Make an E-commerce Website for Sporty Shoes .

Course-end Project 1

DESCRIPTION

Project objective:

As a Full Stack Developer, complete the features of the application by planning the development and pushing the source code to the GitHub repository.

Background of the problem statement:

Sporty Shoes is a company that manufactures and sells sports shoes. They have a walk-in store, and now, they wish to launch their e-commerce portal sportyshoes.com.

You're asked to develop a prototype of the application. It will be then presented to the relevant stakeholders for budget approval. Your manager has set up a meeting where you're asked to do the following:

- Presenting the specification document which has the product's capabilities, appearance, and user interactions
- Setting up Git and GitHub account to store and track your enhancements of the prototype
- Explaining the Java concepts used in the project
- Discussing the generic features of the product:
- There will be an admin to manage the website. An administrator login will be required to access the admin page.

The admin should be able to change his password if he wants, he should be able to:

- Manage the products in the store including categorizing them
- Browse the list of users who have signed up and be able to search users
- See purchase reports filtered by date and category

Mandatary spring boot project file:

1. 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
https://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>2.7.14</version>
		<relativePath /> <!-- lookup parent from repository -->
	</parent>
	<groupId>com.simplilearn</groupId>
	<artifactId>SpringBootProjectSportyShoes</artifactId>
	<version>0.0.1-SNAPSHOT</version>
	<packaging>war</packaging>
	<name>SpringBootProjectSportyShoes</name>
	<description>Demo project for Spring Boot mvc</description>
	<properties>
		<java.version>1.8</java.version>
	</properties>
	<dependencies>
		<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-starter-web</artifactId>
		</dependency>


		<dependency>
			<groupId>jstl</groupId>
			<artifactId>jstl</artifactId>
			<version>1.2</version>
```

```
</dependency>

<dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-devtools</artifactId>
        <scope>runtime</scope>
        <optional>true</optional>
</dependency>
<dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-tomcat</artifactId>
        <scope>provided</scope>
</dependency>

<dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-data-jpa</artifactId>

</dependency>

<dependency>
        <groupId>javax.servlet</groupId>
        <artifactId>jstl</artifactId>
</dependency>


<dependency>
        <groupId>mysql</groupId>
        <artifactId>mysql-connector-java</artifactId>
```

```
            <version>8.0.33</version>
        </dependency>

        <dependency>
            <groupId>org.apache.tomcat.embed</groupId>
            <artifactId>tomcat-embed-jasper</artifactId>
        </dependency>

        <dependency>
            <groupId>com.h2database</groupId>
            <artifactId>h2</artifactId>
        </dependency>



        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>
</dependencies>

<build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
            </plugin>
        </plugins>
</build>
```

```
</project>
```

2.

```
package org.simplilearn;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class SpringMvcApp2Application {

	public static void main(String[] args) {
		SpringApplication.run(SpringMvcApp2Application.class, args);
	}

}
```

3.

```
package org.simplilearn;

import org.springframework.boot.builder.SpringApplicationBuilder;
import org.springframework.boot.web.servlet.support.SpringBootServletInitializer;

public class ServletInitializer extends SpringBootServletInitializer {

	@Override
	protected SpringApplicationBuilder configure(SpringApplicationBuilder application) {
```

```
		return application.sources(SpringMvcApp2Application.class);
	}

}
```

---

4. Entities

a. Cart

```
package org.simplilearn.entities;

import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;
import javax.persistence.OneToOne;

@Entity
public class Cart {
	@Id
	@GeneratedValue(strategy = GenerationType.IDENTITY)
	private int id;
	private String item;
	private int qty;
	private int price;
```

```
    @OneToOne
  @JoinColumn(name = "user_id")
  private User user;


    @ManyToOne(fetch = FetchType.LAZY) // Assuming many carts can be associated with one product
      @JoinColumn(name = "product_id") // Adjust the column name according to your database schema
      private Product product;


    public User getUser() {
        return user;
    }


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


    public Cart() {


    }


    public int getId() {
        return id;
    }


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

```
public String getItem() {
        return item;
}

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



public int getPrice() {
        return price;
}

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

public int getQty() {
        return qty;
}

public void setQty(int qty) {
        this.qty = qty;
}

 public Product getProduct() {
      return product;
```

```
        }

        public void setProduct(Product product) {
            this.product = product;
        }

}
```

```
package org.simplilearn.entities;

public class CartItem {
    private int pid;
    private String name;
    private int quantity;
    private int price;
    private Product product; // Add reference to the Product entity

    public CartItem(int pid, String name, int quantity, int price, Product product) {
        this.pid = pid;
        this.name = name;
        this.quantity = quantity;
        this.price = price;
        this.product = product;
    }

    // Getters and setters for all the fields (pid, name, quantity, price, product)

    public int getPid() {
        return pid;
```

```
}

public void setPid(int pid) {
   this.pid = pid;
}

public String getName() {
   return name;
}

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

public int getQuantity() {
   return quantity;
}

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

public int getPrice() {
   return price;
}

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

```
    public Product getProduct() {
        return product;
    }

    public void setProduct(Product product) {
        this.product = product;
    }
}
```

```
package org.simplilearn.entities;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;

@Entity
public class Category {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String name;

 public Category() {
            // TODO Auto-generated constructor stub
}

public Category(Long id, String name) {
```

```
        super();
        this.id = id;
        this.name = name;
}

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;
}

}
```

```
package org.simplilearn.entities;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
```

```
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;
import javax.persistence.OneToMany;
import javax.persistence.Table;

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

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

    private Date orderDate;

    @ManyToOne
    @JoinColumn(name = "user_id")
    private User user;

    @OneToMany(mappedBy = "order")
    private List<OrderItem> orderItems = new ArrayList<>();

    private int totalCartValue; // Add a field to store the total cart value
```

```
public Order() {
    // Empty constructor required by JPA
}

public Order(User user, List<OrderItem> orderItems, int totalCartValue) {
    this.user = user;
    this.orderItems = orderItems;
    this.totalCartValue = totalCartValue;
    this.orderDate = new Date(); // Set the order date to the current date
}

// Getters and setters for other fields...

public Long getId() {
    return id;
}

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

public Date getOrderDate() {
    return orderDate;
}

public void setOrderDate(Date orderDate) {
    this.orderDate = orderDate;
}
```

```
    public User getUser() {
        return user;
    }

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

    public List<OrderItem> getOrderItems() {
        return orderItems;
    }

    public void setOrderItems(List<OrderItem> orderItems) {
        this.orderItems = orderItems;
    }

    public int getTotalCartValue() {
        return totalCartValue;
    }

    public void setTotalCartValue(int totalCartValue) {
        this.totalCartValue = totalCartValue;
    }
}
```

