**INFO6250: Web Development Tools and Methods, Section 06**

**Title: Best Buy Deals**

**By**

**Ajay Mohandas (001426741)**

Contents

[Introduction 2](#_Toc7198827)

[Functionality 2](#_Toc7198828)

[Roles and Responsibilities 2](#_Toc7198829)

[Technologies Used 2](#_Toc7198830)

[Screenshot 3](#_Toc7198831)

[Database Relational Model 7](#_Toc7198832)

[Appendix 7](#_Toc7198833)

[Controller class files 7](#_Toc7198834)

[View 18](#_Toc7198835)

[POJO class 28](#_Toc7198836)

[Exception class 41](#_Toc7198837)

[DAO class 42](#_Toc7198838)

# Introduction

Comparison shopping websites are channels for collecting product information, including pricing, from participating retailers and then display that collective information on a single results page in response to a shopper's search query. In this way, shoppers can compare prices, shipping options, and service from multiple retailers on a single page and choose the merchant that offers the best overall value.

# Functionality

My application will help customers compare the product with various other products enlisted by merchants. Criteria for comparison will be price, quality of the product and shipping charges. Any discounts or offers listed by merchants will also be considered to provide the best offers to the customers.

# Roles and Responsibilities

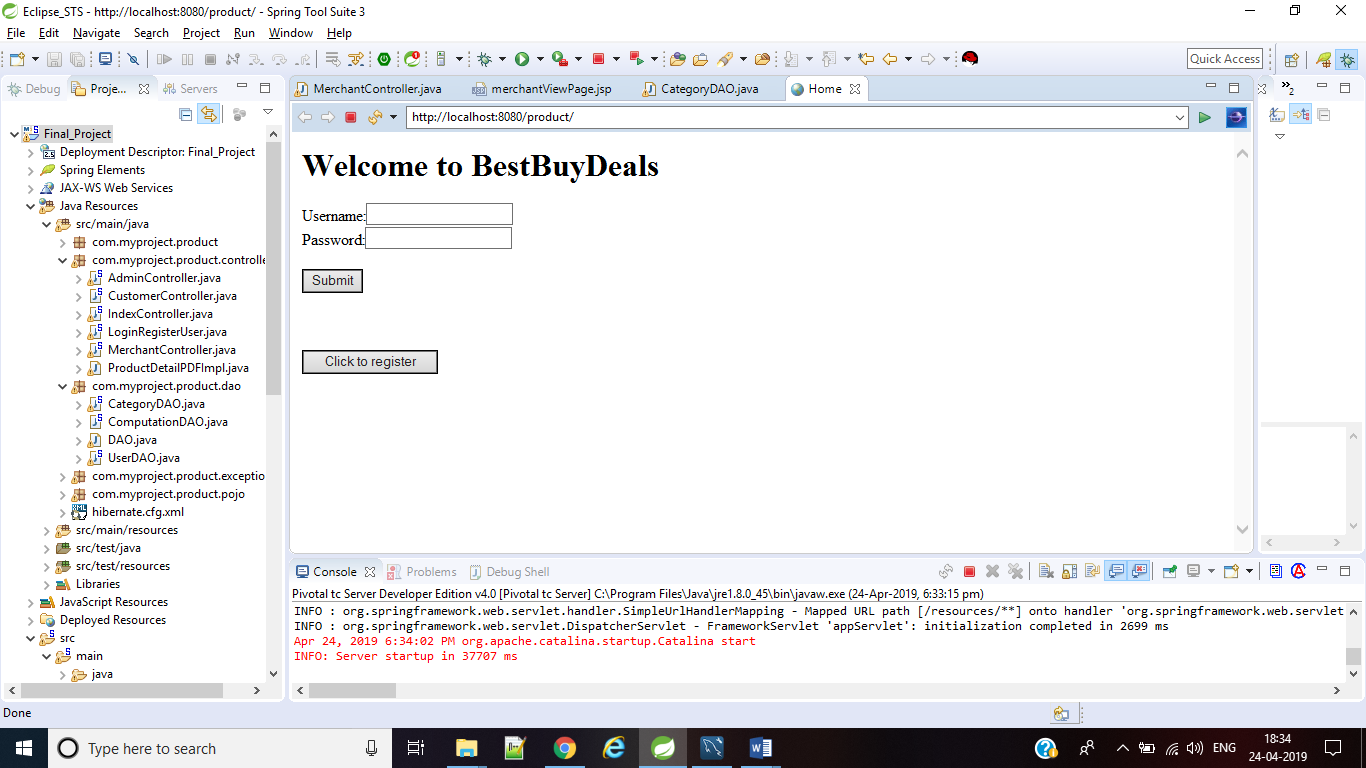
1. Application has three roles, they are admin, merchant and customer
2. Admin can view list of categories and add new one. Admin can view list of customers and merchants as well
3. Merchant can add new product, update them and check user’s feedback for a particular product
4. Customer can select a category from a list and can view details description of a product by selecting them
5. Customer can also get a pdf view of list of products under a category
6. After success registration an email confirmation is also sent to the user’s email address

