|  |
| --- |
| E-commerce Website for Sporty Shoes  **(Source Code)** |

Contents

[1. GitHub Link. 1](#_Toc87560184)

[2. Folder Structure: 2](#_Toc87560185)

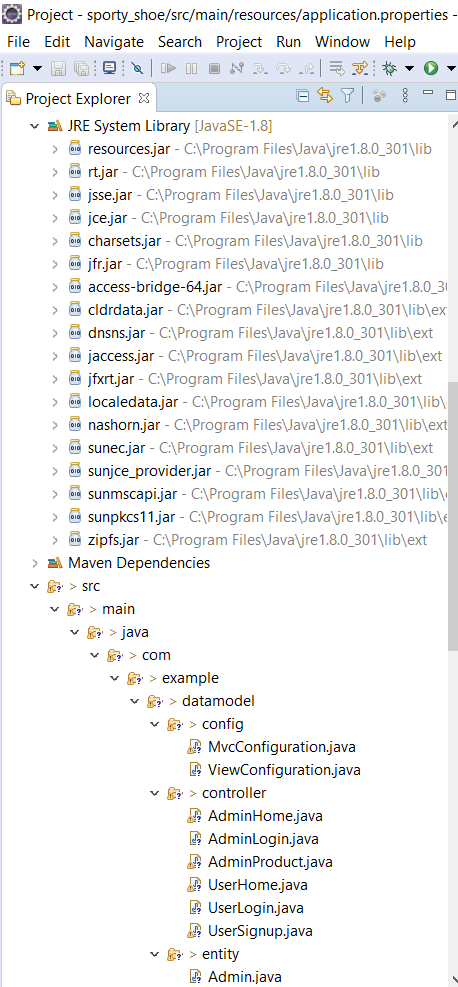
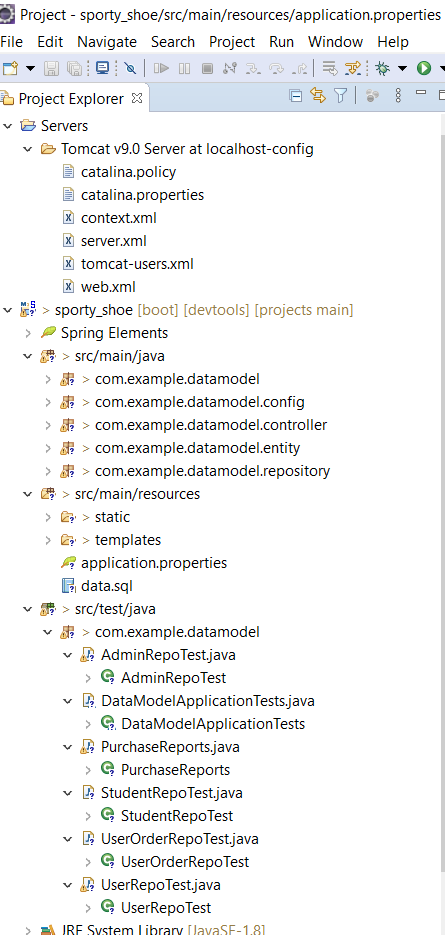
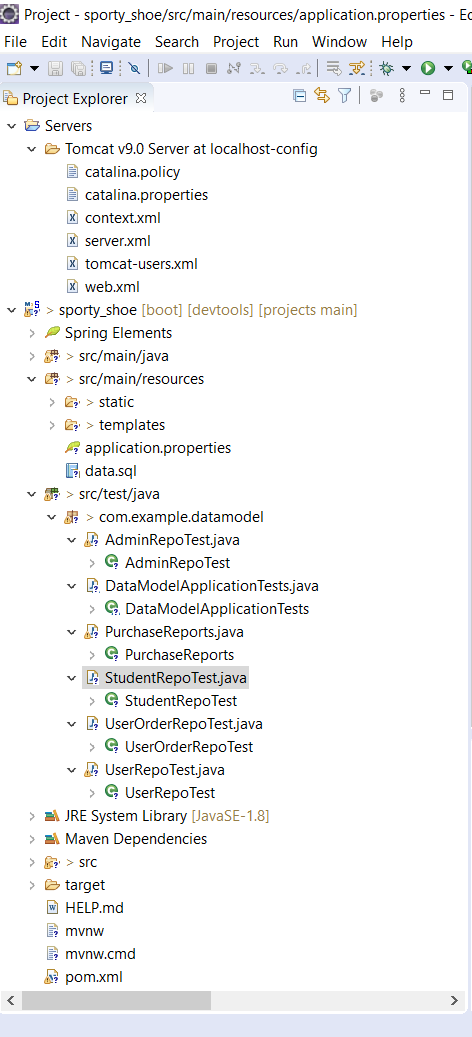
[3. Project Code: 7](#_Toc87560186)

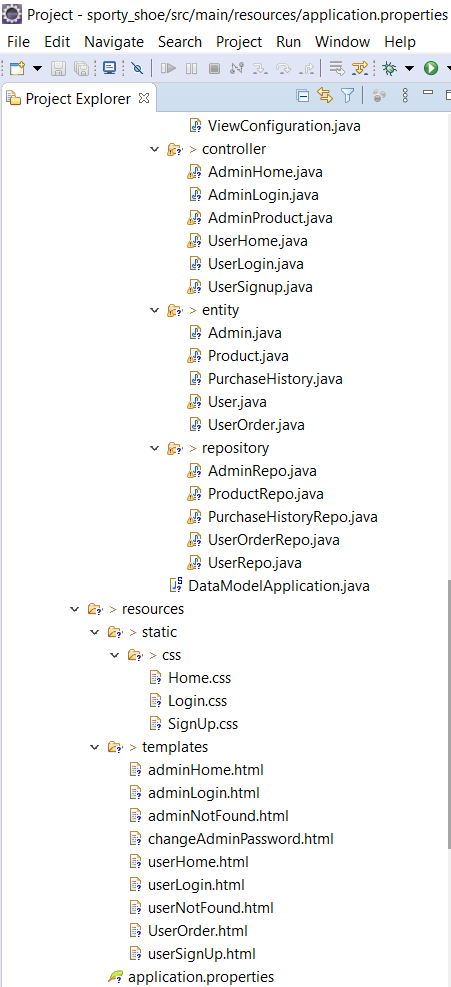
# GitHub Link.

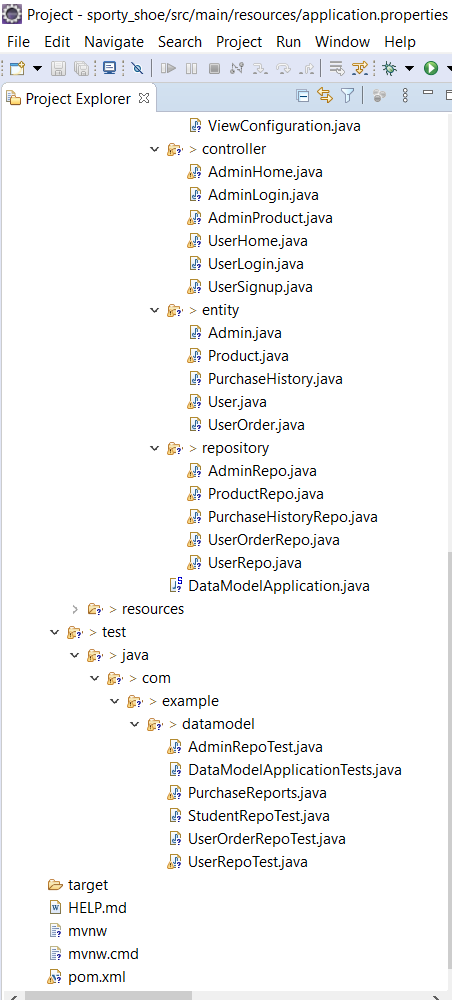
https://github.com/skibrahimsk/ThirdPhaseProject.git

# Folder Structure:

|  |
| --- |
| **Folder Structure** |
|  |







# Project Code:

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>2.5.6</version>

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

</parent>

<groupId>com.sportsshoes</groupId>

<artifactId>sporty\_shoe</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>sporty\_shoe</name>

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

<properties>

<java.version>1.8</java.version>

</properties>

<dependencies>

<dependency>

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

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

</dependency>

<dependency>

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

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

</dependency>

<dependency>

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

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

</dependency>

<dependency>

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

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

</dependency>

<dependency>

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

<artifactId>spring-boot-devtools</artifactId>

<scope>runtime</scope>

<optional>true</optional>

</dependency>

<dependency>

<groupId>mysql</groupId>

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

<scope>runtime</scope>

</dependency>

<dependency>

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

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

<scope>test</scope>

</dependency>

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

<dependency>

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

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

<version>2.5.6</version>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

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

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

</plugin>

</plugins>

</build>

</project>

1. **DataModelApplication.java**

package com.example.datamodel;

import com.example.datamodel.repository.\*;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

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

import org.springframework.boot.CommandLineRunner;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import java.util.Date;

@SpringBootApplication

public class DataModelApplication implements CommandLineRunner {

@Autowired

private UserOrderRepo userOrderRepo;

@Autowired

private ProductRepo productRepo;

@Autowired

private AdminRepo adminRepo;

@Autowired

private UserRepo userRepo;

@Autowired

private PurchaseHistoryRepo purchaseHistoryRepo;

private Logger logger = LoggerFactory.getLogger(this.getClass());

public static void main(String[] args) {

SpringApplication.run(DataModelApplication.class, args);

}

@Override

public void run(String... args) throws Exception {

//purchaseHistoryRepo.

