SPRING BOOT JAVA APPLICATION

PROJECT :Supermarket Billing System

controller/AdminController.java

```
package com.supermarket.super_market_billing_system.controller;

import com.supermarket.super_market_billing_system.model.Bill;
import com.supermarket.super_market_billing_system.model.Product;
import com.supermarket.super_market_billing_system.service.BillService;
import com.supermarket.super_market_billing_system.service.ProductService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@Controller
@RequestMapping("/admin")
public class AdminController {
    @Autowired
    private ProductService productService;

    @Autowired
    private BillService billService;

    @GetMapping("/home")
    public String home(Model model) {
```

```
        model.addAttribute("products", productService.getAllProducts());
        return "admin_home";
    }

    @PostMapping("/add")
    public String addProduct(Product product) {
        productService.addProduct(product);
        return "redirect:/admin/home";
    }

    @GetMapping("/delete/{id}")
    public String deleteProduct(@PathVariable Long id) {
        productService.deleteProduct(id);
        return "redirect:/admin/home";
    }

    @GetMapping("/report")
    public String viewReport(Model model) {
        List<Bill> bills = billService.getAllBills();
        model.addAttribute("bills", bills);
        return "report";
    }
}
```

controller/AuthController.java

```
package com.supermarket.super_market_billing_system.controller;

import com.supermarket.super_market_billing_system.model.User;
import com.supermarket.super_market_billing_system.service.UserService;
```

```
import jakarta.servlet.http.HttpSession;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.*;
import org.springframework.ui.Model;

@Controller
public class AuthController {
    @Autowired
    private UserService userService;

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

    @PostMapping("/login")
    public String login(@RequestParam String username, @RequestParam String
password, HttpSession session) {
        User user = userService.validateUser(username, password);
        if (user != null) {
            session.setAttribute("user", user);
            return user.getRole().equals("admin") ? "redirect:/admin/home" :
"redirect:/customer/home";
        }
        return "login";
    }
}
```

controller/CustomerController.java

```
package com.supermarket.super_market_billing_system.controller;


import com.supermarket.super_market_billing_system.model.Bill;
import com.supermarket.super_market_billing_system.model.Product;
import com.supermarket.super_market_billing_system.service.BillService;
import com.supermarket.super_market_billing_system.service.ProductService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@Controller
@RequestMapping("/customer")
public class CustomerController {
    @Autowired
    private ProductService productService;

    @Autowired
    private BillService billService;

    @GetMapping("/home")
    public String home(Model model) {
        model.addAttribute("products", productService.getAllProducts());
        return "customer_home";
    }
```

```
    @PostMapping("/buy")
    public String buy(@RequestParam String customer, @RequestParam Long
productId, @RequestParam int quantity) {
        Product product = productService.getProductById(productId);
        if (product != null && product.getQuantity() >= quantity) {
            product.setQuantity(product.getQuantity() - quantity);
            productService.updateProduct(product);

            Bill bill = new Bill();
            bill.setCustomer(customer);
            bill.setProduct(product.getName());
            bill.setQuantity(quantity);
            bill.setTotal(quantity * product.getPrice());
            billService.addBill(bill);
        }
        return "redirect:/customer/home";
    }

    @GetMapping("/bill")
    public String viewBill(@RequestParam(required = false) String customer, Model
model) {
        model.addAttribute("bills", customer != null ?
billService.getBillsByCustomer(customer) : billService.getAllBills());
        return "bill";
    }
}
```

model/Bill.java

```
package com.supermarket.super_market_billing_system.model;
```

```
import jakarta.persistence.*;

@Entity
public class Bill {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String customer;
    private String product;
    private int quantity;
    private double total;

    // Getters and Setters

                public Long getId() {
                        return id;
                }
                public void setId(Long id) {
                        this.id = id;
                }
                public String getCustomer() {
                        return customer;
                }
                public void setCustomer(String customer) {
                        this.customer = customer;
                }
                public String getProduct() {
```

```
        return product;
    }
    public void setProduct(String product) {
        this.product = product;
    }
    public int getQuantity() {
        return quantity;
    }
    public void setQuantity(int quantity) {
        this.quantity = quantity;
    }
    public double getTotal() {
        return total;
    }
    public void setTotal(double total) {
        this.total = total;
    }
}
```

model/Product.java

```
package com.supermarket.super_market_billing_system.model;


import jakarta.persistence.*;

@Entity
public class Product {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
```

```
private Long id;
private String name;
private double price;
private int quantity;

// Getters and Setters
public Long getId() {
        return id;
    }
    public void setId(Long id) {
        this.id = id;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public double getPrice() {
        return price;
    }
    public void setPrice(double price) {
        this.price = price;
    }
    public int getQuantity() {
        return quantity;
    }
    public void setQuantity(int quantity) {
        this.quantity = quantity;
```

```
        }

}
```

model/User.java

```
package com.supermarket.super_market_billing_system.model;

import jakarta.persistence.*;

@Entity
public class User {
    @Id
    private String username;
    private String password;
    private String role;

    // Getters and Setters

                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;
    }
}
```

repository/BillRepository.java

```
package com.supermarket.super_market_billing_system.repository;

import com.supermarket.super_market_billing_system.model.Bill;
import org.springframework.data.jpa.repository.JpaRepository;

import java.util.List;

public interface BillRepository extends JpaRepository<Bill, Long> {
    List<Bill> findByCustomer(String customer);
}
```

repository/ProductRepository.java

```
package com.supermarket.super_market_billing_system.repository;


import com.supermarket.super_market_billing_system.model.Product;
import org.springframework.data.jpa.repository.JpaRepository;

public interface ProductRepository extends JpaRepository<Product, Long> {
```

```
}
```

repositroy/UserRepository.java

```
package com.supermarket.super_market_billing_system.repository;



import com.supermarket.super_market_billing_system.model.User;
import org.springframework.data.jpa.repository.JpaRepository;

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

service/BillService.java