```
package org.simplilearn.entities;
```

```
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;

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

    private String itemName;
    private int quantity;
    private int price;

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

    public OrderItem() {
        // Default constructor
    }

    // Constructor without Order parameter (we'll set it later in CartController)
    public OrderItem(String itemName, int quantity, int price) {
        this.itemName = itemName;
```

```
    this.quantity = quantity;
    this.price = price;
}




        public Long getId() {
    return id;
}

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

public String getItemName() {
    return itemName;
}

public void setItemName(String itemName) {
    this.itemName = itemName;
}

public int getQuantity() {
    return quantity;
}

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

```
    public int getPrice() {
        return price;
    }

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

    public Order getOrder() {
        return order;
    }

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


}
```

```
package org.simplilearn.entities;

import java.time.LocalDate;
import java.time.LocalDateTime;

import javax.persistence.CascadeType;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
```

```
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;
import javax.persistence.Table;

import org.hibernate.annotations.CreationTimestamp;

@Entity
@Table(name = "products")
public class Product {
	@Id
	@GeneratedValue(strategy = GenerationType.IDENTITY)
	private int pid;
	private String name;
	private int price;
	private String imageUrl;
	@CreationTimestamp
	private LocalDateTime creationDate;
	@ManyToOne(cascade = CascadeType.PERSIST)
	@JoinColumn(name = "uid")
	private User user;

	public int getPid() {
		return pid;
	}

	public void setPid(int pid) {
		this.pid = pid;
```

```
}

public String getName() {
        return name;
}

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

public int getPrice() {
        return price;
}

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

public String getImageUrl() {
        return imageUrl;
}

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

public LocalDateTime getCreationDate() {
```

```
		return creationDate;
	}

	public void setCreationDate(LocalDateTime creationDate) {
		this.creationDate = creationDate;
	}

	public User getUser() {
		return user;
	}

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

}
```

```
package org.simplilearn.entities;

import javax.persistence.*;
import java.util.Date;
import java.util.List;

@Entity
@Table(name = "purchases")
public class Purchase {
```

```
@Id
@GeneratedValue(strategy = GenerationType.IDENTITY)
private Long id;

@ManyToOne
@JoinColumn(name = "user_id")
private User user;

@Column(name = "purchase_date")
@Temporal(TemporalType.DATE)
private Date purchaseDate;

@ManyToOne // Assuming many purchases can belong to one category
@JoinColumn(name = "category_id") // The name of the foreign key column in the purchase table
private Category category;

@OneToMany(mappedBy = "purchase", cascade = CascadeType.ALL)
private List<PurchaseItem> purchaseItems;

@Column(name = "total_amount")
private double totalAmount;

public Purchase() {
		// TODO Auto-generated constructor stub
	}

	public Purchase(User user, List<PurchaseItem> purchaseItems, double totalAmount) {
		super();
```

```
        this.user = user;
        this.purchaseItems = purchaseItems;
        this.totalAmount = totalAmount;
}

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 Date getPurchaseDate() {
        return purchaseDate;
}

public void setPurchaseDate(Date purchaseDate) {
        this.purchaseDate = purchaseDate;
}
```

```
	public List<PurchaseItem> getPurchaseItems() {
		return purchaseItems;
	}

	public void setPurchaseItems(List<PurchaseItem> purchaseItems) {
		this.purchaseItems = purchaseItems;
	}

	public double getTotalAmount() {
		return totalAmount;
	}

	public void setTotalAmount(double totalAmount) {
		this.totalAmount = totalAmount;
	}

	public Category getCategory() {
		return category;
	}

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

}
```

```
package org.simplilearn.entities;

import javax.persistence.*;

@Entity
@Table(name = "purchase_items")
public class PurchaseItem {

   @Id
   @GeneratedValue(strategy = GenerationType.IDENTITY)
   private Long id;

   @ManyToOne
   @JoinColumn(name = "purchase_id")
   private Purchase purchase;

   @ManyToOne
   @JoinColumn(name = "product_id")
   private Product product;

   private int quantity;

   private double price;

   public PurchaseItem() {
            // TODO Auto-generated constructor stub
      }

      public PurchaseItem(Purchase purchase, Product product, int quantity, double price) {
```

```
        super();
        this.purchase = purchase;
        this.product = product;
        this.quantity = quantity;
        this.price = price;
}

public Long getId() {
        return id;
}

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

public Purchase getPurchase() {
        return purchase;
}

public void setPurchase(Purchase purchase) {
        this.purchase = purchase;
}

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 double getPrice() {
		return price;
	}

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

}
```

```
package org.simplilearn.entities;

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

import javax.persistence.CascadeType;
import javax.persistence.Entity;
```

```
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.OneToMany;
import javax.persistence.OneToOne;
import javax.persistence.Table;

@Entity
@Table(name = "users")
public class User {
	@Id
	@GeneratedValue(strategy = GenerationType.IDENTITY)
	private int id;
	private String username;
	private String password;
	private String email;
	private String role;

	@OneToOne(mappedBy = "user", cascade = CascadeType.ALL, orphanRemoval = true)
	@JoinColumn(name="cartId")
	private Cart cart;

	@OneToMany(mappedBy = "user")
	private List<Product> products = new ArrayList<>();

	@OneToMany(mappedBy = "user")
	private List<Cart> carts = new ArrayList<>();
```

```
    // Helper method to add a cart to the user
    public void addCart(Cart cart) {
        carts.add(cart);
        cart.setUser(this);
    }

    // Helper method to remove a cart from the user
    public void removeCart(Cart cart) {
        carts.remove(cart);
        cart.setUser(null);
    }

//helper method
    public void addProduct(Product product) {
        products.add(product);

    }


    public Cart getCart() {
        return cart;
    }


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

```
public String getEmail() {
	return email;
}

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

public int getId() {
	return id;
}

public void setId(int 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 getRole() {
		return role;
	}

	public void setRole(String role) {
		this.role = role;
	}

	public List<Product> getProducts() {
		return products;
	}

	public void setProducts(List<Product> products) {
		this.products = products;
	}

}
```

Controllers

```
package org.simplilearn.controllers;
```

```
import java.util.Date;
import java.util.List;

import javax.servlet.http.HttpSession;

import org.simplilearn.entities.Category;
import org.simplilearn.entities.Product;
import org.simplilearn.entities.Purchase;
import org.simplilearn.entities.User;
import org.simplilearn.models.LoginDto;
import org.simplilearn.models.UserDto;
import org.simplilearn.services.CategoryService;
import org.simplilearn.services.ProductService;
import org.simplilearn.services.UserService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.format.annotation.DateTimeFormat;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RequestParam;