// logger.info("list result -> {}",adminRepo.filterByDateAndCategory());

// logger.info("gotcha -> {}", userOrderRepo.insert());

}

}

1. **MvcConfiguration.java**.

package com.example.datamodel.config;

import org.springframework.context.annotation.Configuration;

import org.springframework.web.servlet.config.annotation.EnableWebMvc;

@Configuration

public class MvcConfiguration {

}

1. **ViewConfiguration.java**

package com.example.datamodel.config;

import org.springframework.context.ApplicationContext;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.web.servlet.ViewResolver;

import org.thymeleaf.spring5.SpringTemplateEngine;

import org.thymeleaf.spring5.templateresolver.SpringResourceTemplateResolver;

import org.thymeleaf.spring5.view.ThymeleafViewResolver;

import org.thymeleaf.templateresolver.ITemplateResolver;

@Configuration

public class ViewConfiguration {

private final ApplicationContext applicationContext;

public ViewConfiguration(ApplicationContext applicationContext) {

this.applicationContext = applicationContext;

}

@Bean

public ITemplateResolver templateResolver() {

SpringResourceTemplateResolver templateResolver = new SpringResourceTemplateResolver();

templateResolver.setApplicationContext(applicationContext);

templateResolver.setPrefix("classpath:/templates/");

templateResolver.setSuffix(".html");

templateResolver.setTemplateMode("HTML");

templateResolver.setCharacterEncoding("UTF-8");

return templateResolver;

}

@Bean

public SpringTemplateEngine templateEngine() {

SpringTemplateEngine templateEngine = new SpringTemplateEngine();

templateEngine.setEnableSpringELCompiler(true);

templateEngine.setTemplateResolver(templateResolver());

return templateEngine;

}

@Bean

public ViewResolver viewResolver() {

ThymeleafViewResolver viewResolver = new ThymeleafViewResolver();

viewResolver.setTemplateEngine(templateEngine());

return viewResolver;

}

}

1. **AdminHome.java**

package com.example.datamodel.controller;

import com.example.datamodel.entity.\*;

import com.example.datamodel.repository.AdminRepo;

import com.example.datamodel.repository.ProductRepo;

import com.example.datamodel.repository.PurchaseHistoryRepo;

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.RequestParam;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.List;

import java.util.Map;

@Controller

public class AdminHome {

@Autowired

private AdminRepo adminRepo;

@Autowired

private ProductRepo productRepo;

@Autowired

private PurchaseHistoryRepo purchaseHistoryRepo;

@RequestMapping("/change")

public String change() {

return "changeAdminPassword";

}

@PostMapping("/changePassword")

public String changePassword(@RequestParam Map<String,String>maps) {

String email = maps.get("email");

String oldPassword = maps.get("oldPassword");

String newPassword = maps.get("newPassword");

Admin admin = adminRepo.changePasswordByEmail(email,oldPassword,newPassword);

if(admin == null) {

return "adminNotFound";

} else {

return "adminLogin";

}

}

@RequestMapping("/getUsers")

public String getUsers(Model model) {

List<User> users = adminRepo.getAllUsersForAdmin();

model.addAttribute("users",users);

return "adminHome";

}

@RequestMapping("/searchUser")

public String searchUserByName(Model model, @RequestParam Map<String, String> maps) {

String name = maps.get("name");

List<User> users = adminRepo.searchUser(name);

model.addAttribute("users",users);

return "adminHome";

}

@RequestMapping("/purchaseHistory")

public String purchaseReports(Model model) {

List<PurchaseHistory> purchaseHistories = purchaseHistoryRepo.getPurchaseHistory();

model.addAttribute("purchaseHistory",purchaseHistories);

return "adminHome";

}

}

1. **AdminLogoin.java**

package com.example.datamodel.controller;

import com.example.datamodel.repository.AdminRepo;

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

import org.springframework.stereotype.Controller;

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

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

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

import java.util.Map;

@Controller

public class AdminLogin {

@Autowired

private AdminRepo adminRepo;

@RequestMapping("/admin")

public String admin() {

return "adminLogin";

}

@PostMapping("/adminLogin")

public String adminLogin(@RequestParam Map<String, String> maps) {

String email = maps.get("email");

String password = maps.get("password");

if(adminRepo.verifyAdmin(email,password) ==null) {

return "adminNotFound";

}

else {

return "adminHome";

}

}

}

1. **AdminProduct.java**

package com.example.datamodel.controller;

import com.example.datamodel.repository.AdminRepo;

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

import org.springframework.stereotype.Controller;

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

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

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

import java.util.Map;

@Controller

public class AdminLogin {

@Autowired

private AdminRepo adminRepo;

@RequestMapping("/admin")

public String admin() {

return "adminLogin";

}

@PostMapping("/adminLogin")

public String adminLogin(@RequestParam Map<String, String> maps) {

String email = maps.get("email");

String password = maps.get("password");

if(adminRepo.verifyAdmin(email,password) ==null) {

return "adminNotFound";

}

else {

return "adminHome";

}

}

}

1. **UserHome.java**

package com.example.datamodel.controller;

import com.example.datamodel.entity.Product;

import com.example.datamodel.entity.User;

import com.example.datamodel.entity.UserOrder;

import com.example.datamodel.repository.ProductRepo;

import com.example.datamodel.repository.UserOrderRepo;

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

import org.springframework.boot.web.servlet.server.Session;

import org.springframework.stereotype.Controller;

import org.springframework.ui.Model;

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

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

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import javax.servlet.http.HttpSession;

import java.util.List;

import java.util.Map;

@Controller

public class UserHome {

@Autowired

private ProductRepo productRepo;

@Autowired

private UserOrderRepo userOrderRepo;

@RequestMapping("/add")

public String addProduct(Model model, HttpServletRequest request, HttpServletResponse response) {

List<Product> products = productRepo.getAllProducts();

model.addAttribute("products",products);

return "userHome";

}

@RequestMapping("/addProducts")

public String addProductToOrder(Model model, @RequestParam("id") Long id, HttpServletRequest request, HttpServletResponse response) {

User user = (User)request.getSession().getAttribute("user");

Long userId = user.getId();

UserOrder userOrder = productRepo.addProductToUserOrder(id,userId);

Long orderId = userOrder.getId();

Long price = userOrderRepo.addTotalCostToOrder(orderId);

model.addAttribute("order",userOrder);

model.addAttribute("price",price);

return "UserOrder";

}

}

1. **UserLogin.java**

package com.example.datamodel.controller;

import com.example.datamodel.entity.User;

import com.example.datamodel.repository.UserRepo;

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

import org.springframework.stereotype.Controller;

import org.springframework.ui.Model;

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

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import javax.servlet.http.HttpSession;

import java.util.Map;

@Controller

@SessionAttributes({"user"})

public class UserLogin {

@Autowired

private UserRepo userRepo;

@Autowired

HttpSession session;

@RequestMapping("/")

public String loginPage() {

return "userLogin";

}

@PostMapping("/userLogin")

public String loginUser(Model model, @RequestParam Map<String, String> maps, HttpServletRequest request, HttpServletResponse response) throws Exception {

String email = maps.get("email");

String password = maps.get("password");

User user = userRepo.verifyUser(email,password);

request.getSession().setAttribute("user",user);

if(userRepo.verifyUser(email,password) ==null) {

return "userNotFound";

}

else {

return "userHome";

}

}

}

1. **UserSignup.java**

package com.example.datamodel.controller;

import com.example.datamodel.entity.User;

import com.example.datamodel.repository.UserRepo;

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

import org.springframework.stereotype.Controller;

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

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

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

@Controller

public class UserSignup {

@Autowired

private UserRepo userRepo;

@RequestMapping("/signup")

public String signUp() {

return "userSignUp";

}

@RequestMapping("/UserSignUp")

public String createUser(@ModelAttribute User user) {

User u = userRepo.addNewUser(user);

return "userLogin";

}

}

1. **Admin.java**

**package** com.example.datamodel.entity;

**import** javax.persistence.\*;

/\*

\* STORES ADMIN DETAILS TO DATABASE

\* \*/

@Entity

@NamedQuery(name = "get\_all\_admins", query = "select a from Admin a")

**public** **class** Admin {

@Id

@GeneratedValue(strategy = GenerationType.***IDENTITY***)

@Column(name = "id", updatable = **false**, nullable = **false**)

**private** Long id;

**private** String password;

**private** String adminName;

**private** String email;

**public** Admin(String password, String adminName, String email) {

**this**.password = password;

**this**.adminName = adminName;

**this**.email = email;

}

/\*

\* Empty constructor to make JPA happy

\* so any kind of data can be acceptable\*/

**public** Admin() {

}

/\*

\* getter & setter for fetching and updating the data\*/

**public** Long getId() {

**return** id;

}

**public** **void** setId(Long id) {

**this**.id = id;

}

**public** String getPassword() {

**return** password;

}

**public** **void** setPassword(String password) {

**this**.password = password;

}

**public** String getAdminName() {

**return** adminName;

}

**public** **void** setAdminName(String adminName) {

**this**.adminName = adminName;

}

**public** String getEmail() {

**return** email;

}

**public** **void** setEmail(String email) {

**this**.email = email;

}

@Override

**public** String toString() {

**return** "Admin{" +

"id=" + id +

", password='" + password + '\'' +

", adminName='" + adminName + '\'' +

", email='" + email + '\'' +

'}';

}

}

1. **Product.java**

package com.example.datamodel.entity;

import javax.persistence.\*;

import java.util.ArrayList;

import java.util.Date;

import java.util.List;

@Entity

@NamedQueries(value = {

@NamedQuery(name = "get\_all\_products", query = "Select p FROM Product p")

})

public class Product {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

@Column(name = "id", updatable = false, nullable = false)

private Long id;

private String category;

private String productName;

private Long cost;

@ManyToMany(fetch = FetchType.EAGER)

@JoinTable(name="order\_product",

joinColumns = @JoinColumn(name="product\_id"),

inverseJoinColumns = @JoinColumn(name = "order\_id")

)

private List<UserOrder> userOrders = new ArrayList<>();

public Product() {

}

public Product(String productName,String category, Long cost, List<UserOrder> userOrders) {

this.productName = productName;

this.category = category;

this.cost = cost;

this.userOrders = userOrders;

}

public Product(String productName,String category, Long cost) {

this.productName = productName;

this.category = category;

this.cost = cost;

}

public String getCategory() {

return category;

}

public void setCategory(String category) {

this.category = category;

}

public Long getCost() {

return cost;

}

public void setCost(Long cost) {

this.cost = cost;

}

public Product(String productName) {

this.productName = productName;

}

public Long getId() {

return id;

}

public void setId(Long id) {

this.id = id;

}

public String getProductName() {

return productName;

}

public void setProductName(String productName) {

this.productName = productName;

}

public List<UserOrder> getUserOrders() {

return userOrders;

}

public void addUserOrders(UserOrder userOrder) {

this.userOrders.add(userOrder);

}

public void delUserOrders(UserOrder userOrder) {

this.userOrders.remove(userOrder);

}

@Override

public String toString() {

return "Product{" +

"id=" + id +

", productName='" + productName + '\'' +

'}';

}

}

PurchaseHistory.java

package com.example.datamodel.entity;

import javax.persistence.\*;

import java.util.Date;

@Entity

public class PurchaseHistory {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

@Column(name = "id", updatable = false, nullable = false)

private Long id;

private Long productId;

private Long orderId;

private Long cost;

private Long total;

private Long UserId;

private String category;

private String productName;

private Date date;

public PurchaseHistory() {

}

public PurchaseHistory(Long productId, Long orderId, Long cost, Long total, Long userId, String category, String productName, Date date) {

this.productId = productId;

this.orderId = orderId;

this.cost = cost;

this.total = total;