```
package com.supermarket.super_market_billing_system.service;



import com.supermarket.super_market_billing_system.model.Bill;
import com.supermarket.super_market_billing_system.repository.BillRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.List;

@Service
public class BillService {
    @Autowired
    private BillRepository billRepository;
```

```
    public void addBill(Bill bill) {
        billRepository.save(bill);
    }

    public List<Bill> getBillsByCustomer(String customer) {
        return billRepository.findByCustomer(customer);
    }

    public List<Bill> getAllBills() {
        return billRepository.findAll();
    }
}
```

service/ProductService.java

```
package com.supermarket.super_market_billing_system.service;


import com.supermarket.super_market_billing_system.model.Product;
import com.supermarket.super_market_billing_system.repository.ProductRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.List;

@Service
public class ProductService {
    @Autowired
    private ProductRepository productRepository;
```

```
    public void addProduct(Product product) {
        productRepository.save(product);
    }

    public void deleteProduct(Long id) {
        productRepository.deleteById(id);
    }

    public List<Product> getAllProducts() {
        return productRepository.findAll();
    }

    public Product getProductById(Long id) {
        return productRepository.findById(id).orElse(null);
    }

    public void updateProduct(Product product) {
        productRepository.save(product);
    }
}
```

service/UserService.java

```
package com.supermarket.super_market_billing_system.service;

import com.supermarket.super_market_billing_system.model.User;
import com.supermarket.super_market_billing_system.repository.UserRepository;
import org.springframework.beans.factory.annotation.Autowired;
```

```
import org.springframework.stereotype.Service;

@Service
public class UserService {
    @Autowired
    private UserRepository userRepository;

    public User validateUser(String username, String password) {
        return userRepository.findByUsernameAndPassword(username, password);
    }
}
```

SuperMarketBillingSystemApplication.java

```
package com.supermarket.super_market_billing_system;

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

@SpringBootApplication
public class SuperMarketBillingSystemApplication {

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

}
```

templates/admin_home.html

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Admin Dashboard - Supermarket Billing System</title>
    <style>
        * {
            box-sizing: border-box;
            margin: 0;
            padding: 0;
            font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
        }

        body {
            background-color: #f5f5f5;
            min-height: 100vh;
        }

        .header {
            background-color: #3a86ff;
            color: white;
            padding: 15px 20px;
            display: flex;
            justify-content: space-between;
            align-items: center;
            box-shadow: 0 2px 5px rgba(0,0,0,0.1);
        }
```

```
.logo {
    font-size: 20px;
    font-weight: bold;
}

.nav-links {
    display: flex;
}

.nav-links a {
    color: white;
    text-decoration: none;
    margin-left: 20px;
    padding: 8px 15px;
    border-radius: 4px;
    transition: background-color 0.3s;
}

.nav-links a:hover {
    background-color: rgba(255, 255, 255, 0.2);
}

.container {
    max-width: 1200px;
    margin: 20px auto;
    padding: 0 20px;
}

.dashboard-title {
```

```
    margin: 20px 0;
    color: #333;
    font-size: 24px;
}

.card {
    background-color: white;
    border-radius: 8px;
    box-shadow: 0 2px 10px rgba(0,0,0,0.1);
    padding: 20px;
    margin-bottom: 20px;
}

.product-grid {
    display: grid;
    grid-template-columns: repeat(auto-fill, minmax(300px, 1fr));
    gap: 20px;
    margin-top: 20px;
}

.product-card {
    background-color: white;
    border-radius: 8px;
    box-shadow: 0 2px 5px rgba(0,0,0,0.1);
    overflow: hidden;
    transition: transform 0.3s, box-shadow 0.3s;
}

.product-card:hover {
```

```
    transform: translateY(-5px);
    box-shadow: 0 5px 15px rgba(0,0,0,0.1);
}

.product-content {
    padding: 15px;
}

.product-title {
    font-size: 18px;
    font-weight: 500;
    margin-bottom: 10px;
}

.product-price {
    color: #3a86ff;
    font-weight: bold;
    font-size: 18px;
    margin-bottom: 10px;
}

.product-stock {
    color: #666;
    margin-bottom: 15px;
}

.product-actions {
    display: flex;
    justify-content: flex-end;
```

```
}

.delete-btn {
    padding: 6px 12px;
    background-color: #ff3a3a;
    color: white;
    border: none;
    border-radius: 4px;
    cursor: pointer;
    transition: background-color 0.3s;
}

.delete-btn:hover {
    background-color: #e60000;
}

.add-product-form {
    display: grid;
    grid-template-columns: repeat(auto-fit, minmax(200px, 1fr));
    gap: 15px;
    margin-top: 20px;
}

.form-control {
    margin-bottom: 15px;
}

.form-control label {
    display: block;
```

```
        margin-bottom: 5px;
        color: #555;
    }

    .form-control input {
        width: 100%;
        padding: 10px;
        border: 1px solid #ddd;
        border-radius: 4px;
        font-size: 16px;
    }

    .submit-btn {
        padding: 10px 15px;
        background-color: #3a86ff;
        color: white;
        border: none;
        border-radius: 4px;
        cursor: pointer;
        transition: background-color 0.3s;
        grid-column: 1 / -1;
        width: 200px;
        justify-self: end;
    }

    .submit-btn:hover {
        background-color: #2667cc;
    }
</style>
```

```
</head>
<body>
    <div class="header">
        <div class="logo">Supermarket Admin</div>
        <div class="nav-links">
            <a th:href="@{/admin/home}">Products</a>
            <a th:href="@{/admin/report}">Report</a>
            <a th:href="@{/}">Logout</a>
        </div>
    </div>

    <div class="container">
        <h1 class="dashboard-title">Product Management</h1>

        <div class="card">
            <h2>Add New Product</h2>
            <form class="add-product-form" th:action="@{/admin/add}" method="post">
                <div class="form-control">
                    <label for="name">Product Name</label>
                    <input type="text" id="name" name="name" required>
                </div>
                <div class="form-control">
                    <label for="price">Price</label>
                    <input type="number" id="price" name="price" step="0.01" required>
                </div>
                <div class="form-control">
                    <label for="quantity">Stock Quantity</label>
                    <input type="number" id="quantity" name="quantity" required>
                </div>
```

```
            <button type="submit" class="submit-btn">Add Product</button>
        </form>
    </div>

    <h2 class="dashboard-title">Current Inventory</h2>

    <div class="product-grid">
        <div th:each="product : ${products}" class="product-card">
            <div class="product-content">
                <h3 class="product-title" th:text="${product.name}">Product Name</h3>
                <div class="product-price" th:text="'Rs.' + ${product.price}">$0.00</div>
                <div class="product-stock" th:text="'In Stock: ' + ${product.quantity}">In
Stock: 0</div>
                <div class="product-actions">
                    <a th:href="@{/admin/delete/{id}(id=${product.id})}" class="delete-
btn">Delete</a>
                </div>
            </div>
        </div>
    </div>
</div>

<script>
    // Add any JavaScript functionality here
    document.addEventListener('DOMContentLoaded', function() {
        console.log('Admin dashboard loaded');
    });
</script>
</body>
</html>
```

