
    }
}
```

Template (View)

cart.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Your Cart</title>
</head>
<body>
<h1>Your Cart</h1>
<table>
    <thead>
    <tr>
        <th>Item Name</th>
        <th>Price</th>
        <th>Action</th>
    </tr>
    </thead>
    <tbody>
    <!-- Assuming you have a list of items in the model -->
    <tr th:each="item : ${cart.items}">
        <td th:text="${item.name}"></td>
        <td th:text="${item.price}"></td>
        <td><a th:href="@{/cart/remove(item.id)}">Remove</a></td>
    </tr>
    </tbody>
</table>
<h3>Total: <span th:text="${total}"></span></h3>
<form th:action="@{/checkout}" method="post">
    <button type="submit">Proceed to Checkout</button>
</form>
</body>
</html>
```

checkout.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Checkout</title>
</head>
<body>
```

```
<h1>Checkout</h1>
<form action="/payment" method="post">
   <label for="address">Shipping Address:</label>
   <input type="text" id="address" name="address" required>

   <h3>Select Payment Method:</h3>
   <input type="radio" id="paypal" name="paymentMethod" value="PayPal"
required>
   <label for="paypal">PayPal</label><br>

   <input type="radio" id="creditCard" name="paymentMethod" value="Credit
Card">
   <label for="creditCard">Credit Card</label><br>

   <button type="submit">Complete Purchase</button>
</form>
</body>
</html>
```

itemsearchresult.html

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Item Search Results</title>
</head>
<body>
<h1>Search Results</h1>
<ul>
   <li th:each="item : ${items}">
       <h2 th:text="${item.name}"></h2>
       <p>Price: <span th:text="${item.price}"></span></p>
       <a th:href="@{/cart/add(item.id)}">Add to Cart</a>
   </li>
</ul>
</body>
</html>
```

ordersuccess.html

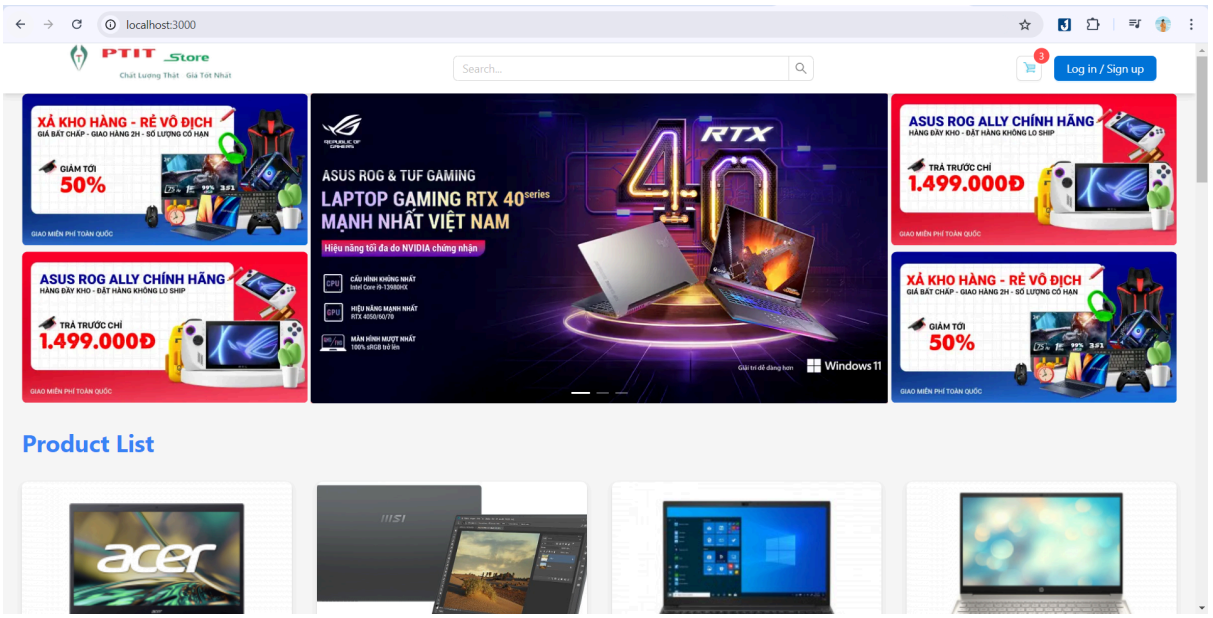
```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Order Success</title>
</head>
<body>
<h1>Order Successful!</h1>
```