@Controller
public class UserController {
	private UserService userService;
	private ProductService productService;
	private CategoryService categoryService;

	@Autowired
```

```
	public UserController(UserService userservice, ProductService productService, CategoryService
categoryService) {
		super();
		this.userService = userservice;
		this.productService = productService;
		this.categoryService = categoryService;
	}

	@RequestMapping("/")
	public String showHome() {
		return "home";
	}

	@RequestMapping("/showLogin")
	public String showLogin() {
		return "login";
	}

	@RequestMapping("/showSignUp")
	public String showSignUp() {
		return "signup";
	}

	@RequestMapping(value = "/signUp", method = RequestMethod.POST)
	public String register(@RequestParam("username") String username,
@RequestParam("password") String password,
			@RequestParam("email") String email, @RequestParam("role") String role,
Model model) {
		// String username = request.getParameter("username");
		// String password = request.getParameter("password");
```

```
		// String email=request.getParameter("email");
		UserDto userDto = new UserDto();
		userDto.setUsername(username);
		userDto.setPassword(password);
		userDto.setEmail(email);
		userDto.setRole(role);
		User user = userService.register(userDto);
		if (user != null)
			model.addAttribute("msg", "User Registered Successfully");
		else
			model.addAttribute("msg", "Not Registered");
		return "signup";
}

@RequestMapping(value = "/login", method = RequestMethod.POST)
public String login(@RequestParam("username") String username, Model model,
			@RequestParam("password") String password, HttpSession session) {
		String viewName = null;
		LoginDto loginDto = new LoginDto();
		loginDto.setUsername(username);
		loginDto.setPassword(password);
		User user = userService.login(loginDto);
		if (user != null) {
			session.setAttribute("user", user);
			if (user.getRole().equals("Admin"))
				viewName = "adminDashboard";
			else {
				List<Product> products = productService.getAll();
				model.addAttribute("products", products);
```

```
                    viewName = "customerDashboard";
                }

            } else {
                model.addAttribute("msg", "Username/Password is wrong");
                viewName = "login";
            }
            return viewName;
        }

        @RequestMapping(value = "/logout")
        public String logout(HttpSession session) {
            session.setAttribute("user", null);
            session.invalidate();
            return "redirect:/";
        }

        // Add this method to show the change password form
        @RequestMapping("/changePassword")
        public String showChangePasswordForm() {
            return "changePassword";
        }

        // Add this method to handle the password change request
        @RequestMapping(value = "/changePassword", method = RequestMethod.POST)
        public String changePassword(@RequestParam("oldPassword") String oldPassword,
                @RequestParam("newPassword") String newPassword, HttpSession session,
Model model) {
            // Get the user from the session
```

```
                User user = (User) session.getAttribute("user");

                // Check if the old password matches the user's current password
                if (user.getPassword().equals(oldPassword)) {
                        // If the old password matches, update the user's password to the new
password
                        user.setPassword(newPassword);
                        userService.updateUser(user); // Update the user's password in the repository
                        model.addAttribute("msg", "Password changed successfully");
                } else {
                        // If the old password does not match, show an error message
                        model.addAttribute("msg", "Incorrect old password");
                }

                // Redirect back to the adminDashboard page
                return "redirect:/adminDashboard";
        }

        @RequestMapping("/showUsers")
        public String showUsers(Model model) {
                List<User> users = userService.getAllUsers();
                model.addAttribute("users", users);
                return "viewUsers";
        }

        @RequestMapping(value = "/purchaseReport", method = RequestMethod.GET)
        public String showPurchaseReportForm(Model model) {
           List<Category> categories = categoryService.getAllCategories();
           model.addAttribute("categories", categories);
```

```
        return "purchaseReport";
    }


    @RequestMapping(value = "/purchaseReport", method = RequestMethod.POST)
    public String getFilteredPurchaseReport(
        @RequestParam("startDate") @DateTimeFormat(pattern = "yyyy-MM-dd") Date startDate,
        @RequestParam("endDate") @DateTimeFormat(pattern = "yyyy-MM-dd") Date endDate,
        @RequestParam("categoryId") Long categoryId, Model model) {
        Category category = categoryService.findById(categoryId);
        List<Purchase> purchases = userService.getPurchaseReportByDateAndCategory(startDate,
endDate, category);
        model.addAttribute("purchases", purchases);
        return "purchaseReport";
    }

}
```

```
package org.simplilearn.controllers;

import org.simplilearn.entities.*;
import org.simplilearn.models.PurchaseDto;
import org.simplilearn.services.PurchaseService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RequestParam;
```

```
import javax.servlet.http.HttpSession;

@Controller
public class PurchaseController {

    private PurchaseService purchaseService;

    @Autowired
    public PurchaseController(PurchaseService purchaseService) {
        this.purchaseService = purchaseService;
    }

    @RequestMapping(value = "/makePayment", method = RequestMethod.GET)
    public String makePayment(HttpSession session) {
        // Retrieve cartItems and totalCartValue from session
        // Perform payment processing
        // Create a PurchaseDto with user and cartItems data
        // Save the purchase using purchaseService.savePurchase(purchaseDto)

        // After successful purchase, you can redirect to the payment success page
        return "payment_success";
    }
}
```

```
package org.simplilearn.controllers;

import java.util.List;
```

```
import javax.servlet.http.HttpSession;

import org.simplilearn.entities.Product;
import org.simplilearn.entities.User;
import org.simplilearn.models.ProductDto;
import org.simplilearn.services.ProductService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;

@Controller
public class ProductController {
	private ProductService productService;

	@Autowired
	public ProductController(ProductService productService) {
		super();
		this.productService = productService;
	}

	@RequestMapping("/showAdd")
	public String showAddProduct() {
		return "addProduct";
	}
```

```
	@RequestMapping("/add")
	public String addProduct(@RequestParam("name") String name,
				@RequestParam("price") String price,
				@RequestParam("imageUrl") String imageUrl, HttpSession httpSession) {
		int price1 = Integer.parseInt(price);
		ProductDto productDto = new ProductDto(name, price1, imageUrl);
		User user = (User) httpSession.getAttribute("user");

		productService.insertProduct(productDto, user);
		return "adminDashboard";
	}

	@GetMapping("/showProducts")
	public String showProducts(Model model) {
		List<Product> products = productService.getAll();
		model.addAttribute("products", products);
		return "adminDashboard";
	}
	@RequestMapping("/delete/{pid}")
	public String deleteProduct(@PathVariable("pid") int pid) {
		productService.deleteProduct(pid);
		return "redirect:/showProducts";
	}
}
```

```
package org.simplilearn.controllers;
```

```
import org.simplilearn.entities.Category;
import org.simplilearn.models.CategoryDto;
import org.simplilearn.services.CategoryService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RequestParam;

import java.util.List;