UserId = userId;

this.category = category;

this.productName = productName;

this.date = date;

}

public Long getId() {

return id;

}

public void setId(Long id) {

this.id = id;

}

public Long getProductId() {

return productId;

}

public void setProductId(Long productId) {

this.productId = productId;

}

public Long getOrderId() {

return orderId;

}

public void setOrderId(Long orderId) {

this.orderId = orderId;

}

public Long getCost() {

return cost;

}

public void setCost(Long cost) {

this.cost = cost;

}

public Long getTotal() {

return total;

}

public void setTotal(Long total) {

this.total = total;

}

public Long getUserId() {

return UserId;

}

public void setUserId(Long userId) {

UserId = userId;

}

public String getCategory() {

return category;

}

public void setCategory(String category) {

this.category = category;

}

public String getProductName() {

return productName;

}

public void setProductName(String productName) {

this.productName = productName;

}

public Date getDate() {

return date;

}

public void setDate(Date date) {

this.date = date;

}

@Override

public String toString() {

return "PurchaseHistory{" +

"id=" + id +

", productId=" + productId +

", orderId=" + orderId +

", cost=" + cost +

", total=" + total +

", UserId=" + UserId +

", category='" + category + '\'' +

", productName='" + productName + '\'' +

", date=" + date +

'}';

}

}

1. **User.java**

package com.example.datamodel.entity;

import org.springframework.web.context.annotation.SessionScope;

import javax.persistence.\*;

@Entity

@NamedQueries(

value = {

@NamedQuery(name="get\_all\_users", query = "select u from User u")

}

)

public class User {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

@Column(name = "id", updatable = false, nullable = false)

private Long id;

private String userName;

private String email;

private String name;

private String password;

@OneToOne(cascade = {CascadeType.ALL})

private UserOrder userOrder;

public UserOrder getUserOrder() {

return userOrder;

}

public void setUserOrder(UserOrder userOrder) {

this.userOrder = userOrder;

}

public User() {

}

public User(String userName, String email, String name, String password) {

this.userName = userName;

this.email = email;

this.name = name;

this.password = password;

}

public Long getId() {

return id;

}

public void setId(Long id) {

this.id = id;

}

public String getUserName() {

return userName;

}

public void setUserName(String userName) {

this.userName = userName;

}

public String getEmail() {

return email;

}

public void setEmail(String email) {

this.email = email;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getPassword() {

return password;

}

public void setPassword(String password) {

this.password = password;

}

@Override

public String toString() {

return "User{" +

"id=" + id +

", userName='" + userName + '\'' +

", email='" + email + '\'' +

", name='" + name + '\'' +

", password='" + password + '\'' +

'}';

}

}

1. **UserOrder.java**

package com.example.datamodel.entity;

import javax.persistence.\*;

import java.util.ArrayList;

import java.util.Date;

import java.util.List;

@Entity

@NamedQueries(value = {

@NamedQuery(name = "get\_all\_orders", query = "Select u FROM UserOrder u")

})

public class UserOrder {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

@Column(name = "id", updatable = false, nullable = false)

private Long id;

private Long total;

public User getUser() {

return user;

}

public void setUser(User user) {

this.user = user;

}

@OneToOne(cascade = {CascadeType.ALL})

private User user;

private Date date;

@ManyToMany(mappedBy = "userOrders",fetch = FetchType.EAGER)

private List<Product> products = new ArrayList<>();

public UserOrder() {

}

public UserOrder(Long total, Date date) {

this.total = total;

this.date = date;

}

public UserOrder(Date date) {

this.date = date;

this.id= user.getId();

}

public UserOrder(Date date,Long id) {

this.date = date;

}

public UserOrder(Long total, Date date, List<Product> products) {

this.total = total;

this.date = date;

this.products = products;

}

public UserOrder(Long total) {

this.total = total;

}

public Date getDate() {

return date;

}

public void setDate(Date date) {

this.date = date;

}

public Long getId() {

return id;

}

public void setId(Long id) {

this.id = id;

}

public Long getTotal() {

return total;

}

public void setTotal(Long total) {

this.total = total;

}

public List<Product> getProducts() {

return products;

}

public void addProducts(Product product) {

this.products.add(product);

}

public void delProducts(Product product) {

this.products.remove(product);

}

@Override

public String toString() {

return "UserOrder{" +

"id=" + id +

", total=" + total +

", date=" + date +

", products=" + products +

'}';

}

}

1. **AdminRepo.java**

package com.example.datamodel.repository;

import com.example.datamodel.entity.Admin;

import com.example.datamodel.entity.User;

import com.example.datamodel.entity.UserOrder;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

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

import org.springframework.stereotype.Repository;

import javax.persistence.EntityManager;

import javax.persistence.TypedQuery;

import javax.transaction.Transactional;

import java.util.ArrayList;

import java.util.List;

@Repository

@Transactional

public class AdminRepo {

@Autowired

EntityManager em;

private Logger log = LoggerFactory.getLogger(this.getClass());

@Autowired

private UserOrderRepo userOrderRepo;

@Autowired

private ProductRepo productRepo;

@Autowired

private UserRepo userRepo;

/\*

\* Add a new admin if not exists already

\* Using persist of Entity Manager

\* if exist used merge to update the admin data

\* \*/

public Admin saveAdmin(Admin admin) {

if(admin.getId() == null) {

em.persist(admin);

} else {

em.merge(admin);

}

return admin;

}

/\*

\* find a admin using id

\* used Entity manager's find method

\* which returns the admin object\*/

public Admin findById(Long adminId) {

Admin admin =em.find(Admin.class, adminId);

log.info("admin retrieved -> {}",admin);

return admin;

}

/\*

\* update admin password

\* first find user using findById

\* then verify with oldPassword

\* then update newPassword\*/

public String changePassword(Long adminId, String oldPassword, String newPassword) {

Admin admin = findById(adminId);

if(admin.getPassword() == oldPassword) {

admin.setPassword(newPassword);

return "Password change successfull";

} else {

return "Password mismatch";

}

}

public void filterByDateAndCategory() {

List<UserOrder> orders = userOrderRepo.getAllOrders();

List<List> ordersWithProducts = new ArrayList<>();

for(Long i = 0L; i < orders.size(); i++) {

UserOrder userOrder = em.find(UserOrder.class,i);

ordersWithProducts.add(userOrder.getProducts());

}

}

public List<Admin> getAdmins() {

TypedQuery<Admin> query = em.createNamedQuery("get\_all\_admins",Admin.class);

List<Admin> result = query.getResultList();

return result;

}

public Admin verifyAdmin(String email, String password) {

Admin admin = null;

List<Admin> admins = getAdmins();

for(int i = 0; i<admins.size();i++) {

admin = admins.get(i);

if(admin.getEmail().equals(email)) {

if( admin.getPassword().equals(password) ) {

break;

}

}

}

log.info("admin -> {}",admin);

return admin;

}

public Admin changePasswordByEmail(String email, String oldPassword, String newPassword) {

Admin admin = verifyAdmin(email,oldPassword);

if(admin == null) {

return null;

} else {

Long id = admin.getId();

changePassword(id,oldPassword,newPassword);

return admin;

}

}

public List<User> getAllUsersForAdmin() {

List<User> users = userRepo.getUsers();

return users;

}

public List<User> searchUser(String name) {

List users = userRepo.searchUserByName(name);

return users;

}

}

1. **ProductRepo.java**

package com.example.datamodel.repository;

import com.example.datamodel.entity.Product;

import com.example.datamodel.entity.User;

import com.example.datamodel.entity.UserOrder;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

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

import org.springframework.stereotype.Repository;

import javax.persistence.EntityManager;

import javax.persistence.Query;

import javax.persistence.TypedQuery;

import javax.transaction.Transactional;

import java.util.List;

@Repository

@Transactional

public class ProductRepo {

@Autowired

EntityManager em;

@Autowired

private UserRepo userRepo;

@Autowired

private UserOrderRepo userOrderRepo;

private Logger log = LoggerFactory.getLogger(this.getClass());

public Product searchProduct(Long id) {

Product product = em.find(Product.class, id);

return product;

}

public List<Product> getAllProducts() {

TypedQuery<Product> query = em.createNamedQuery("get\_all\_products", Product.class);

List<Product> result = query.getResultList();

return result;

}

public String delProduct(Long id) {

Product product = searchProduct(id);

em.remove(product);

return "deleted Successfully";

}

public Product addProduct(Product product) {

em.persist(product);

return product;

}

public Product updateProduct(Long id, Product product) {

Product p1 = searchProduct(id);

if (p1.getId() == null) {

em.persist(product);

} else {

em.merge(product);

}

return product;

}

public UserOrder addProductToOrder(Long orderId, Long productId) {

UserOrder userOrder = em.find(UserOrder.class, orderId);

Product product = em.find(Product.class, productId);

userOrder.addProducts(product);

product.addUserOrders(userOrder);

em.persist(userOrder);

return userOrder;

}

public UserOrder addProductToUserOrder(Long ProductId, Long userId) {

UserOrder userOrder = userOrderRepo.createNewOrder(userId);

Long orderId = userOrder.getId();

UserOrder userOrder1 = addProductToOrder(orderId, ProductId);

return userOrder1;

}

public List<Object[]> testPurchaseRepoQuery() {

Query q = em.createQuery("select uo,p from UserOrder uo JOIN uo.products p order by p.category desc ");

List<Object[]> resultList = q.getResultList();

return resultList;

}

}

1. **PurchaseHistory.java**

package com.example.datamodel.repository;

import com.example.datamodel.entity.Product;

import com.example.datamodel.entity.PurchaseHistory;

import com.example.datamodel.entity.UserOrder;

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

import org.springframework.stereotype.Repository;

import javax.persistence.\*;

import javax.transaction.Transactional;

import java.util.\*;

@Repository

@Transactional

public class PurchaseHistoryRepo {

@Autowired

EntityManager em;

@Autowired

private ProductRepo productRepo;

public void save(PurchaseHistory purchaseHistory) {

em.persist(purchaseHistory);

}

public void addDataToPurchaseHistory() {

em.createNativeQuery("TRUNCATE TABLE purchase\_history;").executeUpdate();

List<Object[]> resultlist = productRepo.testPurchaseRepoQuery();

for(Object[] result:resultlist) {

Product product = (Product) result[1];

UserOrder userOrder = (UserOrder) result[0];

Long productId= product.getId();

Long orderId = userOrder.getId();

Long cost = product.getCost();

Long total = userOrder.getTotal();

Long userId=userOrder.getUser().getId();

String category = product.getCategory();

String productName = product.getProductName();

Date date = userOrder.getDate();

