Project structure

A screenshot of a computer

AI-generated content may be incorrect.

**AdminController.java**

package com.flipkart.shoppingkart.controller;

import java.util.List;

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

import org.springframework.web.bind.annotation.DeleteMapping;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.PostMapping;

import org.springframework.web.bind.annotation.RequestBody;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RequestParam;

import org.springframework.web.bind.annotation.RestController;

import com.flipkart.shoppingkart.entity.Cart;

import com.flipkart.shoppingkart.entity.Product;

import com.flipkart.shoppingkart.entity.User;

import com.flipkart.shoppingkart.repository.CartRepository;

import com.flipkart.shoppingkart.repository.ProductRepository;

import com.flipkart.shoppingkart.repository.UserRepository;

import com.flipkart.shoppingkart.service.AdminService;

@RestController

@RequestMapping("/admin")

public class AdminController {

@Autowired

private AdminService adminService;

@GetMapping("/allcarts")

public List<Cart> getAllCart()

{

return adminService.getAllCarts();

}

@GetMapping("/cart/id")

public Cart getCartById(@RequestParam Long cartId)

{

return adminService.getCartById(cartId);

}

@GetMapping("/user/all")

public List<User> getAllUsers()

{

return adminService.getAllUsers();

}

@GetMapping("/user/id")

public User getUserById(@RequestParam Long userId)

{

return adminService.getUserById(userId);

}

@GetMapping("/product/all")

public List<Product> getAllProducts()

{

return adminService.getAllProducts();

}

@GetMapping("/product/id")

public Product getProductById(@RequestParam Long productId)

{

return adminService.getProductById(productId);

}

@GetMapping("/product/name")

public Product getProductByName(String productName)

{

return adminService.getProductByName(productName);

}

@PostMapping("product/add")

public Product addProduct(@RequestBody Product product)

{

return adminService.addProduct(product);

}

@DeleteMapping("product/delete")

public String deleteProduct(@RequestParam Long productId)

{

return adminService.deleteProduct(productId);

}

}

**CartController.java**

**package** com.flipkart.shoppingkart.controller;

**import** java.util.ArrayList;

**import** java.util.List;

**import** java.util.Optional;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.web.bind.annotation.GetMapping;

**import** org.springframework.web.bind.annotation.PostMapping;

**import** org.springframework.web.bind.annotation.RequestMapping;

**import** org.springframework.web.bind.annotation.RequestParam;

**import** org.springframework.web.bind.annotation.RestController;

**import** com.flipkart.shoppingkart.entity.Cart;

**import** com.flipkart.shoppingkart.entity.Product;

**import** com.flipkart.shoppingkart.entity.User;

**import** com.flipkart.shoppingkart.service.CartService;

@RestController

@RequestMapping("/cart")

**public** **class** CartController {

@Autowired

**private** CartService cartService;

@PostMapping("/add")

**public** String addToCart(@RequestParam Long userid,

@RequestParam Long productid,

@RequestParam **int** quantity,

@RequestParam(defaultValue = "sms") String notifyby )

{

**return** cartService.addToCart(userid, productid, quantity, notifyby);

//return "product added to card and notification sent by"+notifyby;

}

@GetMapping("/all")

**public** List<Cart> getAll()

{

**return** cartService.getAll();

}

@GetMapping("/userid")

**public** List<Cart> getByUserid(@RequestParam Long userid)

{

**return** cartService.getByUserid(userid);

}

@GetMapping("/userid/requireddata")

**public** List<String> getByUserid2(@RequestParam Long userid)

{

List<Cart> userCart = cartService.getByUserid(userid);

List<String>data = **new** ArrayList<>();

**int** i=1;

**for**(Cart cart:userCart)

{

String msg = i++ +" Product Name: "+cart.getProduct().getName()+

" Cost: "+cart.getProduct().getPrice()+

" Quantity: "+cart.getQuantity();

data.add(msg);

}

**return** data;

}

@GetMapping("/productid")

**public** List<String> getByProductid(@RequestParam Long productid)

{

Optional<Cart>productCart = cartService.getByProductid(productid);

List<String>data = **new** ArrayList<>();

**int** i=1;

// for(Cart cart:productCart)

// {

// String msg = i++ +" Product Name: "+cart.getProduct().getName()+

// " Cost: "+cart.getProduct().getPrice()+

// " Quantity: "+cart.getQuantity()+

// " User id "+cart.getUser().getId()+

// " User Name: "+cart.getUser().getName();

//

// data.add(msg);