# Technologies Used

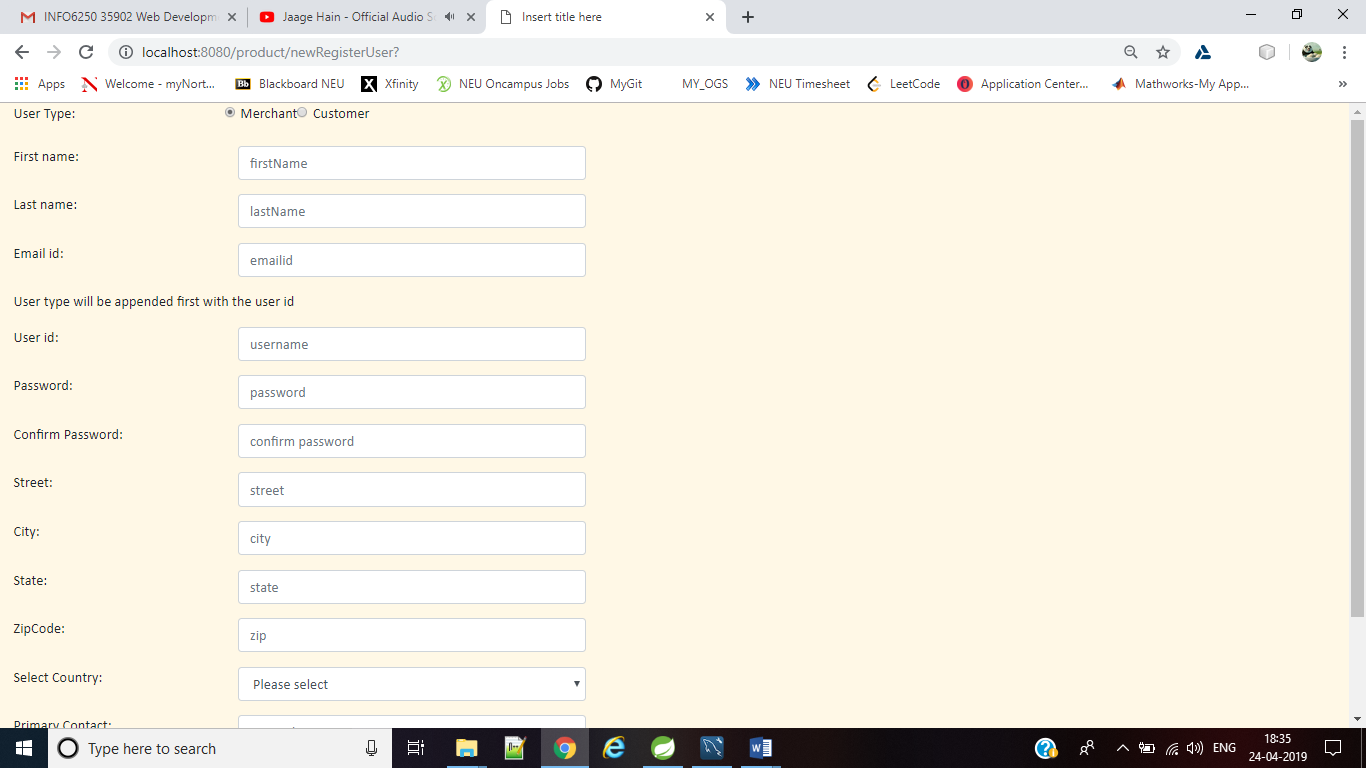
1. Used Spring MVC model and hibernate 5.4 to develop this application
2. MYSQL was used to persist the data and database configuration is stored in hibernate configuration xml file
3. Singleton design pattern is used to prevent duplicate object instantiation for connecting database to access data
4. Used itext dependency for PDF view of list of products
5. After successful registration an email is sent to new user with username on it
6. Apache common package is used to upload a file
7. Bcrypt - a mindrot dependency is used to hash the password and stored in database