@Controller
public class CategoryController {
   private CategoryService categoryService;

   @Autowired
   public CategoryController(CategoryService categoryService) {
      this.categoryService = categoryService;
   }

   // Show category creation form
   @RequestMapping("/showAddCategory")
   public String showAddCategoryForm() {
      return "addCategory";
   }
```

```
    @PostMapping("/addCategory")
    public String addCategory(@RequestParam("name") String name, Model model) {
        CategoryDto categoryDto = new CategoryDto();
        categoryDto.setName(name);

        Category category = categoryService.createCategory(categoryDto);
        if (category != null) {
            model.addAttribute("msg", "Category created successfully");
        } else {
            model.addAttribute("msg", "Failed to create category");
        }

        return "addCategory"; // Redirect back to the category creation form
    }

    // Show all categories
    @RequestMapping("/showCategories")
    public String showCategories(Model model) {
        List<Category> categories = categoryService.getAllCategories();
        model.addAttribute("categories", categories);
        return "categories";
    }
}
```

```
package org.simplilearn.controllers;
```

```
import java.util.Date;
import java.util.List;
import java.util.stream.Collectors;

import javax.servlet.http.HttpSession;

import org.simplilearn.entities.CartItem;
import org.simplilearn.entities.Order;
import org.simplilearn.entities.OrderItem;
import org.simplilearn.entities.Product;
import org.simplilearn.entities.User;
import org.simplilearn.models.CartModel;
import org.simplilearn.services.CartService;
import org.simplilearn.services.OrderService;
import org.simplilearn.services.ProductService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;

@Controller
public class CartController {
	private ProductService productService;
	private CartService cartService;
	private OrderService orderService;

	@Autowired
```

```
	public CartController(ProductService productService, CartService cartService, OrderService
orderService) {
		this.productService = productService;
		this.cartService = cartService;
		this.orderService = orderService;
	}

	@RequestMapping("/showCart")
	public String showCart(HttpSession session, Model model) {
		User user = (User) session.getAttribute("user");
		if (user != null) {
		List<CartItem> cartItems = cartService.getCartItems(user);
		int totalCartValue = cartService.calculateTotalCartValue(user);

		model.addAttribute("cartItems", cartItems);
		model.addAttribute("totalCartValue", totalCartValue);
		}else {
			return "redirect:/login";
		}
		return "cart";
	}

	@RequestMapping("/addToCart/{pid}")
   public String addToCart(@PathVariable("pid") int pid, HttpSession session) {
      Product product = productService.getProduct(pid);
      CartModel cartModel = new CartModel(product.getName(), product.getPrice());
      User user = (User) session.getAttribute("user");
      cartService.addToCart(user, cartModel);
      return "redirect:/showCart";
```

```
    }

    @RequestMapping("/removeFromCart/{pid}")
    public String removeFromCart(@PathVariable("pid") int pid, HttpSession session) {
        User user = (User) session.getAttribute("user");
        cartService.removeFromCart(user, pid);
        return "redirect:/showCart";
    }

    @RequestMapping("/checkout")
    public String checkout(HttpSession session, Model model) {
        User user = (User) session.getAttribute("user");

        // Get the cart items and total cart value
        List<CartItem> cartItems = cartService.getCartItems(user);
        int totalCartValue = cartService.calculateTotalCartValue(user);

        // Create a new order with the cart items and total cart value
        Order order = new Order();
        order.setUser(user);
        order.setOrderDate(new Date());
        order.setTotalCartValue(totalCartValue);

        // Create a new OrderItem for each CartItem in the cart and add them to the order
        List<OrderItem> orderItems = cartItems.stream().map(cartItem -> {
            OrderItem orderItem = new OrderItem(cartItem.getName(), cartItem.getQuantity(),
cartItem.getPrice());
            orderItem.setOrder(order);
            return orderItem;
```

```
        }).collect(Collectors.toList());

        order.setOrderItems(orderItems);

        // Save the order entity to the database
        orderService.saveOrder(order);

        // Clear the cart for the user after checkout
        cartService.clearCart(user);

        // Set a session attribute to indicate successful checkout
        session.setAttribute("checkoutSuccess", true);

        // Redirect to the payment page before redirecting to the dashboard
        return "redirect:/makePayment";
    }

    @RequestMapping("/makePayment")
    public String makePayment(HttpSession session) {
        // In a real application, you would implement payment gateway integration here
        User user = (User) session.getAttribute("user");
        session.removeAttribute("checkoutSuccess"); // Clear the checkout success attribute
        return "redirect:/checkout_success";
    }

    @RequestMapping("/checkout_success")
    public String checkoutSuccess(HttpSession session) {
        // Get the user from the session
        User user = (User) session.getAttribute("user");
```

```
        // Redirect to the user's dashboard based on their role (e.g., admin or regular user)
        if (user.getRole().equals("admin")) {
            return "redirect:/adminDashboard";
        } else {
            return "redirect:/customerDashboard";
        }
    }
    @RequestMapping("/purchase")
    public String purchase(HttpSession session, Model model) {
        User user = (User) session.getAttribute("user");

        // Get the cart items and total cart value
        List<CartItem> cartItems = cartService.getCartItems(user);
        int totalCartValue = cartService.calculateTotalCartValue(user);

        model.addAttribute("cartItems", cartItems);
        model.addAttribute("totalCartValue", totalCartValue);

        return "purchase";
    }


}
```

Model (DTO)

```
package org.simplilearn.models;
```

```
public class CartModel {
private String item;
private int price;

public CartModel() {
	// TODO Auto-generated constructor stub
}

public CartModel(String item, int price) {
	super();
	this.item = item;
	this.price = price;
}

public String getItem() {
	return item;
}

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

public int getPrice() {
	return price;
}

public void setPrice(int price) {
	this.price = price;
```

```
}

}
```

```
package org.simplilearn.models;

public class CategoryDto {
   private String name;

 public CategoryDto() {
        // TODO Auto-generated constructor stub
}

public CategoryDto(String name) {
        super();
        this.name = name;
}

public String getName() {
        return name;
}

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

```
}
```

```
package org.simplilearn.models;

public class LoginDto {
private String username;
private String password;

public LoginDto() {
	// TODO Auto-generated constructor stub
}

public LoginDto(String username, String password) {
	super();
	this.username = username;
	this.password = password;
}

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;
}

}
```

```
package org.simplilearn.models;

import java.util.List;

public class OrderDTO {
   private Long userId;
   private List<OrderItemDTO> orderItems;

   public Long getUserId() {
      return userId;
   }

   public void setUserId(Long userId) {
      this.userId = userId;
   }

   public List<OrderItemDTO> getOrderItems() {
```

```
        return orderItems;
    }

    public void setOrderItems(List<OrderItemDTO> orderItems) {
        this.orderItems = orderItems;
    }
}
```

```
package org.simplilearn.models;

public class OrderItemDTO {
    private String itemName;
    private int quantity;
    private int price;

    public String getItemName() {
        return itemName;
    }

    public void setItemName(String itemName) {
        this.itemName = itemName;
    }

    public int getQuantity() {
        return quantity;
    }

    public void setQuantity(int quantity) {
        this.quantity = quantity;
```

```
    }

    public int getPrice() {
        return price;
    }

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