// }

**return** data;

}

}

**ProductController.java**

package com.flipkart.shoppingkart.controller;

import java.util.List;

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

import org.springframework.web.bind.annotation.DeleteMapping;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.PostMapping;

import org.springframework.web.bind.annotation.PutMapping;

import org.springframework.web.bind.annotation.RequestBody;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RequestParam;

import org.springframework.web.bind.annotation.RestController;

import com.flipkart.shoppingkart.entity.Product;

import com.flipkart.shoppingkart.service.ProductService;

@RestController

@RequestMapping("/product")

public class ProductController {

@Autowired

private ProductService productService;

@GetMapping("/all")

public List<Product> getProducts()

{

return productService.getProducts();

}

@GetMapping("/id")

public Product getProduct(@RequestParam Long productid)

{

return productService.getProduct(productid);

}

@GetMapping("/name")

public Product getProduct(@RequestParam String name)

{

return productService.getProductByName(name);

}

@PostMapping("/add")

public Product addProduct(@RequestBody Product product)

{

return productService.addProduct(product);

}

@PutMapping("/update")

public Product updateProduct(@RequestBody Product product)

{

return productService.addProduct(product);

}

@DeleteMapping("/delete")

public String deleteProduct(@RequestParam Long productId)

{

return productService.deleteProduct(productId);

}

}

**UserController.java**

package com.flipkart.shoppingkart.controller;

import java.util.List;

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

import org.springframework.web.bind.annotation.DeleteMapping;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.PostMapping;

import org.springframework.web.bind.annotation.PutMapping;

import org.springframework.web.bind.annotation.RequestBody;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RequestParam;

import org.springframework.web.bind.annotation.RestController;

import com.flipkart.shoppingkart.entity.Cart;

import com.flipkart.shoppingkart.entity.Product;

import com.flipkart.shoppingkart.entity.User;

import com.flipkart.shoppingkart.service.UserService;

@RestController

@RequestMapping("/user")

public class UserController {

@Autowired

private UserService userService;

//user registration

@PostMapping("/register")

public User registerUser(@RequestBody User user)

{

return userService.registerUser(user);

}

@GetMapping("/id")

public User getUserById(@RequestParam Long userid)

{

return userService.getUserById(userid);

}

@PutMapping("/update")

public String updateUserName(@RequestParam Long userid, String username)

{

return userService.updateUserName(userid,username);

}

@GetMapping("/productlist")

public List<Product> getProductlist()

{

return userService.getProductlist();

}

@GetMapping("/product/id")

public Product getProductById(@RequestParam Long productid)

{

return userService.getProductById(productid);

}

@GetMapping("/product/name")

public Product getProductByName(@RequestParam String productname)

{

return userService.getProductByName(productname);

}

@GetMapping("/login")

public String userLogin(@RequestParam String username, String password)

{

return userService.userLogin(username, password);

}

@PostMapping("/cart/add")

public String addToCart(String productName, int quantity)

{

return userService.addToCart(productName, quantity);

}

@GetMapping("/cart/get")

public List<String> getCart()

{

return userService.getCart();

}

@GetMapping("/cart/amount")

public Double getCartAmount()

{

return userService.getCartAmount();

}

@DeleteMapping("/checkout")

public String checkout(@RequestParam Long carid)

{

return "";

}

}

**Cart.java**

package com.flipkart.shoppingkart.entity;

import java.time.LocalDateTime;

import jakarta.persistence.Entity;

import jakarta.persistence.GeneratedValue;

import jakarta.persistence.GenerationType;

import jakarta.persistence.Id;

import jakarta.persistence.JoinColumn;

import jakarta.persistence.ManyToOne;

import jakarta.persistence.OneToMany;

import jakarta.persistence.OneToOne;

@Entity

public class Cart {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

@ManyToOne

@JoinColumn(name = "user\_id")

private User user;

@ManyToOne

@JoinColumn(name = "product\_id")

private Product product;

private int quantity;

private LocalDateTime date;

public Long getId() {

return id;

}

public void setId(Long id) {

this.id = id;

}

public User getUser() {

return user;

}

public void setUser(User user) {

this.user = user;

}

public Product getProduct() {

return product;

}

public void setProduct(Product product) {

this.product = product;

}

public int getQuantity() {

return quantity;

}

public void setQuantity(int quantity) {

this.quantity = quantity;

}

public LocalDateTime getDate() {

return date;

}

public void setDate(LocalDateTime date) {

this.date = date;