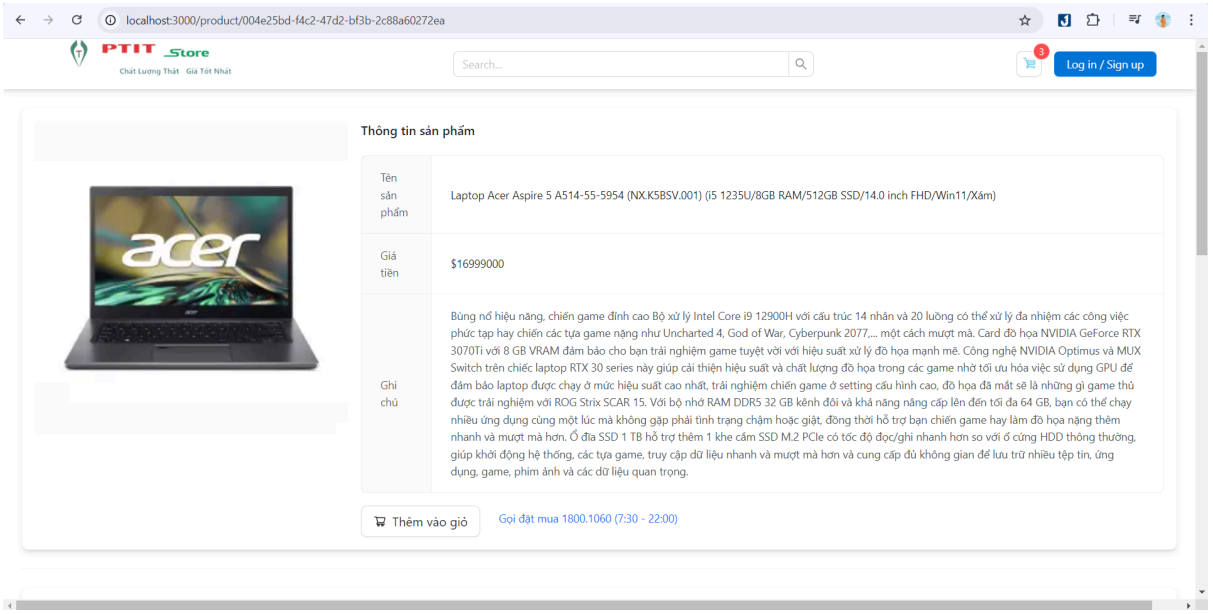
```
<p>Your order has been placed successfully.</p>
<a href="/">Go to Homepage</a>
</body>
</html>
```