PurchaseHistory purchaseHistory = new PurchaseHistory(productId,orderId,cost,total,userId,category,productName,date);

save(purchaseHistory);

}

}

public List<PurchaseHistory> getPurchaseHistory() {

addDataToPurchaseHistory();

TypedQuery<PurchaseHistory> query = (TypedQuery<PurchaseHistory>) em.createNativeQuery("SELECT \* FROM datamodel.purchase\_history order by category and date desc;",PurchaseHistory.class);

List<PurchaseHistory> result = query.getResultList();

return result;

}

}

1. **UserOrderRepo.java**

package com.example.datamodel.repository;

import com.example.datamodel.entity.Product;

import com.example.datamodel.entity.User;

import com.example.datamodel.entity.UserOrder;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

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

import org.springframework.context.annotation.Lazy;

import org.springframework.stereotype.Repository;

import javax.persistence.EntityManager;

import javax.persistence.TypedQuery;

import javax.transaction.Transactional;

import java.util.Date;

import java.util.List;

@Repository

@Transactional

public class UserOrderRepo {

@Autowired

EntityManager em;

private UserRepo userRepo;

@Autowired

public UserOrderRepo(@Lazy UserRepo userRepo) {

this.userRepo = userRepo;

}

private Logger log = LoggerFactory.getLogger(this.getClass());

public UserOrder insert() {

UserOrder userOrder = new UserOrder(700L);

Product product = new Product("rahu");

em.persist(userOrder);

em.persist(product);

userOrder.addProducts(product);

product.addUserOrders(userOrder);

em.persist(userOrder);

return userOrder;

}

public Long totalCost(Long orderId) {

UserOrder userOrder = em.find(UserOrder.class,orderId);

List<Product> products = userOrder.getProducts();

Long sum = 0L;

for(int i = 0; i < products.size(); i++) {

sum += products.get(i).getCost();

}

return sum;

}

public Long addTotalCostToOrder(Long orderId) {

Long sum = totalCost(orderId);

UserOrder userOrder = em.find(UserOrder.class,orderId);

userOrder.setTotal(sum);

em.persist(userOrder);

em.flush();

return userOrder.getTotal();

}

public List<UserOrder> getAllOrders() {

TypedQuery<UserOrder> query = em.createNamedQuery("get\_all\_orders",UserOrder.class);

List<UserOrder> result = query.getResultList();

return result;

}

public UserOrder createNewOrder(Long id) {

User user = em.find(User.class,id);

UserOrder userOrder = user.getUserOrder();

if(userOrder==null) {

userOrder = new UserOrder(new Date(),id);

em.persist(userOrder);

userRepo.addUserOrderToUser(id,userOrder);

userOrder.setUser(em.find(User.class,id));

em.persist(userOrder);

return userOrder;

} else {

return userOrder;

}

}

public UserOrder getOrder(Long orderId) {

UserOrder userOrder = em.find(UserOrder.class,orderId);

return userOrder;

}

}

1. **UserRepo.java**

package com.example.datamodel.repository;

import com.example.datamodel.entity.User;

import com.example.datamodel.entity.UserOrder;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

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

import org.springframework.stereotype.Repository;

import javax.persistence.EntityManager;

import javax.persistence.Query;

import javax.persistence.TypedQuery;

import javax.transaction.Transactional;

import java.util.List;

import java.util.Queue;

@Repository

@Transactional

public class UserRepo {

@Autowired

EntityManager em;

private Logger log = LoggerFactory.getLogger(this.getClass());

/\*

\* find user by user id

\* using entity manager's find method

\* return type is user object\*/

public User findUserById(Long userId) {

User user = em.find(User.class, userId);

log.info("user object returned -> {}", user);

return user;

}

public User userUpdate(Long userId, String oldPassword, String newPassword) {

User user = em.find(User.class, userId);

if (user.getPassword() == oldPassword) {

user.setPassword(newPassword);

if (user.getPassword() == newPassword) {

return user;

} else {

return null;

}

} else {

return null;

}

}

public User addNewUser(User user) {

em.persist(user);

return user;

}

public List<User> getUsers() {

TypedQuery<User> query = em.createNamedQuery("get\_all\_users",User.class);

List<User> result = query.getResultList();

return result;

}

public User verifyUser(String email, String password) {

User user = null;

List<User> users = getUsers();

User newuser = null;

for (int i = 0; i < users.size(); i++) {

user = users.get(i);

if (user.getEmail().equals(email)) {

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

newuser = user;

break;

}

}

}

log.info("user -> {}", user);

return newuser;

}

public List<User> searchUserByName(String name) {

Query query = em.createNativeQuery("Select \* from user where name = :name",User.class);

query.setParameter("name",name);

List result = query.getResultList();

return result;

}

public void addUserOrderToUser(Long id, UserOrder userOrder) {

User user = em.find(User.class,id);

user.setUserOrder(userOrder);

em.merge(user);

}

public UserOrder verifyUserToOrder(Long userId) {

User user = em.find(User.class,userId);

return user.getUserOrder();

}

}

1. **Home.css**

@charset *"ISO-8859-1"*;

**ul** {

list-style-type: *none*;

margin: *0*;

padding: *0*;

width: *100%*;

top: *0*;

overflow: *hidden*;

position: *-webkit-sticky*; /\* Safari \*/

position: *sticky*;

background-color: *#333*;

}

**li** {

float: *left*;

}

**li** **a** {

display: *block*;

color: *white*;

text-align: *center*;

padding: *14px* *16px*;

text-decoration: *none*;

}

**li***:last-child* {

border-bottom: *none*;

}

**li** **a***:hover* {

background-color: *#111*;

color: *white*;

}

*.active* {

background-color: *#04AA6D*;

color: *white*;

}

\* {

padding: *0*;

margin: *0*;

box-sizing: *border-box*;

}

**body** {

background: *#fefefe*;

font-family: *sans-serif*;

}

*.container* {

width: *90%*;

margin: *50px* *auto*;

}

*.heading* {

text-align: *center*;

font-size: *30px*;

margin-bottom: *50px*;

}

*.row*{

display: *flex*;

flex-direction: *row*;

justify-content: *space-around*;

}

*.card* {

width: *20%*;

background: *#fff*;

border: *1px* *solid* *#ccc*;

margin-bottom: *50px*;

margin-left: *5%*;

border-radius: *20px*;

transition: *0.3s*;

}

*.card-header* {

text-align: *center*;

padding: *50px* *10px*;

background: *linear-gradient(to* *right,* *#ff416c,* *#2bbefd)*;

color: *#fff*;

border-radius: *20px*;

}

*.card-body* {

padding: *30px* *20px*;

align-content: *center*;

text-align: *center*;

font-size: *18px*;

}

*.card-body* *.btn* {

display: *block*;

color: *#fff*;

text-align: *center*;

background: *linear-gradient(to* *right,* *#045159,* *#24f3f3)*;

margin-top: *30px*;

text-decoration: *none*;

padding:*10px* *5px*;

}

*.card:hover* {

transform: *scale(1.05)*;

box-shadow: *0* *0* *40px* *-10px* *rgba(0,* *0,* *0,* *0.75)*;

}

@media screen and (max-width:1000px) {

*.card* {

width: *40%*;

}

}

@media screen and (max-width:620px) {

*.container* {

width: *100%*;

}

*.heading* {

padding: *20px*;

font-size: *20px*;

}

*.card* {

width: *80%*;

}

}

*.link* {

display: *flex*;

text-decoration: *none*;

outline: *none*;

border: *none*;

cursor: *pointer*;

width: *100%*;

height: *60px*;

border-radius: *30px*;

font-weight: *700*;

font-family: *"Lato",* *sans-serif*;

color: *#fff*;

align-items: *center*;

justify-content: *center*;

text-align: *center*;

background-color: *#4a8f83*;

box-shadow: *3px* *3px* *8px* *#b1b1b1,* *-3px* *-3px* *8px* *#fff*;

font-size: *18px*;

}

*.link:hover* {

transform: *scale(1.02)*;

box-shadow: *0* *0* *40px* *-10px* *rgba(0,* *0,* *0,* *0.75)*;

}

*.ticket* {

width: *600px*;

height: *400px*;

margin-left:*30%*;

margin-top:*10%*;

justify-content: *center*;

padding: *60px* *35px* *35px* *35px*;

border-radius: *40px*;

background-color: *#ecf0f3*;

box-shadow: *13px* *13px* *20px* *#cbced1,*

*-13px* *-13px* *20px* *#fff*;

}

*.row* {

display:*flex*;

flex-direction: *row*;

justify-content: *space-between*;

border-radius: *30px*;

margin-left: *5px*;

margin-right: *5px*;

}

*.ticket:hover* {

transform: *scale(1.02)*;

box-shadow: *0* *0* *40px* *-10px* *rgba(0,* *0,* *0,* *0.75)*;

}

*.row:hover* {

transform: *scale(1.02)*;

box-shadow: *0* *0* *40px* *-10px* *rgba(0,* *0,* *0,* *0.75)*;

}

**a***:hover* {

transform: *scale(1.02)*;

outline: *none*;

text-underline: *none*;

outline-color: *#04AA6D*;

box-shadow: *0* *0* *40px* *-10px* *rgba(0,* *0,* *0,* *0.75)*;

}

**a** {

display: *flex*;

text-decoration: *none*;

outline: *none*;

border: *none*;

cursor: *pointer*;

width: *100%*;

height: *60px*;

border-radius: *30px*;

font-weight: *700*;

font-family: *"Lato",* *sans-serif*;

color: *#fff*;

align-items: *center*;

justify-content: *center*;

text-align: *center*;

background-color: *#4a8f83*;

box-shadow: *3px* *3px* *8px* *#b1b1b1,* *-3px* *-3px* *8px* *#fff*;

}

**h1** {

display: *flex*;

text-decoration: *none*;

outline: *none*;

border: *none*;

cursor: *pointer*;

width: *100%*;

height: *60px*;

border-radius: *30px*;

font-weight: *700*;

font-family: *"Lato",* *sans-serif*;

color: *#fff*;

align-items: *center*;

justify-content: *center*;

text-align: *center*;

background-color: *#4a8f83*;

box-shadow: *3px* *3px* *8px* *#b1b1b1,* *-3px* *-3px* *8px* *#fff*;

font-size: *18px*;

}

**h1***:hover* {

transform: *scale(1.02)*;

box-shadow: *0* *0* *40px* *-10px* *rgba(0,* *0,* *0,* *0.75)*;

}

*.loginBox* {

width: *400px*;

height: *700px*;

padding: *60px* *35px* *35px* *35px*;

border-radius: *40px*;

background-color: *#ecf0f3*;

box-shadow: *13px* *13px* *20px* *#cbced1,*

*-13px* *-13px* *20px* *#fff*;