}

@Override

public String toString() {

return "Cart [id=" + id + ", product=" + product.getName() + ", quantity=" + quantity + ", date=" + date + "]";

}

}

**Product.java**

package com.flipkart.shoppingkart.entity;

import jakarta.persistence.Column;

import jakarta.persistence.Entity;

import jakarta.persistence.GeneratedValue;

import jakarta.persistence.GenerationType;

import jakarta.persistence.Id;

import lombok.Data;

@Entity

@Data

public class Product {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String name;

private Double price;

@Column(columnDefinition = "INT DEFAULT 0")

private int quantity;

public int getQuantity() {

return quantity;

}

public void setQuantity(int quantity) {

this.quantity = quantity;

}

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 Double getPrice() {

return price;

}

public void setPrice(Double price) {

this.price = price;

}

}

**User.java**

package com.flipkart.shoppingkart.entity;

import jakarta.persistence.CascadeType;

import jakarta.persistence.Entity;

import jakarta.persistence.GeneratedValue;

import jakarta.persistence.GenerationType;

import jakarta.persistence.Id;

import jakarta.persistence.OneToOne;

import lombok.Data;

import lombok.Getter;

@Data // this provide setter getter and reqd constructors

@Entity // Map this class to the database table

public class User {

@Id // make id as primary key

@GeneratedValue(strategy = GenerationType.IDENTITY) // this will generate values for id automatically

private Long id;

private String name;

private String username;

private String password;

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 getUsername() {

return username;

}

public void setUsername(String username) {

this.username = username;

}

public String getPassword() {

return password;

}

public void setPassword(String password) {

this.password = password;

}

// this represent oneto one relationship with cart

//if we save/delete user then its cart will also be saved/deleted

// @OneToOne(mappedBy = "user", cascade = CascadeType.ALL)

// private Cart cart;

}

**ResourceNotFoundException.ajva**

**package** com.flipkart.shoppingkart.exception;

**public** **class** ResourceNotFoundException **extends** RuntimeException{

**public** ResourceNotFoundException(String message)

{

**super**(message);

}

}

**CartRepository.java**

package com.flipkart.shoppingkart.repository;

import java.util.List;

import java.util.Optional;

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

import org.springframework.stereotype.Repository;

import com.flipkart.shoppingkart.entity.Cart;

@Repository

public interface CartRepository extends JpaRepository<Cart, Long>{

Optional<Cart> findByProductId(Long productid);

Optional<Cart> findByUserId(Long userid);

List<Cart> findAllByUserId(Long userid);

}

**ProductRepository.java**

package com.flipkart.shoppingkart.repository;

import java.util.Optional;

import java.util.Optional;

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

import org.springframework.stereotype.Repository;

import com.flipkart.shoppingkart.entity.Product;

@Repository

public interface ProductRepository extends JpaRepository<Product, Long>{

Optional<Product> findByNameIgnoreCase(String name);

}

**UserRepository.java**

package com.flipkart.shoppingkart.repository;

import java.util.Optional;

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

import org.springframework.stereotype.Repository;

import com.flipkart.shoppingkart.entity.User;

@Repository // This will map the the interface as Spring managed DAO bean and handle exceptions

public interface UserRepository extends JpaRepository<User,Long>{

// JpaRepository will provide methods for

//save, findById, count, deleteById, findAll(), etc

Optional<User> findByUsernameIgnoreCase(String username);

}

**AdminService.java**

package com.flipkart.shoppingkart.service;

import java.util.List;

import java.util.Optional;

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

import org.springframework.stereotype.Service;

import com.flipkart.shoppingkart.entity.Cart;

import com.flipkart.shoppingkart.entity.Product;

import com.flipkart.shoppingkart.entity.User;

import com.flipkart.shoppingkart.exception.ResourceNotFoundException;

import com.flipkart.shoppingkart.repository.CartRepository;

import com.flipkart.shoppingkart.repository.ProductRepository;

import com.flipkart.shoppingkart.repository.UserRepository;

@Service

public class AdminService {

@Autowired

private UserRepository userRepository;

@Autowired

private ProductRepository productRepository;

@Autowired

private CartRepository cartRepository;

public List<Cart> getAllCarts()

{

return cartRepository.findAll();

}

public Cart getCartById(Long cartId)

{

return cartRepository.findById(cartId).orElseThrow(

()-> new ResourceNotFoundException("cart not found with id: "+cartId)

);

}

public List<User> getAllUsers()

{

return userRepository.findAll();

}

public User getUserById(Long userId)