# Screenshot

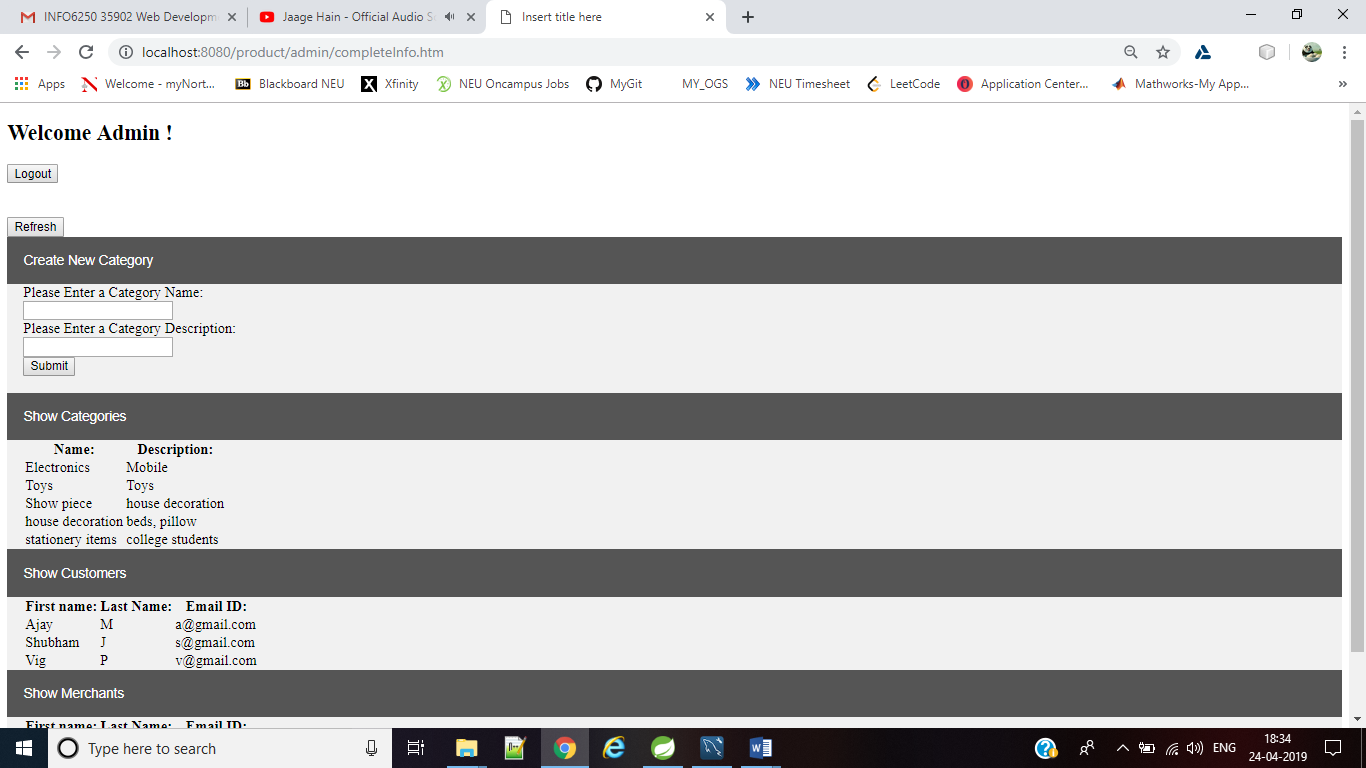
Login Page



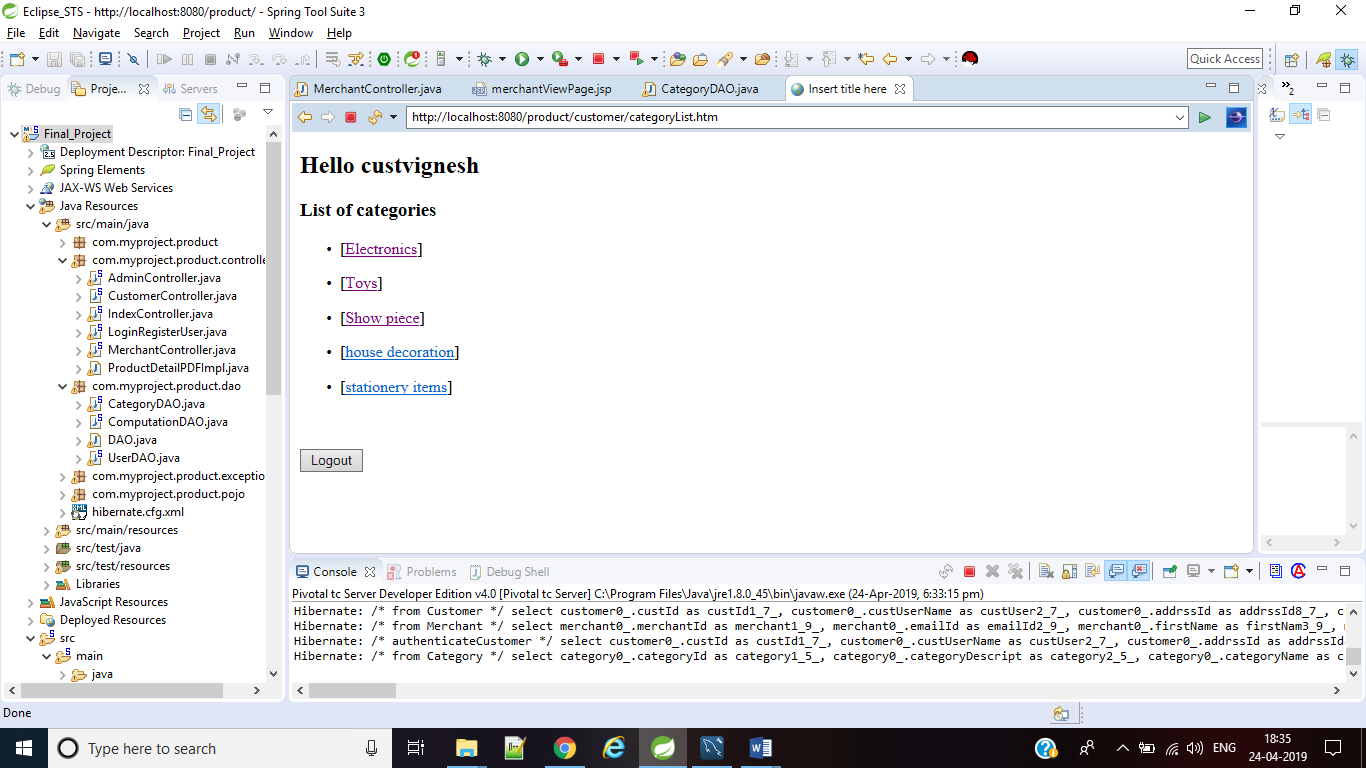
Registration page

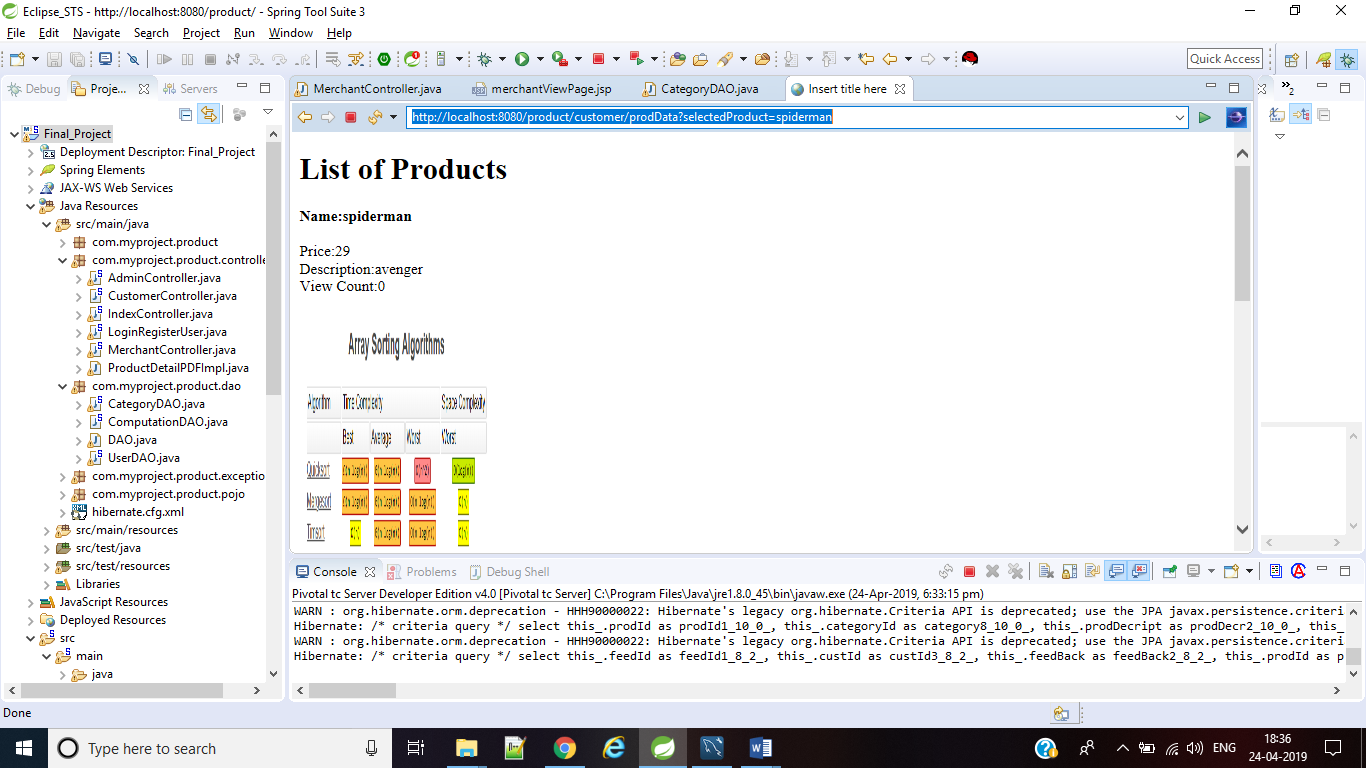


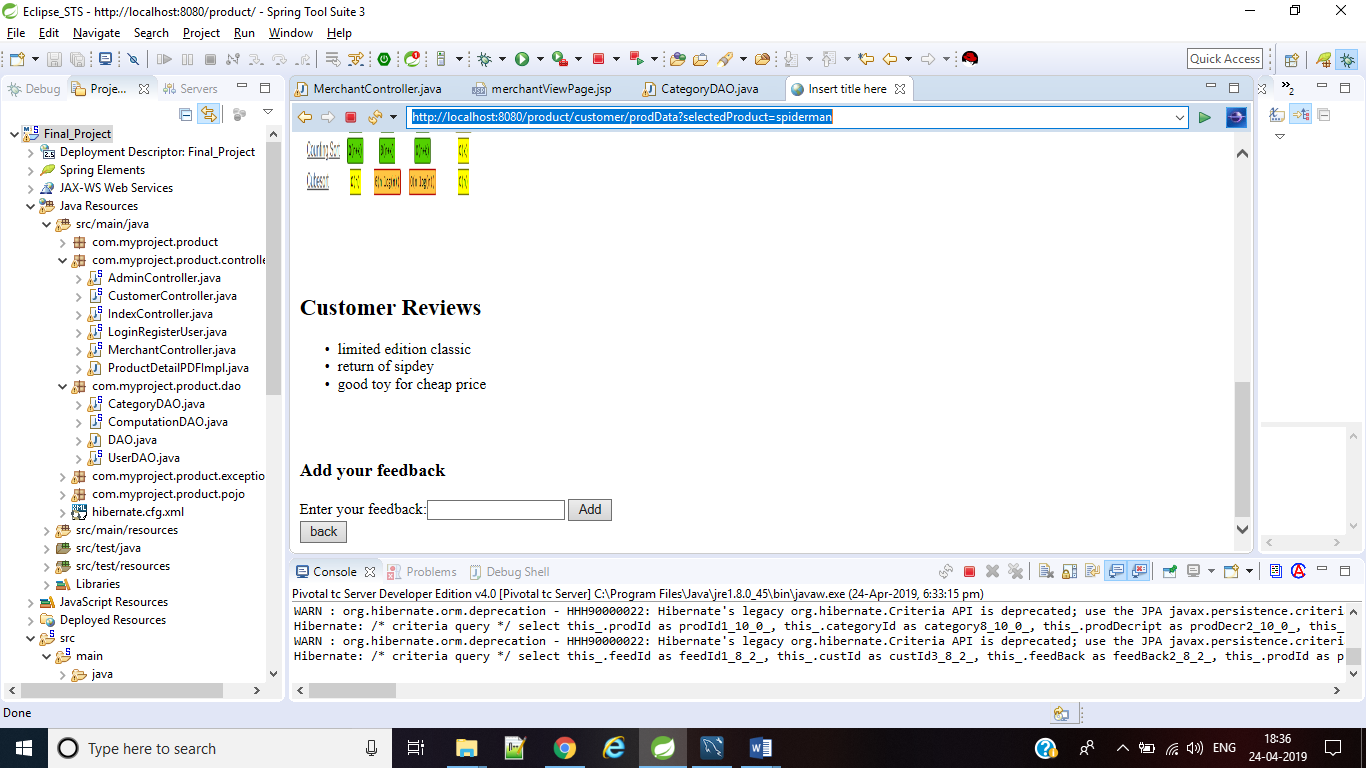