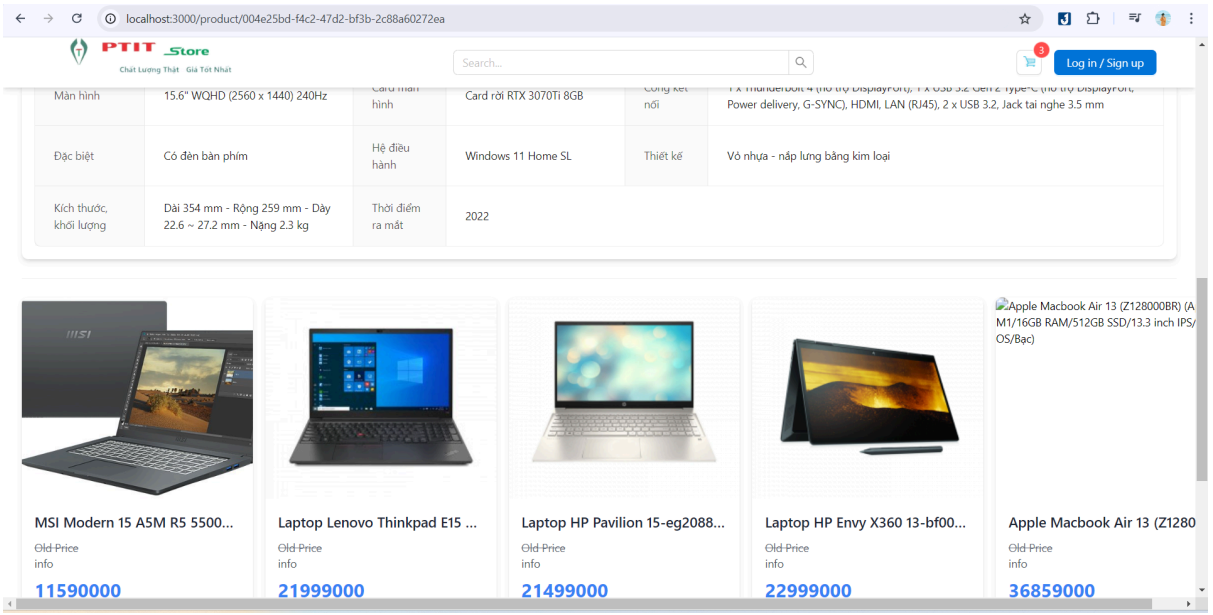
shipping.html

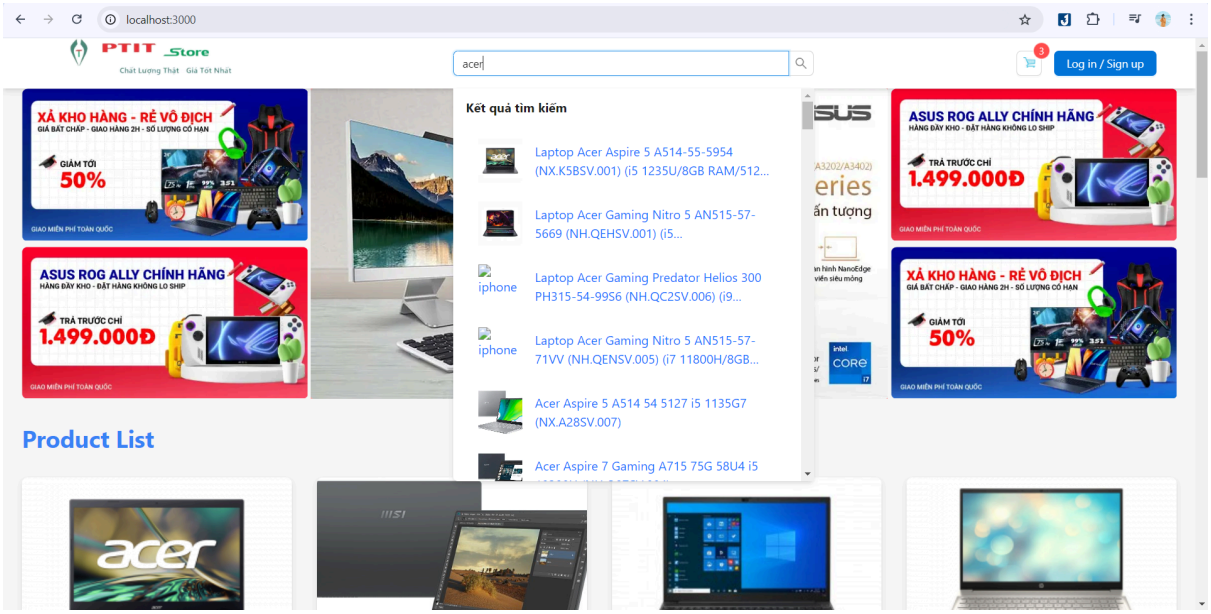
```
<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <title>Shipping Details</title>
</head>
<body>
<h1>Shipping Details</h1>
<p>Your order is being processed and will be shipped to:</p>
<p th:text="${shippingAddress}"></p>
<p>Status: <span th:text="${shippingStatus}"></span></p>
<a href="/">Go to Homepage</a>
</body>
</html>
```