{

return userRepository.findById(userId).orElseThrow(

()->new ResourceNotFoundException("User not found with id: "+userId)

);

}

public List<Product> getAllProducts()

{

return productRepository.findAll();

}

public Product getProductById(Long productId)

{

return productRepository.findById(productId).orElseThrow(

()-> new ResourceNotFoundException("Product is not available with Id: "+productId)

);

}

public Product getProductByName(String productName)

{

return productRepository.findByNameIgnoreCase(productName).orElseThrow(

()-> new ResourceNotFoundException("product is not available with name: "+productName)

);

}

public Product addProduct(Product product)

{

Optional<Product> availableProduct = productRepository.findByNameIgnoreCase(product.getName());

if(availableProduct.isEmpty())

return productRepository.save(product);

product.setQuantity(product.getQuantity()+availableProduct.get().getQuantity());

product.setId(availableProduct.get().getId());

return productRepository.save(product);

}

public String deleteProduct(Long productId)

{

Optional<Product> product = productRepository.findById(productId);

if(product.isEmpty())

return "product is not available";

productRepository.deleteById(productId);

return "Product deleted from DB with id: "+productId;

}

}

**CartService.java**

package com.flipkart.shoppingkart.service;

import java.time.LocalDateTime;

import java.util.ArrayList;

import java.util.Collections;

import java.util.Comparator;

import java.util.List;

import java.util.Optional;

import java.util.stream.Collectors;

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

import org.springframework.beans.factory.annotation.Qualifier;

import org.springframework.stereotype.Service;

import org.springframework.web.bind.annotation.RequestParam;

import com.flipkart.shoppingkart.entity.Cart;

import com.flipkart.shoppingkart.entity.Product;

import com.flipkart.shoppingkart.entity.User;

import com.flipkart.shoppingkart.repository.CartRepository;

import com.flipkart.shoppingkart.repository.ProductRepository;

import com.flipkart.shoppingkart.repository.UserRepository;

@Service

public class CartService {

@Autowired

private ProductRepository productRepository;

@Autowired

private UserRepository userRepository;

@Autowired

private CartRepository cartRepository;

@Autowired

@Qualifier("smsNotification")

private NotificationService smsNotification;

@Autowired

@Qualifier("emailNotification")

private NotificationService emailNotification;

public String addToCart(Long userId, Long productId, int quantity, String notifyBy)

{

Optional<User> user = userRepository.findById(userId);

Optional<Product> product = productRepository.findById(productId);

//checking if user is present in database or not

if(user.isEmpty())

{

return "user not found";

}

//checking if product is present or not

if(product.isEmpty())

{

return "proiduct not available";

}

if(quantity>product.get().getQuantity())

{

return "Only "+product.get().getQuantity()+" quantities are available";

}

product.get().setQuantity(product.get().getQuantity() - quantity);

// if userid and product id already present in cart then only update quantity

Optional<Cart> productCartList = cartRepository.findByProductId(productId);

// for(Cart cart: productCartList)

// {

// if(cart.getUser().getId().equals(userId))

// {

// cart.setQuantity(cart.getQuantity()+quantity);

// cartRepository.save(cart);

//

// NotificationService notifier = notifyBy.equalsIgnoreCase("sms")?smsNotification:emailNotification;

// String msg = "product: "+product.get().getName()+" added to cart";

// notifier.sendNotification(msg, user.get().getName());

//

// return "product added to card and notification sent by"+notifyBy;

// }

// }

// else add new entry in cart

Cart cart = new Cart();

cart.setUser(user.get());

cart.setProduct(product.get());

cart.setDate(LocalDateTime.now());

cart.setQuantity(quantity);

cartRepository.save(cart);

NotificationService notifier = notifyBy.equalsIgnoreCase("sms")?smsNotification:emailNotification;

String msg = "product: "+product.get().getName()+" added to cart";

notifier.sendNotification(msg, user.get().getName());

return "product added to card and notification sent by"+notifyBy;

}

public List<Cart> getAll()

{

return cartRepository.findAll();

}

public List<Cart> getByUserid(Long userid)

{

// List<Cart>userCart = new ArrayList<>();

// for(Cart cart:cartRepository.findAll())

// {

// if(cart.getUser().getId()==userid)

// userCart.add(cart);

// }

// return userCart;

//OR

//using stream

return cartRepository.findAll().stream().filter(

cart->userid.equals(cart.getUser().getId()))

.collect(Collectors.toList());

// OR modify cartRepository check getbyproductid

}

public Optional<Cart> getByProductid(Long productid)