Admin page



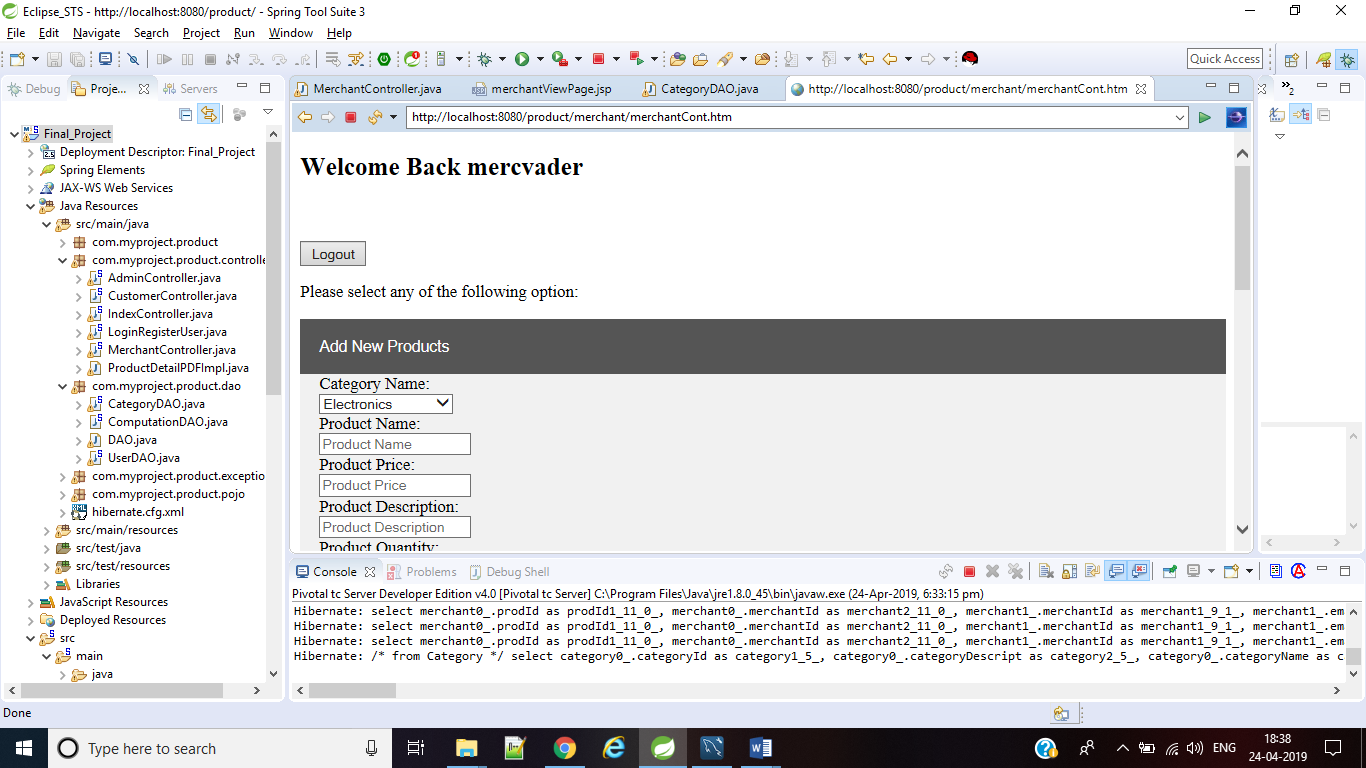
Customer page

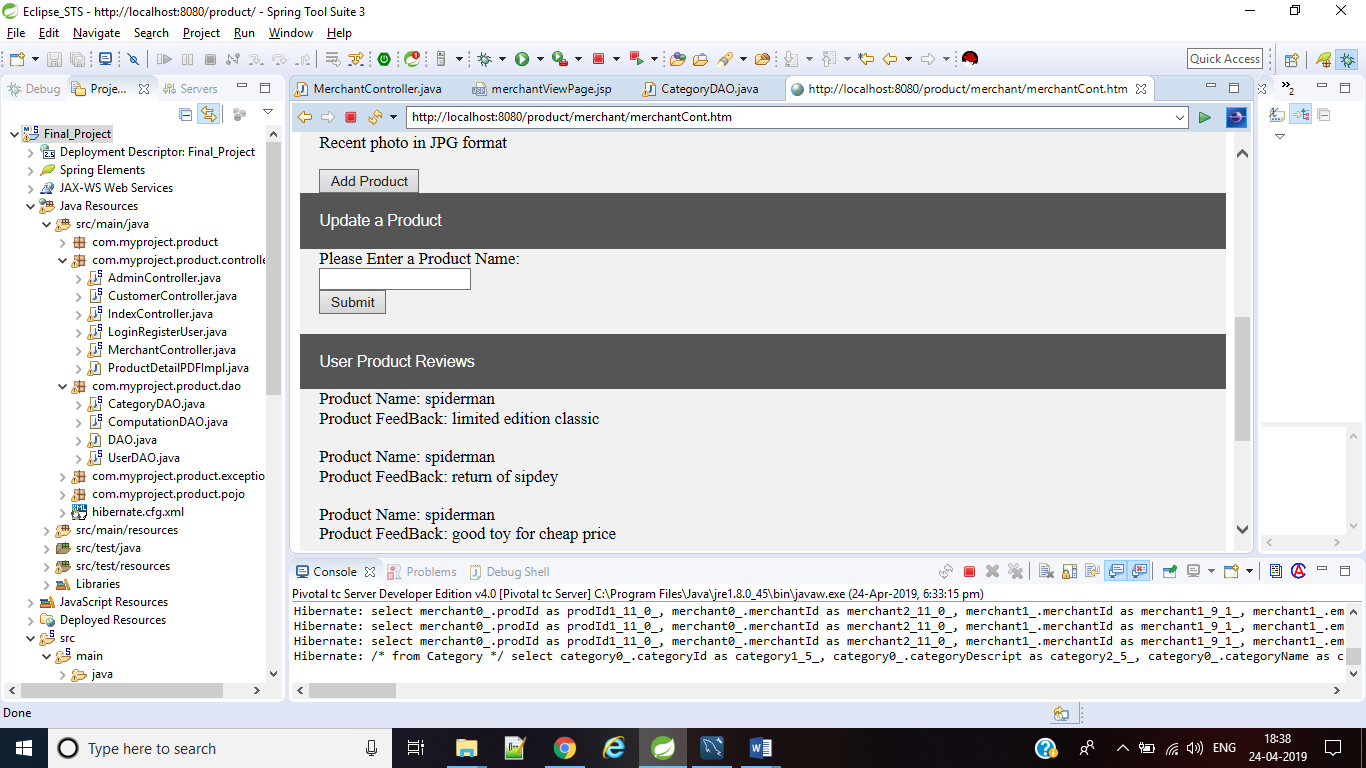


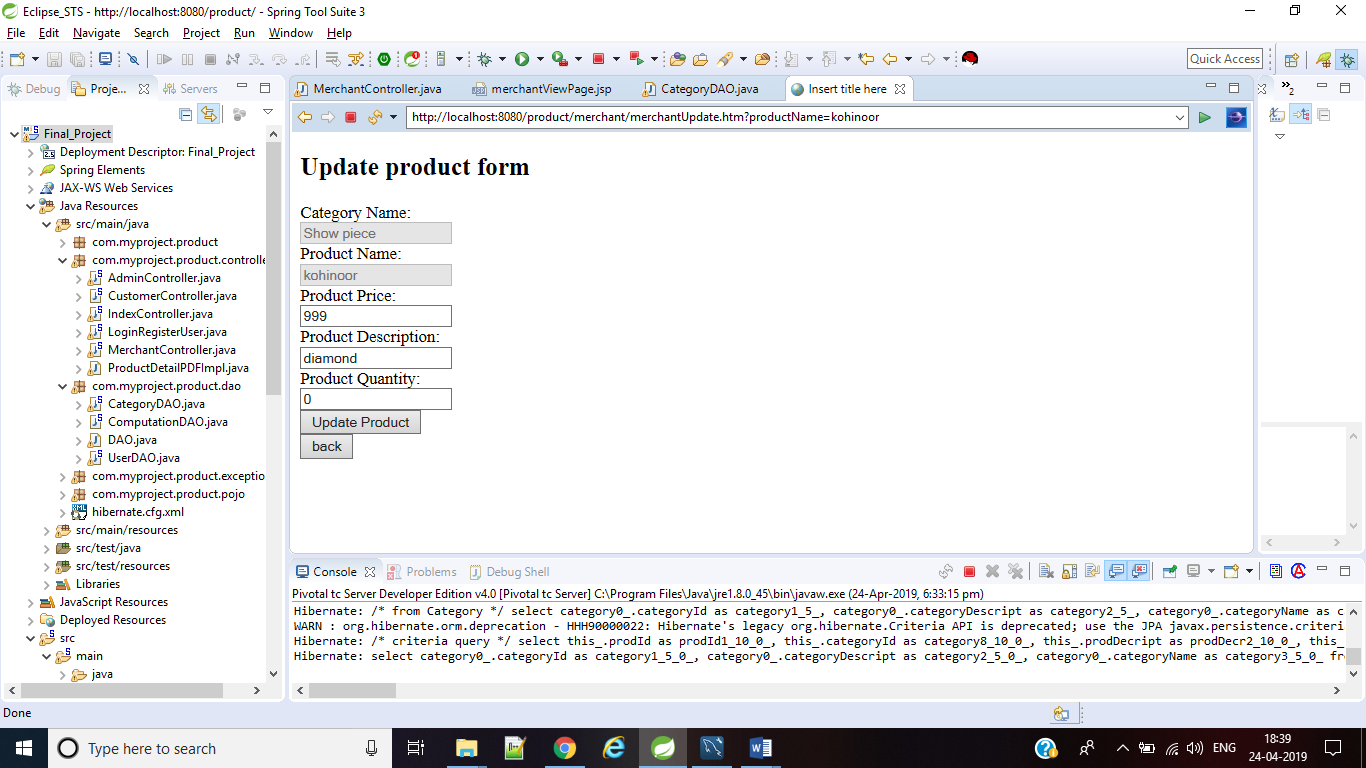