templates/login.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Supermarket Billing System - Login</title>
    <style>
        * {
            box-sizing: border-box;
            margin: 0;
            padding: 0;
            font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
        }

        body {
            background-color: #f5f5f5;
            display: flex;
            justify-content: center;
            align-items: center;
            height: 100vh;
            background-image: linear-gradient(135deg, #f5f7fa 0%, #c3cfe2 100%);
        }

        .login-container {
            background-color: white;
            border-radius: 10px;
            box-shadow: 0 0 20px rgba(0, 0, 0, 0.1);
```

```
    padding: 40px;
    width: 400px;
    text-align: center;
}

.login-logo {
    font-size: 24px;
    font-weight: bold;
    color: #3a86ff;
    margin-bottom: 30px;
}

.login-form input {
    width: 100%;
    padding: 12px;
    margin: 10px 0;
    border: 1px solid #ddd;
    border-radius: 5px;
    font-size: 16px;
}

.login-button {
    background-color: #3a86ff;
    color: white;
    border: none;
    padding: 12px;
    width: 100%;
    border-radius: 5px;
    cursor: pointer;
```

```
            font-size: 16px;
            margin-top: 20px;
            transition: background-color 0.3s;
        }

        .login-button:hover {
            background-color: #2667cc;
        }

        .error-message {
            color: #ff3a3a;
            margin-top: 20px;
            display: none;
        }
    </style>
</head>
<body>
    <div class="login-container">
        <div class="login-logo">Supermarket Billing System</div>
        <form class="login-form" action="/login" method="post">
            <input type="text" name="username" placeholder="Username" required>
            <input type="password" name="password" placeholder="Password"
required>
            <button type="submit" class="login-button">Login</button>
        </form>
        <p class="error-message" id="error-message">Invalid username or
password</p>
    </div>