{

return cartRepository.findByProductId(productid);

}

}

**EmailNotification.java**

**package** com.flipkart.shoppingkart.service;

**import** org.springframework.stereotype.Service;

@Service("emailNotification")

**public** **class** EmailNotification **implements** NotificationService {

**public** **void** sendNotification(String msg, String to)

{

System.***out***.println("Email sent with msg: "+msg+" to the user: "+to);

}

}

**NotificationService.java**

**package** com.flipkart.shoppingkart.service;

**public** **interface** NotificationService {

**void** sendNotification(String msg, String to);

}

**SmsNotification.java**

**package** com.flipkart.shoppingkart.service;

**import** org.springframework.stereotype.Service;

@Service("smsNotification")

**public** **class** SmsNotification **implements** NotificationService{

**public** **void** sendNotification(String msg,String to)

{

System.***out***.println("SMS sent with msg: "+msg+" to the user: "+to);

}

}

**ProductService.java**

package com.flipkart.shoppingkart.service;

import java.util.List;

import java.util.Optional;

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

import org.springframework.stereotype.Service;

import org.springframework.web.bind.annotation.RequestBody;

import org.springframework.web.bind.annotation.RequestParam;

import com.flipkart.shoppingkart.entity.Product;

import com.flipkart.shoppingkart.repository.ProductRepository;

@Service

public class ProductService {

@Autowired

private ProductRepository productRepository;

private Product product;

public List<Product> getProducts()

{

return productRepository.findAll();

}

public Product getProduct( Long productId)

{

return productRepository.findById(productId).orElseThrow();

}

public Product getProductByName(String productName)

{

return productRepository.findByNameIgnoreCase(productName).orElseThrow();

}

public Product addProduct(Product product)

{

Optional<Product> optProduct = productRepository.findByNameIgnoreCase(product.getName());

if(optProduct.isEmpty())

return productRepository.save(product);

optProduct.get().setQuantity(product.getQuantity()+optProduct.get().getQuantity());

return optProduct.get();

}

public String deleteProduct(Long productId)

{

if(productRepository.existsById(productId))

{

productRepository.deleteById(productId);

return "product entry deleted";

}

return "product id does not exist";

}

}

**UserService.java**

package com.flipkart.shoppingkart.service;

import java.time.LocalDateTime;

import java.util.ArrayList;

import java.util.List;

import java.util.Optional;

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

import org.springframework.stereotype.Service;

import com.flipkart.shoppingkart.entity.Cart;

import com.flipkart.shoppingkart.entity.Product;

import com.flipkart.shoppingkart.entity.User;

import com.flipkart.shoppingkart.exception.ResourceNotFoundException;

import com.flipkart.shoppingkart.repository.CartRepository;

import com.flipkart.shoppingkart.repository.ProductRepository;

import com.flipkart.shoppingkart.repository.UserRepository;

@Service

public class UserService {

@Autowired

private UserRepository userRepository;

@Autowired

private ProductRepository productRepository;

@Autowired

private CartRepository cartRepository;

private User user = null;

//user registration

public User registerUser(User user)

{

return userRepository.save(user);

}

//userLogin

public String userLogin(String username, String password)

{

Optional userOpt = userRepository.findByUsernameIgnoreCase(username);

if(userOpt.isEmpty())

return "Check username";

User user = (User)userOpt.get();

if(!password.equals(user.getPassword()))

return "check password";

this.user = user;

return "user loggin successfull";

}

//find user

public User getUserById(Long userid)

{

return userRepository.findById(userid).orElseThrow(

()-> new ResourceNotFoundException("User is not available with id: "+userid)

);

}

//update users name

public String updateUserName(Long userid, String userName)

{

Optional userData = userRepository.findById(userid);

if(userData.isEmpty())

return "user is not found with id: "+userid;

User user = (User) userData.get();

user.setName(userName);

userRepository.save(user);

return "Data successfully updated"+user.toString();

}

//get list of products

public List<Product> getProductlist()

{

return productRepository.findAll();

}

//search product by id

public Product getProductById(Long productid)

{

return productRepository.findById(productid).orElseThrow(

()->new ResourceNotFoundException("product is not available with id: "+productid)

);

}

//search product by name

public Product getProductByName(String productName)

{

return productRepository.findByNameIgnoreCase(productName).orElseThrow(

()->new ResourceNotFoundException("product is not available with Name: "+productName)

);

}

//add to cart

public String addToCart(String productName, int quantity)