```
package org.simplilearn.models;

import org.springframework.data.annotation.CreatedDate;

public class ProductDto {
        private String name;
        private int price;
        private String imageUrl;
public ProductDto() {
        // TODO Auto-generated constructor stub
}
public ProductDto(String name, int price, String imageUrl) {
        super();
        this.name = name;
        this.price = price;
        this.imageUrl = imageUrl;
}
```

```
public String getName() {
	return name;
}
public void setName(String name) {
	this.name = name;
}
public int getPrice() {
	return price;
}
public void setPrice(int price) {
	this.price = price;
}
public String getImageUrl() {
	return imageUrl;
}
public void setImageUrl(String imageUrl) {
	this.imageUrl = imageUrl;
}

}
```

```
package org.simplilearn.models;

import org.simplilearn.entities.CartItem;
import org.simplilearn.entities.User;

import java.util.List;
```

```
public class PurchaseDto {

    private User user;
    private List<CartItem> cartItems;

    public PurchaseDto() {
            // TODO Auto-generated constructor stub
        }

        public PurchaseDto(User user, List<CartItem> cartItems) {
            super();
            this.user = user;
            this.cartItems = cartItems;
        }

        public User getUser() {
            return user;
        }

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

        public List<CartItem> getCartItems() {
            return cartItems;
        }

        public void setCartItems(List<CartItem> cartItems) {
            this.cartItems = cartItems;
```

```
	}

}
```

```
package org.simplilearn.models;

public class UserDto {
private String username;
private String password;
private String email;
private String role;

public UserDto() {
	// TODO Auto-generated constructor stub
}

public UserDto(String username, String password, String email, String role) {
	super();
	this.username = username;
	this.password = password;
	this.email = email;
	this.role = role;
}

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 getRole() {
	return role;
}

public void setRole(String role) {
	this.role = role;
}

}
```

Repositories

```
package org.simplilearn.repositories;

import org.simplilearn.entities.Cart;
import org.simplilearn.entities.User;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

import java.util.List;

@Repository
public interface CartRepository extends JpaRepository<Cart, Integer> {
   List<Cart> findByUser(User user);
   Cart findByItemAndUser(String item, User user);
}
```

```
package org.simplilearn.repositories;

import org.simplilearn.entities.Category;
import org.springframework.data.jpa.repository.JpaRepository;

public interface CategoryRepository extends JpaRepository<Category, Long> {
   // Add custom queries for Category if needed
}
```

```
package org.simplilearn.repositories;

import java.util.List;

import org.simplilearn.entities.Order;
import org.simplilearn.entities.User;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Query;

public interface OrderRepository extends JpaRepository<Order, Long> {
   @Query("SELECT o FROM Order o WHERE o.user = ?1")
   List<Order> findByUser(User user);
}
```

```
package org.simplilearn.repositories;

import org.simplilearn.entities.Product;
import org.springframework.data.jpa.repository.JpaRepository;

public interface ProductRepository extends JpaRepository<Product, Integer>{

}
```

```
package org.simplilearn.repositories;

import java.util.Date;
```

```
import java.util.List;

import org.simplilearn.entities.Category;
import org.simplilearn.entities.Purchase;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

@Repository
public interface PurchaseRepository extends JpaRepository<Purchase, Long> {
        List<Purchase> findByPurchaseDateBetween(Date startDate, Date endDate);

   List<Purchase> findByCategory(Category category);

   List<Purchase> findByPurchaseDateBetweenAndCategory(Date startDate, Date endDate, Category
category);
}
```

```
package org.simplilearn.repositories;

import org.simplilearn.entities.User;
import org.springframework.data.jpa.repository.JpaRepository;

public interface UserRepository extends JpaRepository<User, Integer>{
User findByUsernameAndPassword(String username, String password);
User findByUsername(String username);
}
```

Services

```
package org.simplilearn.services;

import org.simplilearn.entities.Category;
import org.simplilearn.models.CategoryDto;

import java.util.List;

public interface CategoryService {
   Category createCategory(CategoryDto categoryDto);

   List<Category> getAllCategories();
   Category findById(Long categoryId);
}
```

```
package org.simplilearn.services;

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

import org.simplilearn.entities.Cart;
import org.simplilearn.entities.CartItem;
import org.simplilearn.entities.Product;
import org.simplilearn.entities.User;
import org.simplilearn.models.CartModel;
import org.simplilearn.repositories.CartRepository;
import org.simplilearn.repositories.UserRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
```

```
@Service
public class CartService {
    private CartRepository cartRepository;
    private UserRepository userRepository;

    @Autowired
    public CartService(CartRepository cartRepository, UserRepository userRepository) {
        this.cartRepository = cartRepository;
        this.userRepository = userRepository;
    }

    public void addToCart(User user, CartModel cartModel) {
        String item = cartModel.getItem(); // Get the item from the cartModel
        Cart cart = cartRepository.findByItemAndUser(cartModel.getItem(), user);


        if (cart != null) {
            // If the item already exists in the cart, update the quantity and price
            cart.setQty(cart.getQty() + 1);
            cart.setPrice(cart.getQty() * cartModel.getPrice());
        } else {
            // If the item is not in the cart, create a new cart item
            cart = new Cart();
            cart.setItem(item);
            cart.setQty(1);
            cart.setPrice(cartModel.getPrice());
            cart.setUser(user); // Associate the cart with the user
        }
```

```
        cartRepository.save(cart); // Save the cart (either updated or new)
    }




    public List<CartItem> getCartItems(User user) {
        List<CartItem> cartItems = new ArrayList<>();
        Cart cart = user.getCart();
        if (cart != null) {
            List<Cart> cartItemsFromDB = cartRepository.findByUser(user);
            for (Cart cartItem : cartItemsFromDB) {
                // Assuming there is a method getProduct() in the Cart entity to get the associated product
                Product product = cartItem.getProduct(); // Get the associated product
                CartItem item = new CartItem(cartItem.getId(), cartItem.getItem(), cartItem.getQty(),
cartItem.getPrice(), product);
                cartItems.add(item);
            }
        }
        return cartItems;
    }


    public void removeFromCart(User user, int pid) {
        Cart cart = user.getCart();
        if (cart != null) {
            List<Cart> cartItems = cartRepository.findByUser(user);
            for (Cart cartItem : cartItems) {
                if (cartItem.getId() == pid) {
```