    <script>
```

```
        // Check for login failure
        const urlParams = new URLSearchParams(window.location.search);
        if (urlParams.has('error')) {
            document.getElementById('error-message').style.display = 'block';
        }
    </script>
</body>
</html>
```

templates/customer_home.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Customer Dashboard - Supermarket Billing System</title>
    <style>
        * {
            box-sizing: border-box;
            margin: 0;
            padding: 0;
            font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
        }

        body {
            background-color: #f5f5f5;
            min-height: 100vh;
        }
```

```
.header {
    background-color: #3a86ff;
    color: white;
    padding: 15px 20px;
    display: flex;
    justify-content: space-between;
    align-items: center;
    box-shadow: 0 2px 5px rgba(0,0,0,0.1);
}

.logo {
    font-size: 20px;
    font-weight: bold;
}

.nav-links {
    display: flex;
}

.nav-links a {
    color: white;
    text-decoration: none;
    margin-left: 20px;
    padding: 8px 15px;
    border-radius: 4px;
    transition: background-color 0.3s;
}

.nav-links a:hover {
```

```
    background-color: rgba(255, 255, 255, 0.2);
}

.container {
    max-width: 1200px;
    margin: 20px auto;
    padding: 0 20px;
}

.dashboard-title {
    margin: 20px 0;
    color: #333;
    font-size: 24px;
}

.products-container {
    display: grid;
    grid-template-columns: repeat(auto-fill, minmax(300px, 1fr));
    gap: 20px;
}

.product-card {
    background-color: white;
    border-radius: 8px;
    box-shadow: 0 2px 5px rgba(0,0,0,0.1);
    overflow: hidden;
    transition: transform 0.3s, box-shadow 0.3s;
}
```

```
.product-card:hover {
    transform: translateY(-5px);
    box-shadow: 0 5px 15px rgba(0,0,0,0.1);
}

.product-content {
    padding: 20px;
}

.product-title {
    font-size: 18px;
    font-weight: 500;
    margin-bottom: 10px;
}

.product-price {
    color: #3a86ff;
    font-weight: bold;
    font-size: 18px;
    margin-bottom: 10px;
}

.product-stock {
    color: #666;
    margin-bottom: 15px;
}

.buy-form {
    margin-top: 15px;
```

```
}

.form-row {
    display: flex;
    align-items: center;
    margin-bottom: 10px;
}

.form-row label {
    margin-right: 10px;
    min-width: 80px;
}

.form-row input {
    flex: 1;
    padding: 8px;
    border: 1px solid #ddd;
    border-radius: 4px;
}

.buy-btn {
    background-color: #3a86ff;
    color: white;
    border: none;
    padding: 10px 15px;
    border-radius: 4px;
    cursor: pointer;
    width: 100%;
    font-size: 16px;
```

```
    transition: background-color 0.3s;
    margin-top: 10px;
}

.buy-btn:hover {
    background-color: #2667cc;
}

.no-stock {
    color: #ff3a3a;
    font-style: italic;
}

.customer-name-container {
    background-color: white;
    padding: 20px;
    border-radius: 8px;
    box-shadow: 0 2px 5px rgba(0,0,0,0.1);
    margin-bottom: 20px;
}

.customer-name-input {
    display: flex;
    align-items: center;
}

.customer-name-input label {
    margin-right: 10px;
    font-weight: 500;
```

```
        }

        .customer-name-input input {
            padding: 10px;
            border: 1px solid #ddd;
            border-radius: 4px;
            flex: 1;
            max-width: 300px;
        }
    </style>
</head>
<body>
    <div class="header">
        <div class="logo">Supermarket Shopping</div>
        <div class="nav-links">
            <a href="/customer/home">Products</a>
            <a href="/customer/bill" id="view-bills-link">View My Bills</a>
            <a href="/">Logout</a>
        </div>
    </div>

    <div class="container">
        <div class="customer-name-container">
            <div class="customer-name-input">
                <label for="customer-name">Your Name:</label>
                <input type="text" id="customer-name" placeholder="Enter your name">
            </div>
        </div>
```

```
        <h1 class="dashboard-title">Available Products</h1>


        <div class="products-container">
            <div th:each="product : ${products}" class="product-card">
                <div class="product-content">
                    <h3 class="product-title" th:text="${product.name}">Product Name</h3>
                    <div class="product-price" th:text="'Rs.' +
${product.price}">Rs.0.00</div>
                    <div th:if="${product.quantity > 0}" class="product-stock" th:text="'In
Stock: ' + ${product.quantity}">In Stock: 0</div>
                    <div th:if="${product.quantity <= 0}" class="no-stock">Out of
Stock</div>


                    <form th:if="${product.quantity > 0}" class="buy-form"
action="/customer/buy" method="post">
                        <input type="hidden" name="productId" th:value="${product.id}">
                        <input type="hidden" name="customer" id="customer-field">


                        <div class="form-row">
                            <label for="quantity">Quantity:</label>
                            <input type="number" id="quantity" name="quantity" min="1"
th:max="${product.quantity}" value="1">
                        </div>


                        <button type="submit" class="buy-btn">Buy Now</button>
                    </form>
                </div>
            </div>
        </div>
    </div>
```

```
<script>
    document.addEventListener('DOMContentLoaded', function() {
        const customerNameInput = document.getElementById('customer-name');
        const viewBillsLink = document.getElementById('view-bills-link');
        const buyForms = document.querySelectorAll('.buy-form');

        // Check if customer name is saved in localStorage
        if (localStorage.getItem('customerName')) {
            customerNameInput.value = localStorage.getItem('customerName');
        }

        // Update localStorage when customer name changes
        customerNameInput.addEventListener('input', function() {
            localStorage.setItem('customerName', this.value);
            updateCustomerFields();
            updateViewBillsLink();
        });

        // Add customer name to all buy forms
        function updateCustomerFields() {
            const customerName = customerNameInput.value;
            buyForms.forEach(form => {
                // Fix: Use a general selector that will work for all forms
                const customerField = form.querySelector('input[name="customer"]');
                if (customerField) {
                    customerField.value = customerName;
                }
            });
        }
```

```
            // Update the view bills link to include customer name
            function updateViewBillsLink() {
                const customerName = customerNameInput.value;
                if (customerName) {
                    viewBillsLink.href =
`/customer/bill?customer=${encodeURIComponent(customerName)}`;
                } else {
                    viewBillsLink.href = '/customer/bill';
                }
            }

            // Initial update
            updateCustomerFields();
            updateViewBillsLink();

            // Form validation
            buyForms.forEach(form => {
                form.addEventListener('submit', function(e) {
                    const customerName = customerNameInput.value.trim();
                    if (!customerName) {
                        e.preventDefault();
                        alert('Please enter your name before making a purchase');
                    }
                });
            });
        });
    </script>
</body>
```

```
</html>
```

templates/bill.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Bills - Supermarket Billing System</title>
    <style>
        * {
            box-sizing: border-box;
            margin: 0;
            padding: 0;
            font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
        }

        body {
            background-color: #f5f5f5;
            min-height: 100vh;
        }

        .header {
            background-color: #3a86ff;
            color: white;
            padding: 15px 20px;
            display: flex;
            justify-content: space-between;
            align-items: center;
```