{

//check user loggedin or not

if(user==null)

return "login first";

//check product and quantites available or not

Optional productOpt = productRepository.findByNameIgnoreCase(productName);

if(productOpt.isEmpty())

return "Product is not available now";

Product product = (Product)productOpt.get();

if(quantity > product.getQuantity())

return "Only "+product.getQuantity()+" quantities are available";

// check if ited already added to users cart

Optional cartOpt = cartRepository.findByProductId(product.getId());

if(cartOpt.isEmpty())

{

Cart cart = new Cart();

cart.setProduct(product);

cart.setUser(this.user);

cart.setQuantity(quantity);

cart.setDate(LocalDateTime.now());

cartRepository.save(cart);

}

Cart cart =(Cart) cartOpt.get();

cart.setQuantity(quantity+cart.getQuantity());

product.setQuantity(product.getQuantity()-quantity);

productRepository.save(product);

return "Item Added to cart";

}

public List<String> getCart()

{

List<String> cartList = new ArrayList<>();

//check whether user loggedin or not

if(user == null)

{

cartList.add("Login first");

}

//check cart is empty or not

List<Cart> userCart = cartRepository.findAllByUserId(user.getId());

if(userCart.isEmpty())

cartList.add("No cart entry found");

//return all cart entry

else

for(Cart cart: userCart)

{

cartList.add(cart.toString());

}

return cartList;

}

public Double getCartAmount()

{

Double amount =0.0;

if(user == null)

{

return amount;

}

List<Cart> cartOpt = cartRepository.findAllByUserId(user.getId());

if(cartOpt.isEmpty())

return amount;

for(Cart cart: cartOpt)

{

amount += cart.getProduct().getPrice()\*cart.getQuantity();

}

return amount;

}

}

**Application.properties**

spring.application.name=ShoppingApp

# MySQL DB connection

spring.datasource.url=jdbc:mysql://localhost:3306/shopping\_app

spring.datasource.username=root

spring.datasource.password=root

spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

# JPA settings

spring.jpa.hibernate.ddl-auto=update

spring.jpa.show-sql=true

spring.jpa.database-platform=org.hibernate.dialect.MySQL8Dialect

pom.xml

<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>3.5.3</version>

<relativePath/> <!-- lookup parent from repository -->

</parent>

<groupId>com.Shopping</groupId>

<artifactId>ShoppingApp</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>ShoppingApp</name>

<description>Demo project for Spring Boot</description>

<url/>

<licenses>

<license/>

</licenses>

<developers>

<developer/>

</developers>

<scm>

<connection/>

<developerConnection/>

<tag/>

<url/>

</scm>

<properties>

<java.version>17</java.version>

</properties>

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-data-jpa</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<dependency>

<groupId>org.projectlombok</groupId>

<artifactId>lombok</artifactId>

<optional>true</optional>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-test</artifactId>

<scope>test</scope>

</dependency>

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

<version>8.0.33</version>

</dependency>

<!-- https://mvnrepository.com/artifact/org.junit.jupiter/junit-jupiter -->

<dependency>

<groupId>org.junit.jupiter</groupId>

<artifactId>junit-jupiter</artifactId>

<version>6.0.0-M1</version>

<scope>test</scope>

</dependency>

<!-- https://mvnrepository.com/artifact/org.mockito/mockito-core -->

<dependency>

<groupId>org.mockito</groupId>

<artifactId>mockito-core</artifactId>

<version>5.18.0</version>

<scope>test</scope>

</dependency>

<!-- https://mvnrepository.com/artifact/org.mockito/mockito-inline -->

<dependency>

<groupId>org.mockito</groupId>

<artifactId>mockito-inline</artifactId>

<version>5.2.0</version>

<scope>test</scope>

</dependency>

<!-- https://mvnrepository.com/artifact/org.springframework.boot/spring-boot-starter-test -->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-test</artifactId>

<version>3.5.3</version>

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-compiler-plugin</artifactId>

<configuration>

<annotationProcessorPaths>

<path>

<groupId>org.projectlombok</groupId>

<artifactId>lombok</artifactId>

</path>

</annotationProcessorPaths>

</configuration>

</plugin>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

<configuration>

<excludes>

<exclude>

<groupId>org.projectlombok</groupId>

<artifactId>lombok</artifactId>

</exclude>

</excludes>

</configuration>

</plugin>

<plugin>

<groupId>org.projectlombok</groupId>

<artifactId>lombok</artifactId>

</plugin>

</plugins>

</build>

</project>