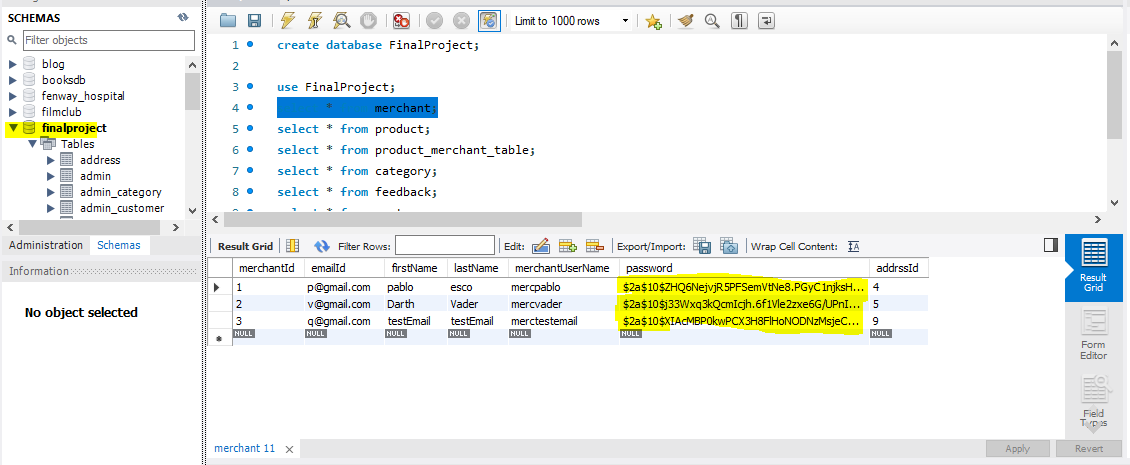
Merchant page



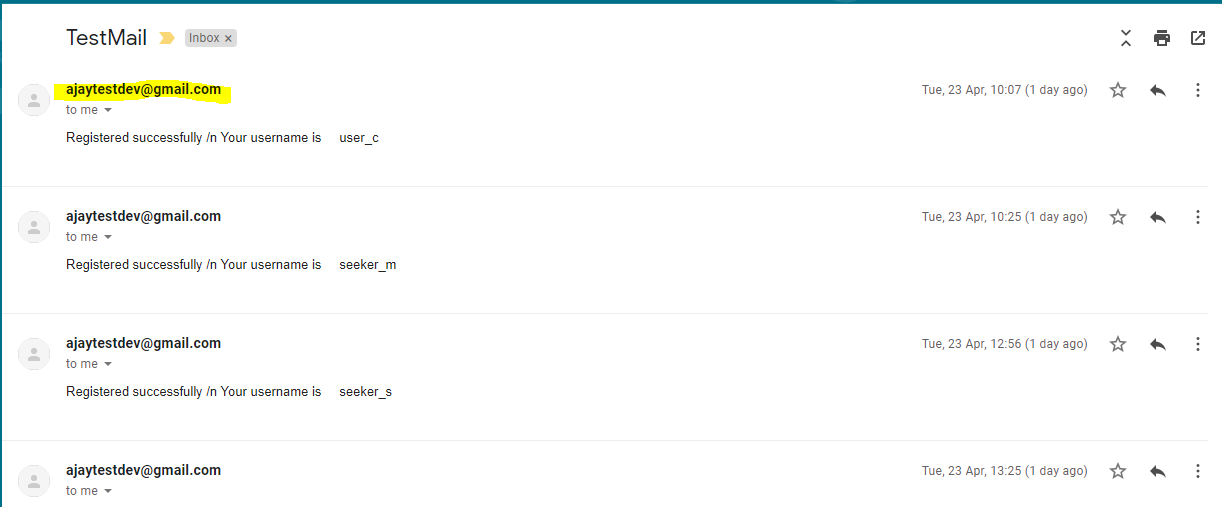




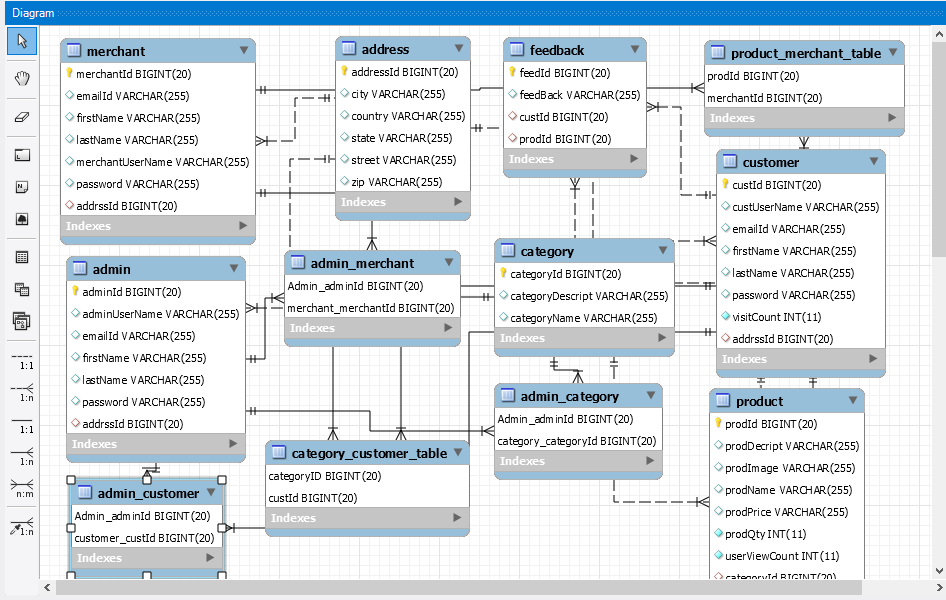
Password encryption



Email confirmation



# Database Relational Model



# Appendix

## Controller class files

1. AdminController

package com.myproject.product.controller;

import java.util.List;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpSession;