```
    box-shadow: 0 2px 5px rgba(0,0,0,0.1);
}

.logo {
    font-size: 20px;
    font-weight: bold;
}

.nav-links {
    display: flex;
}

.nav-links a {
    color: white;
    text-decoration: none;
    margin-left: 20px;
    padding: 8px 15px;
    border-radius: 4px;
    transition: background-color 0.3s;
}

.nav-links a:hover {
    background-color: rgba(255, 255, 255, 0.2);
}

.container {
    max-width: 1200px;
    margin: 20px auto;
    padding: 0 20px;
```

```
}

.dashboard-title {
    margin: 20px 0;
    color: #333;
    font-size: 24px;
}

.bills-container {
    background-color: white;
    border-radius: 8px;
    box-shadow: 0 2px 5px rgba(0,0,0,0.1);
    overflow: hidden;
}

.bill-table {
    width: 100%;
    border-collapse: collapse;
}

.bill-table th, .bill-table td {
    padding: 12px 15px;
    text-align: left;
    border-bottom: 1px solid #ddd;
}

.bill-table th {
    background-color: #f8f9fa;
    font-weight: 600;
```

```
    color: #333;
}

.bill-table tbody tr:hover {
    background-color: #f5f5f5;
}

.bill-total {
    color: #3a86ff;
    font-weight: bold;
}

.bill-date {
    color: #666;
    font-size: 14px;
}

.bill-actions {
    display: flex;
    gap: 10px;
}

.print-btn {
    padding: 6px 12px;
    background-color: #3a86ff;
    color: white;
    border: none;
    border-radius: 4px;
    cursor: pointer;
```

```
    text-decoration: none;
    font-size: 14px;
    transition: background-color 0.3s;
}

.print-btn:hover {
    background-color: #2667cc;
}

.no-bills {
    padding: 30px;
    text-align: center;
    color: #666;
    font-style: italic;
}

.bills-summary {
    margin-top: 20px;
    padding: 15px;
    background-color: white;
    border-radius: 8px;
    box-shadow: 0 2px 5px rgba(0,0,0,0.1);
}

.summary-item {
    display: flex;
    justify-content: space-between;
    padding: 10px 0;
    border-bottom: 1px solid #eee;
```

```
        }

        .summary-item:last-child {
            border-bottom: none;
            font-weight: bold;
        }

        .summary-label {
            color: #555;
        }

        .summary-value {
            color: #333;
        }
    </style>
</head>
<body>
    <div class="header">
        <div class="logo">Supermarket Bills</div>
        <div class="nav-links">
            <a href="#" id="back-link">Back</a>
            <a href="/">Logout</a>
        </div>
    </div>

    <div class="container">
        <h1 class="dashboard-title" id="bills-title">Bills</h1>

        <div class="bills-container">
```

```
        <table class="bill-table">
            <thead>
                <tr>
                    <th>ID</th>
                    <th>Customer</th>
                    <th>Product</th>
                    <th>Quantity</th>
                    <th>Total</th>
                    <th>Actions</th>
                </tr>
            </thead>
            <tbody>
                <tr th:if="${bills.empty}">
                    <td colspan="6" class="no-bills">No bills found</td>
                </tr>
                <tr th:each="bill : ${bills}">
                    <td th:text="${bill.id}">1</td>
                    <td th:text="${bill.customer}">Customer Name</td>
                    <td th:text="${bill.product}">Product Name</td>
                    <td th:text="${bill.quantity}">1</td>
                    <td class="bill-total" th:text="'Rs.' + ${bill.total}">Rs.0.00</td>
                    <td class="bill-actions">
                        <button class="print-btn" th:onclick="'printBill(' + ${bill.id} +
')'">Print</button>
                    </td>
                </tr>
            </tbody>
        </table>
    </div>
```

```
    <div class="bills-summary" id="bills-summary">
        <h3>Summary</h3>
        <div class="summary-item">
            <span class="summary-label">Total Bills:</span>
            <span class="summary-value" id="total-bills">0</span>
        </div>
        <div class="summary-item">
            <span class="summary-label">Total Products:</span>
            <span class="summary-value" id="total-products">0</span>
        </div>
        <div class="summary-item">
            <span class="summary-label">Total Amount:</span>
            <span class="summary-value" id="total-amount">Rs.0.00</span>
        </div>
    </div>
</div>

<script>
    document.addEventListener('DOMContentLoaded', function() {
        const backLink = document.getElementById('back-link');
        const billsTitle = document.getElementById('bills-title');
        const urlParams = new URLSearchParams(window.location.search);
        const customer = urlParams.get('customer');

        // Set back link based on current path
        if (window.location.pathname.includes('/admin')) {
            backLink.href = '/admin/home';
            billsTitle.textContent = 'All Bills';
```

```
        } else {
            backLink.href = '/customer/home';
            if (customer) {
                billsTitle.textContent = `Bills for ${customer}`;
            } else {
                billsTitle.textContent = 'All Bills';
            }
        }

        // Calculate summary statistics - FIXED calculation logic for total amount with Rs. prefix
        const billRows = Array.from(document.querySelectorAll('.bill-table tbody tr')).filter(row => !row.querySelector('.no-bills'));

        if (billRows.length > 0) {
            // Count the total number of bills
            const totalBills = billRows.length;

            // Sum the quantities from all bills
            const totalProducts = billRows.reduce((sum, row) => {
                // Get quantity from the 4th cell (index 3)
                const quantity = parseInt(row.cells[3].textContent) || 0;
                return sum + quantity;
            }, 0);