```
                cartRepository.delete(cartItem);
                break;
            }
        }
    }
}

public int calculateTotalCartValue(User user) {
    List<Cart> cartItems = cartRepository.findByUser(user);
    return cartItems.stream().mapToInt(Cart::getPrice).sum();
}


// Method to clear the cart items for the given user
public void clearCart(User user) {
    List<Cart> cartItems = cartRepository.findByUser(user);
    cartRepository.deleteAll(cartItems);
}
}
```

```
package org.simplilearn.services;

import java.util.Date;
import java.util.List;

import org.simplilearn.entities.Category;
import org.simplilearn.entities.Purchase;
import org.simplilearn.entities.User;
```

```
import org.simplilearn.models.LoginDto;

import org.simplilearn.models.UserDto;


public interface UserService {

User register (UserDto userDto);

void updateUser(User user);

User login(LoginDto loginDto);

List<User> getAllUsers();

List<Purchase> getPurchaseReportByDateAndCategory(Date startDate, Date endDate, Category
category);

}
```

```
package org.simplilearn.services;


import java.util.Date;

import java.util.List;


import org.simplilearn.entities.Category;

import org.simplilearn.entities.Purchase;

import org.simplilearn.entities.User;

import org.simplilearn.models.LoginDto;

import org.simplilearn.models.UserDto;

import org.simplilearn.repositories.PurchaseRepository;

import org.simplilearn.repositories.UserRepository;

import org.springframework.stereotype.Service;


@Service

public class UserServiceImpl implements UserService{
```

```
private UserRepository userRepository;

private PurchaseRepository purchaseRepository;



	public UserServiceImpl(UserRepository userRepository, PurchaseRepository
purchaseRepository) {

	super();

	this.userRepository = userRepository;

	this.purchaseRepository = purchaseRepository;

}



	@Override

	public User register(UserDto userDto) {

		User user = new User();

		user.setUsername(userDto.getUsername());

		user.setPassword(userDto.getPassword());

		user.setEmail(userDto.getEmail());

		user.setRole(userDto.getRole());

		return userRepository.save(user);

	}



	@Override

	public User login(LoginDto loginDto) {

		String username=loginDto.getUsername();

		String password=loginDto.getPassword();

		User user=userRepository.findByUsernameAndPassword(username, password);
```

```
            return user;
        }


        @Override
        public void updateUser(User user) {
            userRepository.save(user);

        }


        @Override
        public List<User> getAllUsers() {
            // TODO Auto-generated method stub
            return userRepository.findAll();
        }


        @Override
          public List<Purchase> getPurchaseReportByDateAndCategory(Date startDate, Date endDate,
Category category) {
            if (startDate == null || endDate == null || category == null) {
               throw new IllegalArgumentException("Invalid filter criteria");
            }

            return purchaseRepository.findByPurchaseDateBetweenAndCategory(startDate, endDate,
category);
          }
        }
```

```
package org.simplilearn.services;

import org.simplilearn.entities.Purchase;
import org.simplilearn.models.PurchaseDto;

public interface PurchaseService {
   Purchase savePurchase(PurchaseDto purchaseDto);
}
```

```
package org.simplilearn.services;

import org.simplilearn.entities.CartItem;
import org.simplilearn.entities.Purchase;
import org.simplilearn.entities.PurchaseItem;
import org.simplilearn.models.PurchaseDto;
import org.simplilearn.repositories.PurchaseRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.Date;
import java.util.List;
import java.util.stream.Collectors;

@Service
public class PurchaseServiceImpl implements PurchaseService {
```

```
    private PurchaseRepository purchaseRepository;

    @Autowired
    public PurchaseServiceImpl(PurchaseRepository purchaseRepository) {
        this.purchaseRepository = purchaseRepository;
    }

    @Override
    public Purchase savePurchase(PurchaseDto purchaseDto) {
        Purchase purchase = new Purchase();
        purchase.setUser(purchaseDto.getUser());
        purchase.setPurchaseDate(new Date());

        List<PurchaseItem> purchaseItems = purchaseDto.getCartItems().stream().map(cartItem -> {
            PurchaseItem purchaseItem = new PurchaseItem();
            purchaseItem.setPurchase(purchase);
            purchaseItem.setProduct(cartItem.getProduct());
            purchaseItem.setQuantity(cartItem.getQuantity());
            purchaseItem.setPrice(cartItem.getPrice());
            return purchaseItem;
        }).collect(Collectors.toList());

        purchase.setPurchaseItems(purchaseItems);

        double totalAmount = purchaseItems.stream().mapToDouble(item -> item.getQuantity() *
item.getPrice()).sum();
        purchase.setTotalAmount(totalAmount);

        return purchaseRepository.save(purchase);
```

```
    }
}
```

```
package org.simplilearn.services;

import java.util.List;

import org.simplilearn.entities.Product;
import org.simplilearn.entities.User;
import org.simplilearn.models.ProductDto;

public interface ProductService {
void insertProduct(ProductDto productDto, User user);
void deleteProduct(int pid);
List<Product> getAll();
Product getProduct(int pid);
}
```

```
package org.simplilearn.services;

import java.util.List;