}

*.logo* {

background: *url(https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcQN59jZY3e-DGR0qDhobXi5vhnlVUv1-qw8aQ&usqp=CAU)*;

width: *100px*;

height: *100px*;

border-radius: *50%*;

margin: *0* *auto*;

box-shadow: *0px* *0px* *2px* *#5f5f5f,*

*0px* *0px* *0px* *5px* *#ecf0f3,*

*8px* *8px* *15px* *#a7aaaf,*

*-8px* *-8px* *15px* *#fff*;

}

*.title* {

text-align: *center*;

font-size: *28px*;

padding-top: *24px*;

letter-spacing: *0.5px*;

}

*.fields* {

width: *100%*;

padding: *75px* *5px* *5px* *5px*;

}

*.fields* **input** {

border: *none*;

outline: *none*;

background: *none*;

font-size: *18px*;

text-align: *center*;

color: *#555*;

padding: *20px* *10px* *20px* *5px*;

}

*.username***,**

*.password* {

margin-bottom: *30px*;

border-radius: *25px*;

text-decoration-color: *#04AA6D*;

color: *#2bbefd*;

background: *linear-gradient(to* *right,* *#ff416c,* *#2bbefd)*;

box-shadow: *3px* *3px* *8px* *#b1b1b1,* *-3px* *-3px* *8px* *#fff*;

}

*::placeholder* { /\* Chrome, Firefox, Opera, Safari 10.1+ \*/

color: *white*;

text-align: *center*;

opacity: *1*; /\* Firefox \*/

}

*.fields* **svg** {

height: *22px*;

margin: *0* *10px* *-3px* *25px*;

}

*.signin-button* {

outline: *none*;

border: *none*;

cursor: *pointer*;

width: *100%*;

height: *60px*;

border-radius: *30px*;

font-weight: *700*;

font-family: *"Lato",* *sans-serif*;

color: *#fff*;

text-align: *center*;

background: *linear-gradient(to* *right,* *#ff416c,* *#2bbefd)*;

box-shadow: *3px* *3px* *8px* *#b1b1b1,* *-3px* *-3px* *8px* *#fff*;

}

**a** {

display: *flex*;

text-decoration: *none*;

outline: *none*;

border: *none*;

cursor: *pointer*;

width: *100%*;

height: *60px*;

border-radius: *30px*;

font-weight: *700*;

font-family: *"Lato",* *sans-serif*;

color: *#fff*;

align-items: *center*;

justify-content: *center*;

text-align: *center*;

background-color: *gray*;

box-shadow: *3px* *3px* *8px* *#b1b1b1,* *-3px* *-3px* *8px* *#fff*;

}

1. **Login.css**

@charset *"ISO-8859-1"*;

\* {

box-sizing: *border-box*;

}

**body** {

margin: *0*;

height: *100vh*;

width: *100vw*;

overflow: *hidden*;

font-family: *"Lato",* *sans-serif*;

font-weight: *700*;

display: *flex*;

align-items: *center*;

justify-content: *center*;

color: *#555*;

background-color: *#ecf0f3*;

}

*.loginBox* {

width: *400px*;

height: *700px*;

padding: *60px* *35px* *35px* *35px*;

border-radius: *40px*;

background-color: *#ecf0f3*;

box-shadow: *13px* *13px* *20px* *#cbced1,*

*-13px* *-13px* *20px* *#fff*;

}

*.logo* {

background: *url(https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcQN59jZY3e-DGR0qDhobXi5vhnlVUv1-qw8aQ&usqp=CAU)*;

width: *100px*;

height: *100px*;

border-radius: *50%*;

margin: *0* *auto*;

box-shadow: *0px* *0px* *2px* *#5f5f5f,*

*0px* *0px* *0px* *5px* *#ecf0f3,*

*8px* *8px* *15px* *#a7aaaf,*

*-8px* *-8px* *15px* *#fff*;

}

*.title* {

text-align: *center*;

font-size: *28px*;

padding-top: *24px*;

letter-spacing: *0.5px*;

}

*.fields* {

width: *100%*;

padding: *75px* *5px* *5px* *5px*;

}

*.fields* **input** {

border: *none*;

outline: *none*;

background: *none*;

font-size: *18px*;

color: *#555*;

padding: *20px* *10px* *20px* *5px*;

}

*.username***,**

*.password* {

margin-bottom: *30px*;

border-radius: *25px*;

box-shadow: *inset* *8px* *8px* *8px* *#cbced1,* *inset*

*-8px* *-8px* *8px* *#fff*;

}

*.fields* **svg** {

height: *22px*;

margin: *0* *10px* *-3px* *25px*;

}

*.signin-button* {

outline: *none*;

border: *none*;

cursor: *pointer*;

width: *100%*;

height: *60px*;

border-radius: *30px*;

font-weight: *700*;

font-family: *"Lato",* *sans-serif*;

color: *#fff*;

text-align: *center*;

background-color: *#FD8469*;

box-shadow: *3px* *3px* *8px* *#b1b1b1,* *-3px* *-3px* *8px* *#fff*;

}

**a** {

display: *flex*;

text-decoration: *none*;

outline: *none*;

border: *none*;

cursor: *pointer*;

width: *100%*;

height: *60px*;

border-radius: *30px*;

font-weight: *700*;

font-family: *"Lato",* *sans-serif*;

color: *#fff*;

align-items: *center*;

justify-content: *center*;

text-align: *center*;

background-color: *gray*;

box-shadow: *3px* *3px* *8px* *#b1b1b1,* *-3px* *-3px* *8px* *#fff*;

}

1. **SignUp.css**

@charset *"ISO-8859-1"*;

\* {

box-sizing: *border-box*;

}

**body** {

margin: *0*;

height: *100vh*;

width: *100vw*;

overflow: *hidden*;

font-family: *"Lato",* *sans-serif*;

font-weight: *700*;

display: *flex*;

align-items: *center*;

justify-content: *center*;

color: *#555*;

background-color: *#ecf0f3*;

}

*.loginBox* {

width: *400px*;

height: *900px*;

padding: *60px* *35px* *35px* *35px*;

border-radius: *40px*;

background-color: *#ecf0f3*;

box-shadow: *13px* *13px* *20px* *#cbced1,*

*-13px* *-13px* *20px* *#fff*;

}

*.logo* {

background: *url(https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcQN59jZY3e-DGR0qDhobXi5vhnlVUv1-qw8aQ&usqp=CAU)*;

width: *100px*;

height: *100px*;

border-radius: *50%*;

margin: *0* *auto*;

box-shadow: *0px* *0px* *2px* *#5f5f5f,*

*0px* *0px* *0px* *5px* *#ecf0f3,*

*8px* *8px* *15px* *#a7aaaf,*

*-8px* *-8px* *15px* *#fff*;

}

*.title* {

text-align: *center*;

font-size: *28px*;

padding-top: *24px*;

letter-spacing: *0.5px*;

}

*.fields* {

width: *100%*;

padding: *75px* *5px* *5px* *5px*;

}

*.fields* **input** {

border: *none*;

outline: *none*;

background: *none*;

font-size: *18px*;

color: *#555*;

padding: *20px* *10px* *20px* *5px*;

}

*.username***,**

*.password* {

margin-bottom: *30px*;

border-radius: *25px*;

box-shadow: *inset* *8px* *8px* *8px* *#cbced1,* *inset*

*-8px* *-8px* *8px* *#fff*;

}

*.fields* **svg** {

height: *22px*;

margin: *0* *10px* *-3px* *25px*;

}

*.signin-button* {

outline: *none*;

border: *none*;

cursor: *pointer*;

width: *100%*;

height: *60px*;

border-radius: *30px*;

font-weight: *700*;

font-family: *"Lato",* *sans-serif*;

color: *#fff*;

text-align: *center*;

background-color: *#FD8469*;

box-shadow: *3px* *3px* *8px* *#b1b1b1,* *-3px* *-3px* *8px* *#fff*;

}

**a** {

display: *flex*;

text-decoration: *none*;

outline: *none*;

border: *none*;

cursor: *pointer*;

width: *100%*;

height: *60px*;

border-radius: *30px*;

font-weight: *700*;

font-family: *"Lato",* *sans-serif*;

color: *#fff*;

align-items: *center*;

justify-content: *center*;

text-align: *center*;

background-color: *gray*;

box-shadow: *3px* *3px* *8px* *#b1b1b1,* *-3px* *-3px* *8px* *#fff*;

}

1. **adminHome.html**

<!DOCTYPE html>

<html lang=*"en"*>

<head>

<meta charset=*"UTF-8"*>

<title>Title</title>

<link rel=*"stylesheet"* type=*"text/css"* href=*"/css/Home.css"*>

<!-- Latest compiled and minified CSS -->

<link rel=*"stylesheet"* href=*"https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css"*>

<!-- jQuery library -->

<script src=*"https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"*></script>

<!-- Latest compiled JavaScript -->

<script src=*"https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/js/bootstrap.min.js"*></script>

</head>

<body>

<nav>

<ul>

<li><a class =*"active"* href=*"change"*>Change admin password</a></li>

<li><a href=*"getUsers"*>Get All the users</a></li>

<li> <a href=*"getProducts"*>Get All the Products</a></li>

<li style="float:*right*"> <a href=*"purchaseHistory"*>Get All the Purchase records</a></li>

</ul>

</nav>

<div class=*"row"*>

<div class=*"card text-center"*>

<h2 class=*"card-header"*>Search User</h2>

<form action=*"searchUser"*>

<input class=*"username"* type=*"text"* name=*"name"* placeholder=*"Enter name of user"*>

<Button class=*"signin-button"* type=*"submit"* value=*"click"*>Search</Button>

</form>

</div>

<div class=*"card text-center"*>

<h2 class=*"card-header"*>Add product</h2>

<form action=*"addProduct"* method=*"post"*>

<input class=*"username"* type=*"text"* name=*"productName"* placeholder=*"product name"*>

<input class=*"username"* type=*"text"* name=*"category"* placeholder=*"category"*>

<input class=*"username"* type=*"number"* name=*"cost"* placeholder=*"cost"*>

<Button class=*"signin-button"* type=*"submit"* value=*"click"*>Add Product</Button>

</form>

</div>

</div>

<div class=*"row"*>

<div class=*"card text-center"*>

<h2 class=*"card-header"*>Delete Product</h2>

<form action=*"delProduct"* method=*"post"*>

<input class=*"username"* type=*"number"* name=*"id"* placeholder=*"Enter Product Id"*>