            // Calculate total amount from all bills - FIXED for Rs. prefix
            const totalAmount = billRows.reduce((sum, row) => {
                // Get amount from the 5th cell (index 4), removing 'Rs.' and converting to float
                const amountText = row.cells[4].textContent;
```

```
                const amount = parseFloat(amountText.replace('Rs.', '').trim()) || 0;
                return sum + amount;
            }, 0);

            // Update the summary display
            document.getElementById('total-bills').textContent = totalBills;
            document.getElementById('total-products').textContent = totalProducts;
            document.getElementById('total-amount').textContent = 'Rs.' +
totalAmount.toFixed(2);

            // Make sure the summary is visible
            document.getElementById('bills-summary').style.display = 'block';
        } else {
            // If no bills, you can choose to hide or show zeros
            document.getElementById('bills-summary').style.display = 'block';
        }
    });

    function printBill(billId) {
        // Find the bill row
        const billRow = Array.from(document.querySelectorAll('.bill-table tbody
tr')).find(row => {
            return row.cells[0].textContent == billId;
        });

        if (!billRow) return;

        // Extract bill data
        const id = billRow.cells[0].textContent;
        const customer = billRow.cells[1].textContent;
```

```
const product = billRow.cells[2].textContent;
const quantity = billRow.cells[3].textContent;
const total = billRow.cells[4].textContent;

// Create a printable bill
const printWindow = window.open('', '', 'height=500,width=800');
printWindow.document.write(`
    <html>
    <head>
        <title>Bill #${id}</title>
        <style>
            body {
                font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
                padding: 20px;
                max-width: 800px;
                margin: 0 auto;
            }
            .bill-header {
                text-align: center;
                margin-bottom: 30px;
            }
            .bill-title {
                font-size: 24px;
                margin-bottom: 10px;
            }
            .bill-subtitle {
                color: #666;
                font-size: 16px;
            }
```

```
.bill-info {
    margin: 20px 0;
    padding: 15px;
    border: 1px solid #ddd;
    border-radius: 5px;
}
.bill-row {
    display: flex;
    justify-content: space-between;
    padding: 10px 0;
    border-bottom: 1px solid #eee;
}
.bill-row:last-child {
    border-bottom: none;
}
.bill-label {
    font-weight: bold;
    color: #555;
}
.bill-value {
    color: #333;
}
.bill-total-row {
    font-size: 18px;
    font-weight: bold;
    margin-top: 20px;
    padding-top: 10px;
    border-top: 2px solid #ddd;
}
```

```
        .bill-footer {
            margin-top: 40px;
            text-align: center;
            color: #666;
            font-style: italic;
        }
        .print-btn {
            padding: 10px 20px;
            background-color: #3a86ff;
            color: white;
            border: none;
            border-radius: 4px;
            cursor: pointer;
            display: block;
            margin: 30px auto;
            font-size: 16px;
        }
        @media print {
            .print-btn {
                display: none;
            }
        }
    </style>
</head>
<body>
    <div class="bill-header">
        <h1 class="bill-title">Supermarket Billing System</h1>
        <p class="bill-subtitle">Bill Receipt #${id}</p>
        <p class="bill-subtitle">Date: ${new Date().toLocaleDateString()}</p>
```

```
        </div>

        <div class="bill-info">
            <div class="bill-row">
                <span class="bill-label">Customer:</span>
                <span class="bill-value">${customer}</span>
            </div>
            <div class="bill-row">
                <span class="bill-label">Product:</span>
                <span class="bill-value">${product}</span>
            </div>
            <div class="bill-row">
                <span class="bill-label">Quantity:</span>
                <span class="bill-value">${quantity}</span>
            </div>
            <div class="bill-row bill-total-row">
                <span class="bill-label">Total Amount:</span>
                <span class="bill-value">${total}</span>
            </div>
        </div>

        <div class="bill-footer">
            <p>Thank you for shopping with us!</p>
            <p>This is a computer-generated receipt and does not require a signature.</p>
        </div>

        <button class="print-btn" onclick="window.print(); return false;">Print Receipt</button>
    </body>
```