import org.simplilearn.entities.Product;
import org.simplilearn.entities.User;
import org.simplilearn.models.ProductDto;
import org.simplilearn.repositories.ProductRepository;
import org.simplilearn.repositories.UserRepository;
```

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

import org.springframework.stereotype.Service;

@Service

public class ProductServiceImpl implements  ProductService{

private ProductRepository productRepository;

private UserRepository userRepository;

@Autowired

	public ProductServiceImpl(ProductRepository productRepository, UserRepository
userRepository) {

	super();

	this.productRepository = productRepository;

	this.userRepository=userRepository;

}


	@Override

	public void insertProduct(ProductDto productDto, User user) {

		User user1=userRepository.findByUsernameAndPassword(user.getUsername(),
user.getPassword());

		Product product=new Product();

		product.setName(productDto.getName());

		product.setPrice(productDto.getPrice());

		product.setImageUrl(productDto.getImageUrl());

		user1.addProduct(product);

		product.setUser(user1);

		productRepository.save(product);


	}


	@Override
```

```
	public void deleteProduct(int pid) {
		productRepository.deleteById(pid);

	}

	@Override
	public List<Product> getAll() {

		return productRepository.findAll();
	}

	@Override
	public Product getProduct(int pid) {

		return productRepository.findById(pid).orElse(null);
	}

}
```

```
package org.simplilearn.services;

import java.util.List;

import org.simplilearn.entities.Order;
import org.simplilearn.entities.User;

public interface OrderService {
   void saveOrder(Order order);
```

```
    List<Order> getOrdersByUser(User user);
}
```

```
package org.simplilearn.services;

import java.util.List;

import org.simplilearn.entities.Order;
import org.simplilearn.entities.User;
import org.simplilearn.repositories.OrderRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

@Service
public class OrderServiceImpl implements OrderService {
    private OrderRepository orderRepository;

    @Autowired
    public OrderServiceImpl(OrderRepository orderRepository) {
        this.orderRepository = orderRepository;
    }

    @Override
    public void saveOrder(Order order) {
        orderRepository.save(order);
    }

    @Override
    public List<Order> getOrdersByUser(User user) {
```

```
        return orderRepository.findByUser(user);
    }
}
```

```
package org.simplilearn.services;

import org.simplilearn.entities.Category;
import org.simplilearn.models.CategoryDto;
import org.simplilearn.repositories.CategoryRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.List;
import java.util.Optional;

@Service
public class CategoryServiceImpl implements CategoryService {
    private CategoryRepository categoryRepository;

    @Autowired
    public CategoryServiceImpl(CategoryRepository categoryRepository) {
        this.categoryRepository = categoryRepository;
    }

    @Override
    public Category createCategory(CategoryDto categoryDto) {
        Category category = new Category();
        category.setName(categoryDto.getName());
```

```
        return categoryRepository.save(category);
    }

    @Override
    public List<Category> getAllCategories() {
        return categoryRepository.findAll();
    }

    @Override
    public Category findById(Long categoryId) {
        Optional<Category> categoryOptional = categoryRepository.findById(categoryId);
        return categoryOptional.orElse(null);

            }
}
```

JSP

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
        pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Checkout Success</title>
</head>
<body>
        <h1>Checkout Successful</h1>
```

```
        <p>Thank you for your order!</p>
        <p>Your payment was successful, and your order has been placed.</p>
        <p>Your order will be processed and delivered soon.</p>
        <a href="/home">Back to Home</a>
</body>
</html>
```

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
        pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Change Password</title>
</head>
<body>
        <h1>Change Password</h1>
        <form method="post" action="./changePassword">
                <label for="oldPassword">Old Password:</label> <input type="password"
                        name="oldPassword" required><br> <label
                        for="newPassword">New Password:</label> <input type="password"
                        name="newPassword" required><br> <input type="submit"
                        value="Change Password">
        </form>
</body>
</html>
```

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1" pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
  <meta charset="ISO-8859-1">
  <title>Categories</title>
</head>
<body>
  <h1>Categories</h1>
  <table border="1">
    <tr>
      <th>ID</th>
      <th>Name</th>
    </tr>
    <%-- Importing the necessary classes --%>
    <%@ page import="java.util.List" %>
    <%@ page import="org.simplilearn.entities.Category" %>

    <%
    List<Category> categories = (List<Category>) request.getAttribute("categories");
    for (Category category : categories) {
    %>
      <tr>
        <td><%= category.getId() %></td>
        <td><%= category.getName() %></td>
      </tr>
    <% } %>
  </table>
```

```
</body>
</html>
```

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Cart</title>
</head>
<body>
    <h1>Your Cart</h1>
    <table border="1">
        <tr>
            <th>Item Name</th>
            <th>Quantity</th>
            <th>Price</th>
            <th>Remove</th>
        </tr>
        <c:forEach var="item" items="${cartItems}">
            <tr>
                <td>${item.name}</td>
                <td>${item.quantity}</td>
                <td>${item.price}</td>
                <td><a href="./removeFromCart/${item.pid}">Remove</a></td>
            </tr>
```

```
                </c:forEach>
        </table>
        <p>Total Cart Value: ${totalCartValue}</p>
        <a href="./checkout">Proceed to Checkout</a>
</body>
</html>
```

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
        pageEncoding="ISO-8859-1"%>
<%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
        <% if (session.getAttribute("user")!=null)
        {
        %>
        <h1>Hi ${sessionScope.user.username}, Welcome to Admin Dashboard</h1>
        <a href="./showAdd">Add Product</a>
        <a href="./showProducts">Show Products</a>
        <a href="./showCategories">Manage Categories</a>
        <a href="showAddCategory">Add Category</a>
```

```
		<a href="./showUsers">View Users</a>
		<a href="./purchaseReport">Purchase Report</a>
		<a href="./changePassword">Change Password</a>
		<a href="./logout">Logout</a>

		<table border="1">
			<tr>
				<th>Name</th>
				<th>Price</th>
				<th>Image</th>
			</tr>
			<c:forEach var="product" items="${products }">
				<tr>
					<td>${product.name }</td>
					<td>${product.price }</td>
					<td><img style="width: 100px; height: 100px"
						src="${product.imageUrl} alt=" ${product.name}"/></td>
					<td><a href="./delete/${product.pid }">Delete</a></td>
				</tr>
			</c:forEach>
		</table>
		<%
			}
else{
		RequestDispatcher rd = request.getRequestDispatcher("/showLogin");
		request.setAttribute("msg", "Please login First");
		rd.forward(request,response);
}
			%>
```

```
</body>
</html>
```

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
    <form action="./add" method="post">

        Name: <input type="text" name="name"><br> Price: <input
            type="text" name="price"><br> ImageUrl: <input
            type="text" name="imageUrl"><br> <input type="submit"
            value="Submit">
    </form>
</body>
</html>
```

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Add Category</title>
```

```
</head>
<body>
	<h1>Add Category</h1>
	<form action="addCategory" method="post">
		Name: <input type="text" name="name" required> <input
			type="submit" value="Add Category">
	</form>
	<p>${msg}</p>
</body>
</html>
```

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1" pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
  <meta charset="ISO-8859-1">
  <title>Checkout</title>
</head>
<body>
  <h1>Checkout</h1>
  <table border="1">
    <tr>
      <th>Item Name</th>
      <th>Quantity</th>
      <th>Price</th>
    </tr>
    <%-- Importing the necessary classes --%>
    <%@ page import="java.util.List" %>
```

```
    <%@ page import="org.simplilearn.entities.CartItem" %>

    <%
    List<CartItem> cartItems = (List<CartItem>) request.getAttribute("cartItems");
    for (CartItem item : cartItems) {
    %>
      <tr>
        <td><%= item.getName() %></td>
        <td><%= item.getQuantity() %></td>
        <td><%= item.getPrice() %></td>
      </tr>
    <% } %>
  </table>
  <p>Total Cart Value: <%= request.getAttribute("totalCartValue") %></p>
  <a href="./makePayment">Proceed to Payment</a> <!-- Update the URL to point to "makePayment" -
->
</body>
</html>
```