<Button class=*"signin-button"* type=*"submit"* value=*"click"*>Delete Product</Button>

</form>

</div>

<div class=*"card text-center"*>

<h2 class=*"card-header"*>Update product</h2>

<form action=*"updateProduct"* method=*"post"* >

<input class=*"username"* type=*"number"* name=*"id"* placeholder=*"Product Id"*>

<input class=*"username"* type=*"text"* name=*"productName"* placeholder=*"product name"*>

<input class=*"username"* type=*"text"* name=*"category"* placeholder=*"category"*>

<input class=*"username"* type=*"number"* name=*"cost"* placeholder=*"cost"*>

<Button class=*"signin-button"* type=*"submit"* value=*"click"*>Update Product</Button>

</form>

</div>

</div>

<div class=*"container"*>

<div th:if=*"${not #lists.isEmpty(users)}"*>

<h2>user list</h2>

<table class=*"table table-stripped"*>

<tr>

<th>User\_Id</th>

<th>User\_Email</th>

<th>Name</th>

<th>Password</th>

<th>User\_Name</th>

</tr>

<tr th:each=*"user : ${users}"*>

<td th:text=*"${user.id}"*></td>

<td th:text=*"${user.email}"*></td>

<td th:text=*"${user.name}"*></td>

<td th:text=*"${user.password}"*></td>

<td th:text=*"${user.userName}"*></td>

</tr>

</table>

</div>

</div>

<div class=*"container"*>

<div th:if=*"${not #lists.isEmpty(products)}"*>

<h2>Product List</h2>

<table class=*"table table-stripped"*>

<tr>

<th>Product\_Id</th>

<th>Product\_Name</th>

<th>Category</th>

<th>Cost</th>

</tr>

<tr th:each=*"product : ${products}"*>

<td th:text=*"${product.id}"*></td>

<td th:text=*"${product.productName}"*></td>

<td th:text=*"${product.category}"*></td>

<td th:text=*"${product.cost}"*></td>

</tr>

</table>

</div>

</div>

<div class=*"container"*>

<div th:if=*"${not #lists.isEmpty(purchaseHistory)}"*>

<h2>Purchase History List</h2>

<table class=*"table table-stripped"*>

<tr>

<th>id</th>

<th>category</th>

<th>product\_id</th>

<th>order\_id</th>

<th>cost</th>

<th>total</th>

<th>user\_id</th>

<th>product\_name</th>

<th>date</th>

</tr>

<tr th:each=*"purchaseHistories : ${purchaseHistory} "* >

<td th:text=*"${purchaseHistories.id}"*></td>

<td th:text=*"${purchaseHistories.category}"*></td>

<td th:text=*"${purchaseHistories.productId}"*></td>

<td th:text=*"${purchaseHistories.orderId}"*></td>

<td th:text=*"${purchaseHistories.cost}"*></td>

<td th:text=*"${purchaseHistories.total}"*></td>

<td th:text=*"${purchaseHistories.UserId}"*></td>

<td th:text=*"${purchaseHistories.productName}"*></td>

<td th:text=*"${purchaseHistories.date}"*></td>

</tr>

</table>

</div>

</div>

</body>

</html>

1. **adminLogin.html**

<!DOCTYPE html>

<html>

<head>

<meta charset=*"ISO-8859-1"*>

<title>Insert title here</title>

<link rel=*"preconnect"* href=*"https://fonts.googleapis.com"*>

<link rel=*"preconnect"* href=*"https://fonts.gstatic.com"* crossorigin>

<link href=*"https://fonts.googleapis.com/css2?family=Lato:wght@300&display=swap"* rel=*"stylesheet"*>

<link href=*"/css/Login.css"* rel=*"stylesheet"* type=*"text/css"* >

</head>

<body>

<div class=*"loginBox"*>

<div class=*"logo"*></div>

<div class=*"title"*>Login</div>

<form action=*"adminLogin"* method=*"post"*>

<div class=*"fields"*>

<div class=*"username"*>

<svg class=*"svg-icon"* viewBox=*"0 0 20 20"*>

<path d=*"M12.075,10.812c1.358-0.853,2.242-2.507,2.242-4.037c0-2.181-1.795-4.618-4.198-4.618S5.921,4.594,5.921,6.775c0,1.53,0.884,3.185,2.242,4.037c-3.222,0.865-5.6,3.807-5.6,7.298c0,0.23,0.189,0.42,0.42,0.42h14.273c0.23,0,0.42-0.189,0.42-0.42C17.676,14.619,15.297,11.677,12.075,10.812 M6.761,6.775c0-2.162,1.773-3.778,3.358-3.778s3.359,1.616,3.359,3.778c0,2.162-1.774,3.778-3.359,3.778S6.761,8.937,6.761,6.775 M3.415,17.69c0.218-3.51,3.142-6.297,6.704-6.297c3.562,0,6.486,2.787,6.705,6.297H3.415z"*></path>

</svg>

<input class=*"user-input"* type=*"email"* placeholder=*"Enter Your Email"* name=*"email"* required>

</div>

<div class=*"password"*>

<svg class=*"svg-icon"* viewBox=*"0 0 20 20"*>

<path d=*"M17.308,7.564h-1.993c0-2.929-2.385-5.314-5.314-5.314S4.686,4.635,4.686,7.564H2.693c-0.244,0-0.443,0.2-0.443,0.443v9.3c0,0.243,0.199,0.442,0.443,0.442h14.615c0.243,0,0.442-0.199,0.442-0.442v-9.3C17.75,7.764,17.551,7.564,17.308,7.564 M10,3.136c2.442,0,4.43,1.986,4.43,4.428H5.571C5.571,5.122,7.558,3.136,10,3.136 M16.865,16.864H3.136V8.45h13.729V16.864z M10,10.664c-0.854,0-1.55,0.696-1.55,1.551c0,0.699,0.467,1.292,1.107,1.485v0.95c0,0.243,0.2,0.442,0.443,0.442s0.443-0.199,0.443-0.442V13.7c0.64-0.193,1.106-0.786,1.106-1.485C11.55,11.36,10.854,10.664,10,10.664 M10,12.878c-0.366,0-0.664-0.298-0.664-0.663c0-0.366,0.298-0.665,0.664-0.665c0.365,0,0.664,0.299,0.664,0.665C10.664,12.58,10.365,12.878,10,12.878"*></path>

</svg>

<input class=*"user-input"* type=*"password"* placeholder=*"Enter Your password"* name=*"password"* required>

</div>

<div class=*"signin-button"*>

<input class=*"user-input"* type=*"submit"* value=*"Login"*>

</div>

</div>

</form>

<a href=*"change"*>Change Password</a>

</div>

</body>

</html>

adminNotFound.html

<!DOCTYPE html>

<html lang=*"en"*>

<head>

<meta charset=*"UTF-8"*>

<title>Title</title>

</head>

<body>

Admin not found

</body>

</html>

changeAdminPassword.html

<!DOCTYPE html>

<html>

<head>

<meta charset=*"ISO-8859-1"*>

<title>Insert title here</title>

<link rel=*"preconnect"* href=*"https://fonts.googleapis.com"*>

<link rel=*"preconnect"* href=*"https://fonts.gstatic.com"* crossorigin>

<link href=*"https://fonts.googleapis.com/css2?family=Lato:wght@300&display=swap"* rel=*"stylesheet"*>

<link href=*"/css/SignUp.css"* rel=*"stylesheet"* type=*"text/css"* >

</head>

<body>

<div class=*"loginBox"*>

<div class=*"logo"*></div>

<div class=*"title"*>Change Password</div>

<form action=*"changePassword"* method=*"post"*>

<div class=*"fields"*>

<div class=*"username"*>

<svg class=*"svg-icon"* viewBox=*"0 0 20 20"*>

<path d=*"M12.075,10.812c1.358-0.853,2.242-2.507,2.242-4.037c0-2.181-1.795-4.618-4.198-4.618S5.921,4.594,5.921,6.775c0,1.53,0.884,3.185,2.242,4.037c-3.222,0.865-5.6,3.807-5.6,7.298c0,0.23,0.189,0.42,0.42,0.42h14.273c0.23,0,0.42-0.189,0.42-0.42C17.676,14.619,15.297,11.677,12.075,10.812 M6.761,6.775c0-2.162,1.773-3.778,3.358-3.778s3.359,1.616,3.359,3.778c0,2.162-1.774,3.778-3.359,3.778S6.761,8.937,6.761,6.775 M3.415,17.69c0.218-3.51,3.142-6.297,6.704-6.297c3.562,0,6.486,2.787,6.705,6.297H3.415z"*></path>

</svg>

<input class=*"user-input"* type=*"email"* placeholder=*"Enter Your Email Id"* name=*"email"* required>

</div>

<div class=*"password"*>

<svg class=*"svg-icon"* viewBox=*"0 0 20 20"*>

<path d=*"M17.308,7.564h-1.993c0-2.929-2.385-5.314-5.314-5.314S4.686,4.635,4.686,7.564H2.693c-0.244,0-0.443,0.2-0.443,0.443v9.3c0,0.243,0.199,0.442,0.443,0.442h14.615c0.243,0,0.442-0.199,0.442-0.442v-9.3C17.75,7.764,17.551,7.564,17.308,7.564 M10,3.136c2.442,0,4.43,1.986,4.43,4.428H5.571C5.571,5.122,7.558,3.136,10,3.136 M16.865,16.864H3.136V8.45h13.729V16.864z M10,10.664c-0.854,0-1.55,0.696-1.55,1.551c0,0.699,0.467,1.292,1.107,1.485v0.95c0,0.243,0.2,0.442,0.443,0.442s0.443-0.199,0.443-0.442V13.7c0.64-0.193,1.106-0.786,1.106-1.485C11.55,11.36,10.854,10.664,10,10.664 M10,12.878c-0.366,0-0.664-0.298-0.664-0.663c0-0.366,0.298-0.665,0.664-0.665c0.365,0,0.664,0.299,0.664,0.665C10.664,12.58,10.365,12.878,10,12.878"*></path>

</svg>

<input class=*"user-input"* type=*"password"* placeholder=*"Enter Your Old Password"* name=*"oldPassword"* required>