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

import org.springframework.stereotype.Controller;

import org.springframework.ui.ModelMap;

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

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

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

import org.springframework.web.servlet.ModelAndView;

import com.myproject.product.dao.CategoryDAO;

import com.myproject.product.dao.UserDAO;

import com.myproject.product.exception.ProductException;

import com.myproject.product.exception.UserException;

import com.myproject.product.pojo.Category;

import com.myproject.product.pojo.Product;

@Controller

@RequestMapping("/admin")

public class AdminController {

@Autowired

private UserDAO userDao;

@Autowired

private CategoryDAO categoryDao;

//home page for admin

@RequestMapping(value = "/completeInfo.htm", method = RequestMethod.GET)

public ModelAndView showForm(HttpServletRequest request, ModelMap model, Product product) {

ModelAndView mv=null;

HttpSession session = request.getSession();

//getting all the category list

try {

session.setAttribute("categoryList", categoryDao.catList() );

} catch (ProductException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

try {

session.setAttribute("curtomerList", userDao.custList());

} catch (UserException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

try {

session.setAttribute("merchantList", userDao.merchList());

} catch (UserException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

mv = new ModelAndView("adminPage");

return mv;

}

@RequestMapping(value = "/createCategory.htm", method = RequestMethod.POST)

public ModelAndView createCategory(HttpServletRequest req,ModelMap model, HttpSession session) throws ServletException, ProductException{

ModelAndView mv=null;

String catName = req.getParameter("categoryName");

List<Category> categoryList= categoryDao.catList();

for (Category ctemp: categoryList) {

if (ctemp.getCategoryName().equalsIgnoreCase(catName)) {

System.out.println("test cat");

return new ModelAndView("adminError");

}

}

Category c = new Category();

c.setCategoryName(catName);

c.setCategoryDescript(req.getParameter("categoryDesc"));

try {

int flag = categoryDao.createCategory(c);

if (flag==1) {

return new ModelAndView("adminPage");

}

} catch (ProductException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

return new ModelAndView("adminPage");

}

}

1. CustomerController

package com.myproject.product.controller;

import java.util.List;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import javax.servlet.http.HttpSession;

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

import org.springframework.stereotype.Controller;

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

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

import org.springframework.web.servlet.ModelAndView;

import org.springframework.web.servlet.View;

import com.myproject.product.dao.CategoryDAO;

import com.myproject.product.dao.UserDAO;

import com.myproject.product.exception.FeedbackException;

import com.myproject.product.exception.ProductException;

import com.myproject.product.exception.UserException;

import com.myproject.product.pojo.Category;

import com.myproject.product.pojo.Customer;

import com.myproject.product.pojo.Feedback;

import com.myproject.product.pojo.Product;

@Controller

@RequestMapping("/customer")

public class CustomerController {

@Autowired

private CategoryDAO categoryDao;

@Autowired

private UserDAO userDao;

@RequestMapping(value = "/categoryList.htm", method = RequestMethod.GET)

public ModelAndView getCategoryList(HttpServletRequest req, HttpServletResponse resp) throws ServletException{

HttpSession session = req.getSession();

ModelAndView mv = null;

try {

List<Category> catList = categoryDao.catList();

session.setAttribute("catList", catList);

mv = new ModelAndView("custViewPage");

} catch (ProductException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

return mv;

}

@RequestMapping(value = "/getProduct", method = RequestMethod.GET)

public ModelAndView getproductList(HttpServletRequest req, HttpServletResponse resp) throws ServletException{

String selectedCategory = req.getParameter("selectedCategory");

HttpSession session = req.getSession();

ModelAndView mv = null;

try {

List<Product> prodList = categoryDao.prodList(selectedCategory);

session.setAttribute("allProdList", prodList);

session.setAttribute("categoryName", selectedCategory);

mv = new ModelAndView("productViewPage");

} catch (ProductException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

return mv;

}

@RequestMapping(value = "/prodData", method = RequestMethod.GET)

public ModelAndView getproductDetails(HttpServletRequest req, HttpServletResponse resp) throws ServletException{

String selectedProduct = req.getParameter("selectedProduct");

HttpSession session = req.getSession();

ModelAndView mv = null;

try {

Product prodList = categoryDao.productDescript(selectedProduct);

List<Feedback> feed = categoryDao.getFeedback(selectedProduct);

session.setAttribute("prodList", prodList);

session.setAttribute("feedList", feed);

mv = new ModelAndView("productDescription");

} catch (ProductException|FeedbackException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

return mv;

}

@RequestMapping(value = "/getPDFView", method = RequestMethod.GET)

public View getPDFViewProductDetails(HttpServletRequest req, HttpServletResponse resp) {

String selectedCategory = req.getParameter("selectedCategory");

View view = new ProductDetailPDFImpl(categoryDao,selectedCategory);

return view;

}

@RequestMapping(value = "/addFeedback", method = RequestMethod.POST)

public ModelAndView addFeedback(HttpServletRequest req, HttpServletResponse resp) {

ModelAndView mv = null;

Feedback f = new Feedback();

HttpSession session = req.getSession();

String feedback = req.getParameter("feedback");

Product product = (Product) session.getAttribute("prodList");

String customer = (String)session.getAttribute("customerUserName");

try {

Customer cust = userDao.getCustDetail(customer);

f.setCustomer(cust);

f.setFeedBack(feedback);

f.setProduct(product);

int flag = categoryDao.addFeedBack(f);

if (flag==1)

mv = new ModelAndView("productDescription");

else

mv = new ModelAndView("errorRegister");

} catch (UserException|ProductException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

return mv;

}

}

1. LoginRegister

package com.myproject.product.controller;

import java.util.HashSet;

import java.util.Set;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import javax.servlet.http.HttpSession;

import org.mindrot.jbcrypt.BCrypt;

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

import org.springframework.stereotype.Controller;

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

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

import org.springframework.web.servlet.ModelAndView;

import org.springframework.web.servlet.view.RedirectView;

import org.apache.commons.mail.\*;

import com.myproject.product.dao.UserDAO;

import com.myproject.product.exception.UserException;

import com.myproject.product.pojo.\*;