```
            </html>
        `);
        printWindow.document.close();
        printWindow.focus();
    }
</script>
</body>
</html>
```

templates/report.html

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Sales Report - Supermarket Billing System</title>
    <style>
        * {
            box-sizing: border-box;
            margin: 0;
            padding: 0;
            font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
        }

        body {
            background-color: #f5f5f5;
            min-height: 100vh;
        }
```

```
.header {
    background-color: #3a86ff;
    color: white;
    padding: 15px 20px;
    display: flex;
    justify-content: space-between;
    align-items: center;
    box-shadow: 0 2px 5px rgba(0,0,0,0.1);
}

.logo {
    font-size: 20px;
    font-weight: bold;
}

.nav-links {
    display: flex;
}

.nav-links a {
    color: white;
    text-decoration: none;
    margin-left: 20px;
    padding: 8px 15px;
    border-radius: 4px;
    transition: background-color 0.3s;
}

.nav-links a:hover {
```

```
    background-color: rgba(255, 255, 255, 0.2);
}

.container {
    max-width: 1200px;
    margin: 20px auto;
    padding: 0 20px;
}

.dashboard-title {
    margin: 20px 0;
    color: #333;
    font-size: 24px;
}

.report-container {
    display: grid;
    grid-template-columns: 1fr 1fr;
    gap: 20px;
}

.card {
    background-color: white;
    border-radius: 8px;
    box-shadow: 0 2px 5px rgba(0,0,0,0.1);
    padding: 20px;
    height: 100%;
}
```

```
.card-title {
    color: #333;
    font-size: 18px;
    margin-bottom: 15px;
    padding-bottom: 10px;
    border-bottom: 1px solid #eee;
}

.stats-grid {
    display: grid;
    grid-template-columns: repeat(2, 1fr);
    gap: 15px;
    margin-bottom: 20px;
}

.stat-card {
    background-color: #f8f9fa;
    padding: 15px;
    border-radius: 6px;
    text-align: center;
}

.stat-value {
    font-size: 24px;
    font-weight: bold;
    color: #3a86ff;
    margin-bottom: 5px;
}
```

```
.stat-label {
    color: #666;
    font-size: 14px;
}

.table-container {
    overflow-x: auto;
}

.data-table {
    width: 100%;
    border-collapse: collapse;
}

.data-table th, .data-table td {
    padding: 12px 15px;
    text-align: left;
    border-bottom: 1px solid #ddd;
}

.data-table th {
    background-color: #f8f9fa;
    font-weight: 600;
    color: #333;
}

.data-table tbody tr:hover {
    background-color: #f5f5f5;
}
```

```
.chart-container {
    height: 300px;
    margin-top: 20px;
}

.filter-container {
    margin-bottom: 20px;
    padding: 15px;
    background-color: white;
    border-radius: 8px;
    box-shadow: 0 2px 5px rgba(0,0,0,0.1);
    display: flex;
    gap: 15px;
    align-items: center;
    flex-wrap: wrap;
}

.filter-item {
    display: flex;
    align-items: center;
    gap: 10px;
}

.filter-label {
    font-weight: 500;
    color: #555;
}
```

```
.filter-select {
    padding: 8px 12px;
    border: 1px solid #ddd;
    border-radius: 4px;
    color: #333;
    background-color: white;
}

.filter-button {
    padding: 8px 15px;
    background-color: #3a86ff;
    color: white;
    border: none;
    border-radius: 4px;
    cursor: pointer;
    margin-left: auto;
    transition: background-color 0.3s;
}

.filter-button:hover {
    background-color: #2667cc;
}

.print-button {
    background-color: #4CAF50;
}

.print-button:hover {
    background-color: #3e8e41;
```

```
        }

        @media (max-width: 992px) {
            .report-container {
                grid-template-columns: 1fr;
            }
        }

        @media print {
            .header, .filter-container, .nav-links, .filter-button {
                display: none;
            }

            body {
                background-color: white;
            }

            .container {
                max-width: 100%;
                padding: 0;
            }

            .card {
                box-shadow: none;
                border: 1px solid #ddd;
            }
        }
    </style>
</head>
```

```
<body>
    <div class="header">
        <div class="logo">Supermarket Sales Report</div>
        <div class="nav-links">
            <a th:href="@{/admin/home}">Products</a>
            <a th:href="@{/}">Logout</a>
        </div>
    </div>

    <div class="container">
        <h1 class="dashboard-title">Sales Report</h1>

        <div class="filter-container">
            <div class="filter-item">
                <span class="filter-label">Customer:</span>
                <select id="customer-filter" class="filter-select">
                    <option value="">All Customers</option>
                    <!-- Will be populated dynamically -->
                </select>
            </div>
            <div class="filter-item">
                <span class="filter-label">Product:</span>
                <select id="product-filter" class="filter-select">
                    <option value="">All Products</option>
                    <!-- Will be populated dynamically -->
                </select>
            </div>
            <button id="filter-btn" class="filter-button">Apply Filters</button>
            <button id="print-btn" class="filter-button print-button">Print Report</button>
```

```
</div>

<div class="report-container">
    <div class="card">
        <h2 class="card-title">Sales Summary</h2>
        <div class="stats-grid">
            <div class="stat-card">
                <div class="stat-value" id="total-sales">0</div>
                <div class="stat-label">Total Sales</div>
            </div>
            <div class="stat-card">
                <div class="stat-value" id="total-revenue">Rs.0</div>
                <div class="stat-label">Total Revenue</div>
            </div>
            <div class="stat-card">
                <div class="stat-value" id="total-customers">0</div>
                <div class="stat-label">Total Customers</div>
            </div>
            <div class="stat-card">
                <div class="stat-value" id="avg-sale">Rs.0</div>
                <div class="stat-label">Average Sale</div>
            </div>
        </div>

        <div class="chart-container">
            <canvas id="sales-chart"></canvas>
        </div>
    </div>
```

```
<div class="card">
    <h2 class="card-title">Top Products</h2>
    <div class="table-container">
        <table class="data-table" id="products-table">
            <thead>
                <tr>
                    <th>Product</th>
                    <th>Quantity Sold</th>
                    <th>Revenue</th>
                </tr>
            </thead>
            <tbody id="products-body">
                <!-- Will be populated dynamically -->
            </tbody>
        </table>
    </div>

    <h2 class="card-title" style="margin-top: 30px;">Top Customers</h2>
    <div class="table-container">
        <table class="data-table" id="customers-table">
            <thead>
                <tr>
                    <th>Customer</th>
                    <th>Purchases</th>
                    <th>Total Spent</th>
                </tr>
            </thead>