</div>

<div class=*"password"*>

<svg class=*"svg-icon"* viewBox=*"0 0 20 20"*>

<path d=*"M17.308,7.564h-1.993c0-2.929-2.385-5.314-5.314-5.314S4.686,4.635,4.686,7.564H2.693c-0.244,0-0.443,0.2-0.443,0.443v9.3c0,0.243,0.199,0.442,0.443,0.442h14.615c0.243,0,0.442-0.199,0.442-0.442v-9.3C17.75,7.764,17.551,7.564,17.308,7.564 M10,3.136c2.442,0,4.43,1.986,4.43,4.428H5.571C5.571,5.122,7.558,3.136,10,3.136 M16.865,16.864H3.136V8.45h13.729V16.864z M10,10.664c-0.854,0-1.55,0.696-1.55,1.551c0,0.699,0.467,1.292,1.107,1.485v0.95c0,0.243,0.2,0.442,0.443,0.442s0.443-0.199,0.443-0.442V13.7c0.64-0.193,1.106-0.786,1.106-1.485C11.55,11.36,10.854,10.664,10,10.664 M10,12.878c-0.366,0-0.664-0.298-0.664-0.663c0-0.366,0.298-0.665,0.664-0.665c0.365,0,0.664,0.299,0.664,0.665C10.664,12.58,10.365,12.878,10,12.878"*></path>

</svg>

<input class=*"user-input"* type=*"password"* placeholder=*"Enter Your new password"* name=*"newPassword"* required>

</div>

<div class=*"signin-button"*>

<input class=*"user-input"* type=*"submit"* value=*"Change Password"*>

</div>

</div>

</form>

<a href=*""*>Login</a>

</div>

</body>

</html>

1. **userHome.html**

<!DOCTYPE html>

<html lang=*"en"*>

<head>

<meta charset=*"UTF-8"*>

<title>User Home</title>

<link rel=*"stylesheet"* type=*"text/css"* href=*"/css/Home.css"*>

<!-- Latest compiled and minified CSS -->

<link rel=*"stylesheet"* href=*"https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css"*>

<!-- jQuery library -->

<script src=*"https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"*></script>

<!-- Latest compiled JavaScript -->

<script src=*"https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/js/bootstrap.min.js"*></script>

</head>

<body>

<nav>

<ul>

<li><a class =*"active"* href=*"add"*>Get All the Products</a></li>

<li><a th:text=*"${price}"*></a></li>

<li><a href=*"#"*>Dashboard</a></li>

<li style="float:*right*"><a class=*"active"* href=*"#"*>About</a></li>

</ul>

</nav>

<main class=*"dashboard"*>

<div class=*"container"*>

<div th:if=*"${not #lists.isEmpty(products)}"*>

<h2>Product List</h2>

<table class=*"table"*>

<thead>

<tr class=*"table-info"*>

<th scope=*"col"*>Product\_Id</th>

<th scope=*"col"*>Product\_Name</th>

<th scope=*"col"*>Category</th>

<th scope=*"col"*>Cost</th>

</tr>

</thead>

<tbody>

<tr class=*"table-info"* th:each=*"product : ${products}"*>

<form action=*"addProducts"* method=*"post"*>

<th scope=*"row"* th:text=*"${product.id}"*></th>

<td th:text=*"${product.productName}"*></td>

<td th:text=*"${product.category}"*></td>

<td th:text=*"${product.cost}"*></td>

<td>

<input type=*"hidden"* th:value=*"${product.id}"* name=*"id"*/>

<button type=*"submit"* class=*"btn btn-info btn-rounded"* name=*"action"* value=*"order"*>order</button>

</td>

</form>

</tr>

</table>

</div>

</div>

</main>

</body>

</html>

1. **userLogin.html**

<!DOCTYPE html>

<html>

<head>

<meta charset=*"ISO-8859-1"*>

<title>Insert title here</title>

<link rel=*"preconnect"* href=*"https://fonts.googleapis.com"*>

<link rel=*"preconnect"* href=*"https://fonts.gstatic.com"* crossorigin>

<link href=*"https://fonts.googleapis.com/css2?family=Lato:wght@300&display=swap"* rel=*"stylesheet"*>

<link href=*"/css/Login.css"* rel=*"stylesheet"* type=*"text/css"* >

</head>

<body>

<div class=*"loginBox"*>

<div class=*"logo"*></div>

<div class=*"title"*>Login</div>

<form action=*"userLogin"* method=*"post"*>

<div class=*"fields"*>

<div class=*"username"*>

<svg class=*"svg-icon"* viewBox=*"0 0 20 20"*>

<path d=*"M12.075,10.812c1.358-0.853,2.242-2.507,2.242-4.037c0-2.181-1.795-4.618-4.198-4.618S5.921,4.594,5.921,6.775c0,1.53,0.884,3.185,2.242,4.037c-3.222,0.865-5.6,3.807-5.6,7.298c0,0.23,0.189,0.42,0.42,0.42h14.273c0.23,0,0.42-0.189,0.42-0.42C17.676,14.619,15.297,11.677,12.075,10.812 M6.761,6.775c0-2.162,1.773-3.778,3.358-3.778s3.359,1.616,3.359,3.778c0,2.162-1.774,3.778-3.359,3.778S6.761,8.937,6.761,6.775 M3.415,17.69c0.218-3.51,3.142-6.297,6.704-6.297c3.562,0,6.486,2.787,6.705,6.297H3.415z"*></path>

</svg>

<input class=*"user-input"* type=*"email"* placeholder=*"Enter Your Email"* name=*"email"* required>

</div>

<div class=*"password"*>

<svg class=*"svg-icon"* viewBox=*"0 0 20 20"*>

<path d=*"M17.308,7.564h-1.993c0-2.929-2.385-5.314-5.314-5.314S4.686,4.635,4.686,7.564H2.693c-0.244,0-0.443,0.2-0.443,0.443v9.3c0,0.243,0.199,0.442,0.443,0.442h14.615c0.243,0,0.442-0.199,0.442-0.442v-9.3C17.75,7.764,17.551,7.564,17.308,7.564 M10,3.136c2.442,0,4.43,1.986,4.43,4.428H5.571C5.571,5.122,7.558,3.136,10,3.136 M16.865,16.864H3.136V8.45h13.729V16.864z M10,10.664c-0.854,0-1.55,0.696-1.55,1.551c0,0.699,0.467,1.292,1.107,1.485v0.95c0,0.243,0.2,0.442,0.443,0.442s0.443-0.199,0.443-0.442V13.7c0.64-0.193,1.106-0.786,1.106-1.485C11.55,11.36,10.854,10.664,10,10.664 M10,12.878c-0.366,0-0.664-0.298-0.664-0.663c0-0.366,0.298-0.665,0.664-0.665c0.365,0,0.664,0.299,0.664,0.665C10.664,12.58,10.365,12.878,10,12.878"*></path>

</svg>

<input class=*"user-input"* type=*"password"* placeholder=*"Enter Your password"* name=*"password"* required>

</div>

<div class=*"signin-button"*>

<input class=*"user-input"* type=*"submit"* value=*"Login"*>

</div>

</div>

</form>

<a href=*"signup"*>SignUp</a>

</div>

</body>

</html>

1. **userNotFound.html**

<!DOCTYPE html>

<html lang=*"en"*>

<head>

<meta charset=*"UTF-8"*>

<title>Title</title>

</head>

<body>

if new user create account first<a href=*"userSignUp.html"*></a>

Try logging in again<a href=*"userLogin.html"*></a>

</body>

</html>

1. **UserOrder.html**

<!DOCTYPE html>

<html>

<head>

<meta charset=*"ISO-8859-1"*>

<title>Sporty shoes</title>

</head>

<body>

<form action=*"UserSignUp"*>

<h1>thankyou for order product</h1>

<input type =*"submit"* value=*"relogin"*>

</form>

</body>

</html>

1. **userSignup.html**

<!DOCTYPE html>

<html>

<head>

<meta charset=*"ISO-8859-1"*>

<title>Insert title here</title>

<link rel=*"preconnect"* href=*"https://fonts.googleapis.com"*>

<link rel=*"preconnect"* href=*"https://fonts.gstatic.com"* crossorigin>

<link href=*"https://fonts.googleapis.com/css2?family=Lato:wght@300&display=swap"* rel=*"stylesheet"*>

<link href=*"/css/SignUp.css"* rel=*"stylesheet"* type=*"text/css"* >

</head>

<body>

<div class=*"loginBox"*>

<div class=*"logo"*></div>

<div class=*"title"*>SignUp</div>

<form action=*"UserSignUp"* method=*"post"*>

<div class=*"fields"*>

<div class=*"username"*>

<svg class=*"svg-icon"* viewBox=*"0 0 20 20"*>

<path d=*"M12.075,10.812c1.358-0.853,2.242-2.507,2.242-4.037c0-2.181-1.795-4.618-4.198-4.618S5.921,4.594,5.921,6.775c0,1.53,0.884,3.185,2.242,4.037c-3.222,0.865-5.6,3.807-5.6,7.298c0,0.23,0.189,0.42,0.42,0.42h14.273c0.23,0,0.42-0.189,0.42-0.42C17.676,14.619,15.297,11.677,12.075,10.812 M6.761,6.775c0-2.162,1.773-3.778,3.358-3.778s3.359,1.616,3.359,3.778c0,2.162-1.774,3.778-3.359,3.778S6.761,8.937,6.761,6.775 M3.415,17.69c0.218-3.51,3.142-6.297,6.704-6.297c3.562,0,6.486,2.787,6.705,6.297H3.415z"*></path>

</svg>

<input class=*"user-input"* type=*"text"* placeholder=*"Choose Your UserName"* name=*"userName"* required>

</div>

<div class=*"username"*>

<svg class=*"svg-icon"* viewBox=*"0 0 20 20"*>

<path d=*"M12.075,10.812c1.358-0.853,2.242-2.507,2.242-4.037c0-2.181-1.795-4.618-4.198-4.618S5.921,4.594,5.921,6.775c0,1.53,0.884,3.185,2.242,4.037c-3.222,0.865-5.6,3.807-5.6,7.298c0,0.23,0.189,0.42,0.42,0.42h14.273c0.23,0,0.42-0.189,0.42-0.42C17.676,14.619,15.297,11.677,12.075,10.812 M6.761,6.775c0-2.162,1.773-3.778,3.358-3.778s3.359,1.616,3.359,3.778c0,2.162-1.774,3.778-3.359,3.778S6.761,8.937,6.761,6.775 M3.415,17.69c0.218-3.51,3.142-6.297,6.704-6.297c3.562,0,6.486,2.787,6.705,6.297H3.415z"*></path>

</svg>

<input class=*"user-input"* type=*"email"* placeholder=*"Enter Your email"* name=*"email"* required>