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
        pageEncoding="ISO-8859-1"%>
<%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
```

```
<body>
	<h1>Hi ${sessionScope.user.username}, Welcome to Customer
		Dashboard</h1>
	<table border="1">
		<tr>
			<th>Name</th>
			<th>Price</th>
			<th>Image</th>
		</tr>
		<c:forEach var="product" items="${products }">
			<tr>
				<td>${product.name }</td>
				<td>${product.price }</td>
				<td><img style="width: 100px; height: 100px"
					src="${product.imageUrl}" alt="${product.name}" /></td>
				<td><a href="./addToCart/${product.pid}">Add to Cart</a></td>
			</tr>
		</c:forEach>
	</table>
</body>
</html>
```

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1" pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
```

```
    <meta charset="ISO-8859-1">
    <title>Sporty Shoes - Welcome to Home Page</title>
</head>
<body>
    <h1>Welcome to Sporty Shoes</h1>
    <p>Developed by: Om Shankar Mishra</p>
    <p>Your one-stop destination for trendy and comfortable sporty footwear.</p>
    <a href="./showLogin">Login</a>
    <a href="./showSignUp">Signup</a>
    <a href="./showProducts">View Products</a>
</body>
</html>
```

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
        pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
        <form action="./login" method="post">
                Username: <input type="text" name="username"><br>
                Password: <input type="password" name="password"><br> <input
                        type="submit" value="Submit">
        </form>
        <p>${msg}</p>
```

```
</body>
</html>
```

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Make Payment</title>
</head>
<body>
    <h1>Payment Process</h1>
    <p>Your payment is being processed.</p>
    <p>For learning purposes, this is just a placeholder page.</p>
    <p>In a real application, you would implement the payment gateway
        integration here.</p>
    <a href="./checkout_success">Back to Home</a>
    <!-- Update the URL to point to "checkout_success" -->
</body>
</html>
```

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Payment</title>
```

```
</head>
<body>
	<h1>Payment Page</h1>
	<!-- Add a form for payment details -->
	<form action="/processPayment" method="post">
		<!-- Add input fields for card number, expiration date, CVV, etc. -->
		<!-- For simplicity, I'm just adding a single field for demonstration purposes -->
		Card Number: <input type="text" name="cardNumber"><br>
		<!-- Add other fields here as needed -->

		<input type="submit" value="Make Payment">
	</form>
</body>
</html>
```

```
<%@page import="org.simplilearn.entities.CartItem"%>
<%@page import="java.util.List"%>
<%@ page language="java" contentType="text/html; charset=ISO-8859-1" pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
  <meta charset="ISO-8859-1">
  <title>Cart</title>
</head>
<body>
  <h1>Your Cart</h1>
  <table border="1">
    <tr>
```

```
            <th>Item Name</th>
            <th>Quantity</th>
            <th>Price</th>
            <th>Action</th>
        </tr>
        <%-- Loop through the cart items and display details --%>
        <%
            List<CartItem> cartItems = (List<CartItem>) request.getAttribute("cartItems");
            for (CartItem item : cartItems) {
        %>
        <tr>
            <td><%= item.getName() %></td>
            <td><%= item.getQuantity() %></td>
            <td><%= item.getPrice() %></td>
            <td>
                <a href="./removeFromCart/<%= item.getPid() %>">Remove</a>
            </td>
        </tr>
        <%-- End of loop --%>
        <% } %>
    </table>
    <p>Total Cart Value: <%= request.getAttribute("totalCartValue") %></p>
    <a href="./checkout">Proceed to Checkout</a>
</body>
</html>
```

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
```

```
        pageEncoding="ISO-8859-1"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Purchase Report</title>
</head>
<body>
        <h1>Purchase Report</h1>

        <form method="post" action="/purchaseReport">
                <label for="startDate">Start Date:</label> <input type="date"
                        id="startDate" name="startDate" required> <label
                        for="endDate">End Date:</label> <input type="date" id="endDate"
                        name="endDate" required> <label for="categoryId">Category:</label>
                <select id="categoryId" name="categoryId" required>
                        <!-- Populate the options with available categories from the database -->
                        <c:forEach var="category" items="${categories}">
                                <option value="${category.id}">${category.name}</option>
                        </c:forEach>
                </select>

                <button type="submit">Filter</button>
        </form>

        <!-- Display the filtered purchase report -->
        <table>
                <tr>
```

```
                    <th>Purchase Date</th>
                    <th>Category</th>
                    <th>Total Amount</th>
                </tr>
                <c:forEach var="purchase" items="${purchases}">
                    <tr>
                        <td>${purchase.purchaseDate}</td>
                        <td>${purchase.category.name}</td>
                        <td>${purchase.totalAmount}</td>
                    </tr>
                </c:forEach>
        </table>
</body>
</html>
```

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
        pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
        <form action="./signUp" method="post">
                Username: <input type="text" name="username"><br>
                Password: <input type="password" name="password"><br>
                Email:<input type="email" name="email"><br>
```

```
                    <select name="role">
          <option value="Customer">Customer</option>
          <option value="Admin">Admin</option>
       </select><br>
                    <input
                              type="submit" value="Submit">


          </form>
          <p style="color: red">${msg }</p>
</body>
</html>
```

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
          pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>


<html>
<head>
<meta charset="ISO-8859-1">
<title>View Users</title>
</head>
<body>
          <h1>View Users</h1>
          <table border="1">
                    <tr>
                              <th>Username</th>
                              <th>Email</th>
                              <th>Role</th>
```

```
                </tr>
                <c:forEach var="user" items="${users}">
                        <tr>
                                <td>${user.username}</td>
                                <td>${user.email}</td>
                                <td>${user.role}</td>
                        </tr>
                </c:forEach>
        </table>
        <a href="./adminDashboard">Back to Dashboard</a>
</body>
</html>
```

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1" pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<html>
<head>
   <meta charset="ISO-8859-1">
   <title>View Products</title>
</head>
<body>
   <h1>View Products</h1>
   <table border="1">
      <tr>
         <th>Name</th>
         <th>Price</th>
```

```
            <th>Image</th>
        </tr>
        <c:forEach var="product" items="${products}">
            <tr>
                <td>${product.name}</td>
                <td>${product.price}</td>
                <td><img style="width: 100px; height: 100px" src="${product.imageUrl}"
alt="${product.name}" /></td>
            </tr>
        </c:forEach>
    </table>
    <a href="./">Back to Home</a>
</body>
</html>
```

By: Om Shankar Mishra