UI




localhost:3000/product/004e25bd-f4c2-47d2-bf3b-2c88a60272ea
PTIT Store
Chất Lượng Thật Giá Tốt Nhất
Search...
Log in / Sign up
Thông tin sản phẩm
Tên sản phẩm
Laptop Acer Aspire 5 A514-55-5954 (NX.K5BSV.001) (i5 1235U/8GB RAM/512GB SSD/14.0 inch FHD/Win11/Xám)
Giá tiền
$16999000
Ghi chú
Bùng nổ hiệu năng, chiến game đỉnh cao Bộ xử lý Intel Core i9 12900H với cấu trúc 14 nhân và 20 luồng có thể xử lý đa nhiệm các công việc phức tạp hay chiến các tựa game nặng như Uncharted 4, God of War, Cyberpunk 2077,... một cách mượt mà. Card đồ họa NVIDIA GeForce RTX 3070Ti với 8 GB VRAM đảm bảo cho bạn trải nghiệm game tuyệt vời với hiệu suất xử lý đồ họa mạnh mẽ. Công nghệ NVIDIA Optimus và MUX Switch trên chiếc laptop RTX 30 series này giúp cải thiện hiệu suất và chất lượng đồ họa trong các game nhờ tối ưu hóa việc sử dụng GPU để đảm bảo laptop được chạy ở mức hiệu suất cao nhất, trải nghiệm chiến game ở setting cấu hình cao, đồ họa đã mắt sẽ là những gì game thủ được trải nghiệm với ROG Strix SCAR 15. Với bộ nhớ RAM DDR5 32 GB kênh đôi và khả năng nâng cấp lên đến tối đa 64 GB, bạn có thể chạy nhiều ứng dụng cùng một lúc mà không gặp phải tình trạng chậm hoặc giật, đồng thời hỗ trợ bạn chiến game hay làm đồ họa nặng thêm nhanh và mượt mà hơn. Ổ đĩa SSD 1 TB hỗ trợ thêm 1 khe cắm SSD M.2 PCIe có tốc độ đọc/ghi nhanh hơn so với ổ cứng HDD thông thường, giúp khởi động hệ thống, các tựa game, truy cập dữ liệu nhanh và mượt mà hơn và cung cấp đủ không gian để lưu trữ nhiều tệp tin, ứng dụng, game, phim ảnh và các dữ liệu quan trọng.
Thêm vào giỏ
Gọi đặt mua 1800.1060 (7:30 - 22:00)


localhost:3000/product/004e25bd-f4c2-47d2-bf3b-2c88a60272ea
PTIT Store
Chất Lượng Thật Giá Tốt Nhất
Search...
Log in / Sign up
Màn hình
15.6" WQHD (2560 x 1440) 240Hz
Card rời RTX 3070Ti 8GB
Power delivery, G-SYNC), HDMI, LAN (RJ45), 2 x USB 3.2, Jack tai nghe 3.5 mm
Đặc biệt
Có đèn bàn phím
Hệ điều hành
Windows 11 Home SL
Thiết kế
Vỏ nhựa - nắp lưng bằng kim loại
Kích thước, khối lượng
Dài 354 mm - Rộng 259 mm - Dày 22.6 ~ 27.2 mm - Nặng 2.3 kg
Thời điểm ra mắt
2022
MSI Modern 15 A5M R5 5500...
Old Price
info
11590000
Laptop Lenovo Thinkpad E15 ...
Old Price
info
21999000
Laptop HP Pavilion 15-eg2088...
Old Price
info
21499000
Laptop HP Envy X360 13-bf00...
Old Price
info
22999000
Apple Macbook Air 13 (Z128000BR) (A M1/16GB RAM/512GB SSD/13.3 inch IPS/ OS/Bạc)
Apple Macbook Air 13 (Z1280
Old Price
info
36859000


localhost:3000
PTIT Store
Chất Lượng Thật Giá Tốt Nhất
acer
Log in / Sign up
Kết quả tìm kiếm
Laptop Acer Aspire 5 A514-55-5954 (NX.K5BSV.001) (i5 1235U/8GB RAM/512...
Laptop Acer Gaming Nitro 5 AN515-57-5669 (NH.QEHSV.001) (i5...
Laptop Acer Gaming Predator Helios 300 PH315-54-99S6 (NH.QC2SV.006) (i9...
Laptop Acer Gaming Nitro 5 AN515-57-71VV (NH.QENSV.005) (i7 11800H/8GB...
Acer Aspire 5 A514 54 5127 i5 1135G7 (NX.A28SV.007)
Acer Aspire 7 Gaming A715 75G 58U4 i5
XẢ KHO HÀNG - RẺ VÔ ĐỊCH
GIÁ BẤT CHẤP - GIAO HÀNG 2H - SỐ LƯỢNG CÓ HẠN
GIẢM TỚI 50%
ASUS ROG ALLY CHÍNH HÃNG
HÀNG ĐẦY KHO - ĐẶT HÀNG KHÔNG LO SHIP
TRẢ TRƯỚC CHỈ 1.499.000Đ
GIAO MIỄN PHÍ TOÀN QUỐC
Product List