</div>

<div class=*"password"*>

<svg class=*"svg-icon"* viewBox=*"0 0 20 20"*>

<path d=*"M17.308,7.564h-1.993c0-2.929-2.385-5.314-5.314-5.314S4.686,4.635,4.686,7.564H2.693c-0.244,0-0.443,0.2-0.443,0.443v9.3c0,0.243,0.199,0.442,0.443,0.442h14.615c0.243,0,0.442-0.199,0.442-0.442v-9.3C17.75,7.764,17.551,7.564,17.308,7.564 M10,3.136c2.442,0,4.43,1.986,4.43,4.428H5.571C5.571,5.122,7.558,3.136,10,3.136 M16.865,16.864H3.136V8.45h13.729V16.864z M10,10.664c-0.854,0-1.55,0.696-1.55,1.551c0,0.699,0.467,1.292,1.107,1.485v0.95c0,0.243,0.2,0.442,0.443,0.442s0.443-0.199,0.443-0.442V13.7c0.64-0.193,1.106-0.786,1.106-1.485C11.55,11.36,10.854,10.664,10,10.664 M10,12.878c-0.366,0-0.664-0.298-0.664-0.663c0-0.366,0.298-0.665,0.664-0.665c0.365,0,0.664,0.299,0.664,0.665C10.664,12.58,10.365,12.878,10,12.878"*></path>

</svg>

<input class=*"user-input"* type=*"text"* placeholder=*"Enter Your name"* name=*"name"* required>

</div>

<div class=*"password"*>

<svg class=*"svg-icon"* viewBox=*"0 0 20 20"*>

<path d=*"M17.308,7.564h-1.993c0-2.929-2.385-5.314-5.314-5.314S4.686,4.635,4.686,7.564H2.693c-0.244,0-0.443,0.2-0.443,0.443v9.3c0,0.243,0.199,0.442,0.443,0.442h14.615c0.243,0,0.442-0.199,0.442-0.442v-9.3C17.75,7.764,17.551,7.564,17.308,7.564 M10,3.136c2.442,0,4.43,1.986,4.43,4.428H5.571C5.571,5.122,7.558,3.136,10,3.136 M16.865,16.864H3.136V8.45h13.729V16.864z M10,10.664c-0.854,0-1.55,0.696-1.55,1.551c0,0.699,0.467,1.292,1.107,1.485v0.95c0,0.243,0.2,0.442,0.443,0.442s0.443-0.199,0.443-0.442V13.7c0.64-0.193,1.106-0.786,1.106-1.485C11.55,11.36,10.854,10.664,10,10.664 M10,12.878c-0.366,0-0.664-0.298-0.664-0.663c0-0.366,0.298-0.665,0.664-0.665c0.365,0,0.664,0.299,0.664,0.665C10.664,12.58,10.365,12.878,10,12.878"*></path>

</svg>

<input class=*"user-input"* type=*"password"* placeholder=*"Enter Your password"* name=*"password"* required>

</div>

<div class=*"signin-button"*>

<input class=*"user-input"* type=*"submit"* value=*"Sign Up"*>

</div>

</div>

</form>

<a href=*""*>Login</a>

</div>

</body>

</html>

1. **application.properties**

spring.jpa.defer-datasource-initialization=true

#db connection

spring.jpa.hibernate.ddl-auto=update

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

spring.datasource.url=jdbc:mysql://localhost:3306/sportyshoesdbs

spring.datasource.username=root

spring.datasource.password=Ravi@0525

#show querys

spring.jpa.show-sql=true

spring.jpa.properties.hibernate.format\_sql=true

logging.level.org.hibernate.type=trace

1. **Data.sql**

# **insert** **into** user\_order(id, total) **VALUES** (3,100);

# **insert** **into** user\_order(id, total) **VALUES** (4,500);

# **insert** **into** product(id, product\_name) **VALUES** (4,'nam');

# **insert** **into** product(id, product\_name) **values** (5,'kara');

# **insert** **into** product(id, product\_name) **values** (6,'yara');

**insert** **into** order\_product(product\_id, order\_id) **VALUES** (3,4);

**insert** **into** order\_product(product\_id, order\_id) **values** (3,4);

**insert** **into** order\_product(product\_id, order\_id) **values** (4,3);

**insert** **into** order\_product(product\_id, order\_id) **values** (4,3);

1. **AdminRepoTest.java**

package com.example.datamodel;

import com.example.datamodel.entity.UserOrder;

import com.example.datamodel.repository.AdminRepo;

import com.example.datamodel.repository.UserOrderRepo;

import org.junit.jupiter.api.Test;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

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

import org.springframework.boot.test.context.SpringBootTest;

import javax.persistence.EntityManager;

import javax.transaction.Transactional;

import java.util.ArrayList;

import java.util.List;

@SpringBootTest

public class AdminRepoTest {

private Logger log = LoggerFactory.getLogger(this.getClass());

@Autowired

EntityManager em;

@Autowired

AdminRepo adminRepo;

@Autowired

private UserOrderRepo userOrderRepo;

@Test

@Transactional

public void filterByDateAndCategoryTest() {

List<UserOrder> orders = userOrderRepo.getAllOrders();

List<List> ordersWithProducts = new ArrayList<>();

for(Long i = 1L; i <= orders.size(); i++) {

UserOrder userOrder = em.find(UserOrder.class,i);

ordersWithProducts.add(userOrder.getProducts());

}

log.info("here you go look at this -> {}",ordersWithProducts);

}

}

1. **DataModelApplicationTest.java**

package com.example.datamodel;

import org.junit.jupiter.api.Test;

import org.springframework.boot.test.context.SpringBootTest;

@SpringBootTest

class DataModelApplicationTests {

@Test

void contextLoads() {

}

}

1. **PurchaseReports.java**

package com.example.datamodel;

import com.example.datamodel.entity.Product;

import com.example.datamodel.entity.PurchaseHistory;

import com.example.datamodel.entity.UserOrder;

import com.example.datamodel.repository.PurchaseHistoryRepo;

import org.junit.jupiter.api.Test;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

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

import org.springframework.boot.test.context.SpringBootTest;

import org.springframework.test.annotation.DirtiesContext;

import javax.persistence.EntityManager;

import javax.persistence.Query;

import javax.persistence.TypedQuery;

import javax.transaction.Transactional;

import java.util.List;

@SpringBootTest

public class PurchaseReports {

@Autowired

EntityManager em;

@Autowired

private PurchaseHistoryRepo purchaseHistoryRepo;

private Logger logger = LoggerFactory.getLogger(this.getClass());

@Test

@Transactional

@DirtiesContext

void testPurchaseRepoQuery() {

Query q = em.createQuery("select p,uo from Product p JOIN p.userOrders uo order by p.category desc");

List<Object[]> resultList = q.getResultList();

UserOrder userOrder = null;

Product product = null;

for(Object[] result:resultList) {

product = (Product) result[0];

userOrder = (UserOrder) result[1];

// logger.info("product -> {}",product);

}

logger.info("userorder-> {}{}",userOrder,product.getCategory());

}

@Test

@Transactional

@DirtiesContext

void testGetPurchaseHistory() {

TypedQuery<PurchaseHistory> query = (TypedQuery<PurchaseHistory>) em.createNativeQuery("SELECT \* FROM datamodel.purchase\_history order by category and date desc;",PurchaseHistory.class);

List<PurchaseHistory> result = query.getResultList();

logger.info("results -> {}",result);

}

}

1. **StudentRepoTest.java**

package com.example.datamodel;

import com.example.datamodel.entity.Product;

import com.example.datamodel.entity.UserOrder;

import org.junit.jupiter.api.Test;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

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

import org.springframework.boot.test.context.SpringBootTest;

import javax.persistence.EntityManager;

import javax.transaction.Transactional;

@SpringBootTest

public class StudentRepoTest {

@Autowired

EntityManager em;

private Logger log = LoggerFactory.getLogger(this.getClass());

@Test

@Transactional

void retriveProductandOrders() {

Product product = em.find(Product.class,3L);

log.info("products -> {}",product);

log.info("product with order -> {}",product.getUserOrders());

UserOrder userOrder = em.find(UserOrder.class,4L);

log.info("orders -> {}",userOrder);

log.info("order.info ->{}",userOrder.getProducts());

}

}

1. **UserOrderRepoTest.java**

package com.example.datamodel;

import com.example.datamodel.entity.UserOrder;

import com.example.datamodel.repository.UserOrderRepo;

import com.example.datamodel.repository.UserRepo;

import org.junit.jupiter.api.Test;

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

import org.springframework.boot.test.context.SpringBootTest;

import org.springframework.test.annotation.DirtiesContext;

import javax.persistence.EntityManager;

import javax.transaction.Transactional;

import java.util.Date;

@SpringBootTest

public class UserOrderRepoTest {

@Autowired

EntityManager em;

@Autowired

private UserOrderRepo userOrderRepo;

@Autowired

private UserRepo userRepo;

@Test

@Transactional

@DirtiesContext

public void userOrder() {

userOrderRepo.createNewOrder(1l);

}

@Test

@Transactional

@DirtiesContext

public void userOrde() {

UserOrder userOrder = new UserOrder(new Date());

em.persist(userOrder);

}

@Test

@Transactional

@DirtiesContext

public void userOrdernull() {

userRepo.verifyUserToOrder(1L);

}

@Test

@Transactional

public void userOrderaddTo() {

userRepo.addUserOrderToUser(1L,new UserOrder(new Date(),1L));

}

}

1. **UserRepoTest.java**

package com.example.datamodel;

import com.example.datamodel.entity.User;

import com.example.datamodel.repository.UserRepo;

import org.junit.jupiter.api.Test;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

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

import org.springframework.boot.test.context.SpringBootTest;

import org.springframework.test.annotation.DirtiesContext;

import javax.persistence.EntityManager;

import javax.persistence.Query;

import javax.persistence.TypedQuery;

import javax.transaction.Transactional;

import java.util.List;

@SpringBootTest

public class UserRepoTest {

private Logger log = LoggerFactory.getLogger(this.getClass());

@Autowired

EntityManager em;

@Autowired

private UserRepo userRepo;

@Test

@Transactional

public void findUserByEmailTest() {

User user = userRepo.findUserById(1l);

log.info("Username -> {}",user.getUserName());

log.info("name -> {}", user.getName());

}

@Test

@Transactional

@DirtiesContext

public void userUpdateTest() {

User user = userRepo.userUpdate(1L,"saksham@123","sks123");

log.info("user updated -> {}",user);

}

@Test

@Transactional

void getUsers() {

TypedQuery<User> query = em.createNamedQuery("get\_all\_users",User.class);

List<User> result = query.getResultList();

log.info("Users List -> {}", result);

}

@Test

@Transactional

public void searchUser() {

Query query = em.createNativeQuery("select \* from User where user\_id = :userId", User.class);

query.setParameter("userId",1L);

List result = query.getResultList();

log.info("specified user -> {}", result);

}

@Test

@Transactional

public void logintest() {

userRepo.verifyUser("r@gmail,com","1");

}

}