            <tbody id="customers-body">
                <!-- Will be populated dynamically -->
```

```
                    </tbody>
                </table>
            </div>
        </div>
    </div>

    <script
src="https://cdnjs.cloudflare.com/ajax/libs/Chart.js/3.7.0/chart.min.js"></script>
    <script th:inline="javascript">
        document.addEventListener('DOMContentLoaded', function() {
            // Get bills data from the model
            const bills = /*[[${bills}]]*/ [];

            // Process the bills data
            function processData(data) {
                // Sales summary
                const totalSales = data.length;
                const totalRevenue = data.reduce((sum, bill) => sum + bill.total, 0);
                const customers = [...new Set(data.map(bill => bill.customer))];
                const totalCustomers = customers.length;
                const avgSale = totalSales > 0 ? totalRevenue / totalSales : 0;

                document.getElementById('total-sales').textContent = totalSales;
                document.getElementById('total-revenue').textContent = 'Rs.' +
totalRevenue.toFixed(2);
                document.getElementById('total-customers').textContent = totalCustomers;
                document.getElementById('avg-sale').textContent = 'Rs.' +
avgSale.toFixed(2);
```

```
            // Top products
            const productsSummary = {};
            data.forEach(bill => {
                if (!productsSummary[bill.product]) {
                    productsSummary[bill.product] = { quantity: 0, revenue: 0 };
                }
                productsSummary[bill.product].quantity += bill.quantity;
                productsSummary[bill.product].revenue += bill.total;
            });

            const productsArray = Object.entries(productsSummary).map(([product,
data]) => ({
                product,
                quantity: data.quantity,
                revenue: data.revenue
            }));

            productsArray.sort((a, b) => b.revenue - a.revenue);

            const productsBody = document.getElementById('products-body');
            productsBody.innerHTML = '';
            productsArray.forEach(item => {
                const row = document.createElement('tr');
                row.innerHTML = `
                    <td>${item.product}</td>
                    <td>${item.quantity}</td>
                    <td>$${item.revenue.toFixed(2)}</td>
                `;
                productsBody.appendChild(row);
```

```
        });

        // Top customers
        const customersSummary = {};
        data.forEach(bill => {
            if (!customersSummary[bill.customer]) {
                customersSummary[bill.customer] = { purchases: 0, spent: 0 };
            }
            customersSummary[bill.customer].purchases += 1;
            customersSummary[bill.customer].spent += bill.total;
        });

        const customersArray =
Object.entries(customersSummary).map(([customer, data]) => ({
            customer,
            purchases: data.purchases,
            spent: data.spent
        }));

        customersArray.sort((a, b) => b.spent - a.spent);

        const customersBody = document.getElementById('customers-body');
        customersBody.innerHTML = '';
        customersArray.forEach(item => {
            const row = document.createElement('tr');
            row.innerHTML = `
                <td>${item.customer}</td>
                <td>${item.purchases}</td>
                <td>$${item.spent.toFixed(2)}</td>
```

```
    `;
    customersBody.appendChild(row);
});

// Chart
renderChart(productsArray);

// Populate filters
const customerFilter = document.getElementById('customer-filter');
const productFilter = document.getElementById('product-filter');

// Clear existing options
customerFilter.innerHTML = '<option value="">All Customers</option>';
productFilter.innerHTML = '<option value="">All Products</option>';

// Add customer options
customers.forEach(customer => {
    const option = document.createElement('option');
    option.value = customer;
    option.textContent = customer;
    customerFilter.appendChild(option);
});

// Add product options
Object.keys(productsSummary).forEach(product => {
    const option = document.createElement('option');
    option.value = product;
    option.textContent = product;
    productFilter.appendChild(option);
```

```
    });
}

function renderChart(productsData) {
    const ctx = document.getElementById('sales-chart').getContext('2d');

    // Destroy existing chart if it exists
    if (window.salesChart) {
        window.salesChart.destroy();
    }

    // Get top 5 products for the chart
    const topProducts = productsData.slice(0, 5);

    window.salesChart = new Chart(ctx, {
        type: 'bar',
        data: {
            labels: topProducts.map(item => item.product),
            datasets: [{
                label: 'Revenue ($)',
                data: topProducts.map(item => item.revenue),
                backgroundColor: 'rgba(58, 134, 255, 0.7)',
                borderColor: 'rgba(58, 134, 255, 1)',
                borderWidth: 1
            }]
        },
        options: {
            responsive: true,
            maintainAspectRatio: false,
```

```
            scales: {
                y: {
                    beginAtZero: true,
                    title: {
                        display: true,
                        text: 'Revenue ($)'
                    }
                }
            },
            plugins: {
                title: {
                    display: true,
                    text: 'Top 5 Products by Revenue'
                },
                legend: {
                    display: false
                }
            }
        }
    });
}

// Initial data processing
processData(bills);

// Filter button click handler
document.getElementById('filter-btn').addEventListener('click', function() {
    const customerFilter = document.getElementById('customer-filter').value;
    const productFilter = document.getElementById('product-filter').value;
```

```
            let filteredData = bills;

            if (customerFilter) {
                filteredData = filteredData.filter(bill => bill.customer === customerFilter);
            }

            if (productFilter) {
                filteredData = filteredData.filter(bill => bill.product === productFilter);
            }

            processData(filteredData);
        });

        // Print button click handler
        document.getElementById('print-btn').addEventListener('click', function() {
            window.print();
        });
    });
  </script>
</body>
</html>
```

application.properties

```
spring.application.name=super-market-billing-system
# DataSource Configuration
spring.datasource.url=jdbc:mysql://localhost:3306/supermarketdb
spring.datasource.username=root
spring.datasource.password=ram@123
```

```
# JPA/Hibernate Configuration
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL8Dialect

# Thymeleaf
spring.thymeleaf.cache=false

# Server port (optional)
# server.port=8080
```