@Controller

@RequestMapping("/")

public class LoginRegisterUser {

@Autowired

private UserDAO userDao;

@RequestMapping(value = "/login", method = RequestMethod.POST)

public ModelAndView loginPage(HttpServletRequest req, HttpServletResponse resp) throws ServletException{

ModelAndView mv=null;

String user = req.getParameter("username");

String pswd = req.getParameter("password");

try {

String checkValidUser = userDao.authenticateUser(user, pswd);

if (checkValidUser.startsWith("admin"))

mv = new ModelAndView(new RedirectView("admin/completeInfo.htm",false));

else if (checkValidUser.startsWith("merc")) {

HttpSession session = req.getSession();

session.setAttribute("merchantName",user);

mv = new ModelAndView(new RedirectView("merchant/merchantCont.htm",false));}

else if (checkValidUser.startsWith("cust"))

{

HttpSession session = req.getSession();

session.setAttribute("customerUserName",user);

mv = new ModelAndView(new RedirectView("customer/categoryList.htm",false));

}

else if (checkValidUser.equalsIgnoreCase("notAUser"))

mv = new ModelAndView("registerUser");

} catch (UserException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

return mv;

}

@RequestMapping(value = "/register", method = RequestMethod.POST)

public ModelAndView register(HttpServletRequest req, HttpServletResponse resp) throws UserException{

ModelAndView mv=null;

String pswd = req.getParameter("password");

String confirmpswd = req.getParameter("confirmpassword");

if (pswd.equals(confirmpswd)) {

String fName = req.getParameter("firstName");

String lName = req.getParameter("lastName");

String emailId = req.getParameter("emailId");

String street = req.getParameter("street");

String city = req.getParameter("city");

String state = req.getParameter("state");

String zip = req.getParameter("zip");

String country = req.getParameter("Country");

long primaryContact = Long.parseLong(req.getParameter("primePhone"));

long secContatc = Long.parseLong(req.getParameter("secondPhone"));

String username = req.getParameter("username");

Address address = new Address();

address.setStreet(street);

address.setCity(city);

address.setState(state);

address.setZip(zip);

address.setCountry(country);

/\*Set<Phone> ph = new HashSet<Phone>();

ph.add(new Phone(primaryContact));

ph.add(new Phone(secContatc));\*/

String hashedPswd = BCrypt.hashpw(pswd, BCrypt.gensalt());

String userType = req.getParameter("usertype");

if (userType.equalsIgnoreCase("Merchant"))

{

Merchant merch = new Merchant();

merch.setFirstName(fName);

merch.setLastName(lName);

merch.setEmailId(emailId);

merch.setMerchantUserName("merc"+username);

merch.setMerchantAddress(address);

//merch.setContact(ph);

merch.setPassword(hashedPswd);

int flag = userDao.registerMerchant(merch);

try {

SendEmail("ajaymohandas89@gmail.com",merch.getMerchantUserName());

} catch (EmailException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

if (flag==1)

mv = new ModelAndView("home");

}else if (userType.equalsIgnoreCase("Customer"))

{

Customer cust = new Customer();

cust.setFirstName(fName);

cust.setLastName(lName);

cust.setEmailId(emailId);

cust.setCustUserName("cust"+username);

cust.setCustomerAddress(address);

//cust.setContact(ph);

cust.setPassword(hashedPswd);

cust.setVisitCount(0);

int flag = userDao.registerCustomer(cust);

try {

SendEmail("ajaymohandas89@gmail.com",cust.getCustUserName());

} catch (EmailException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

if (flag==1)

mv = new ModelAndView("home");

}

}else {

mv = new ModelAndView("errorRegister");

}

return mv;

}

public void SendEmail(String emailID,String username) throws EmailException {

Email email = new SimpleEmail();

email.setHostName("smtp.googlemail.com");

email.setSmtpPort(465);

//User your gmail username and password

email.setAuthenticator(new DefaultAuthenticator("ajaytestDev@gmail.com", "qwerty123M"));

email.setSSLOnConnect(true);

email.setFrom("no-reply@msis.neu.edu");

email.setSubject("TestMail");

email.setMsg("Registered successfully "+" Your username is "+ username);

email.addTo(emailID);

email.send();

}

@RequestMapping(value = "/logout.htm", method = RequestMethod.POST)

public ModelAndView logout(HttpServletRequest request, HttpServletResponse response) {

request.getSession().invalidate();

return new ModelAndView("home");

}

@RequestMapping(value = "/newRegisterUser", method = RequestMethod.GET)

public ModelAndView newUserRegister(HttpServletRequest req, HttpServletResponse resp) throws ServletException{

return new ModelAndView("registerUser");

}

}

1. MerchantController

package com.myproject.product.controller;

import java.util.List;

import java.io.File;

import java.io.FileOutputStream;

import java.util.ArrayList;

import java.util.Date;

import java.util.HashMap;

import java.util.HashSet;

import java.util.Hashtable;

import java.util.Set;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import javax.servlet.http.HttpSession;

//import org.hibernate.mapping.Set;

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

import org.springframework.beans.propertyeditors.CustomCollectionEditor;

import org.springframework.stereotype.Controller;

import org.springframework.ui.ModelMap;

import org.springframework.validation.BindingResult;

import org.springframework.web.bind.ServletRequestDataBinder;

import org.springframework.web.bind.WebDataBinder;

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

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

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

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

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

import org.springframework.web.multipart.MultipartFile;

import org.springframework.web.servlet.ModelAndView;

import org.springframework.web.servlet.View;

import org.springframework.web.servlet.view.RedirectView;

import com.myproject.product.dao.CategoryDAO;

import com.myproject.product.dao.UserDAO;

import com.myproject.product.exception.ProductException;

import com.myproject.product.exception.UserException;

import com.myproject.product.pojo.Category;

import com.myproject.product.pojo.Feedback;

import com.myproject.product.pojo.Merchant;

import com.myproject.product.pojo.Product;

@Controller

@RequestMapping("/merchant")

public class MerchantController {

@Autowired

private UserDAO userDao;

@Autowired

private CategoryDAO categoryDao;

@RequestMapping(value = "/merchantCont.htm", method = RequestMethod.GET)

public String showForm(HttpServletRequest request, ModelMap model, Product product) throws ProductException, UserException {

Merchant productMerchant = null;

//to print feedback for a given merchant

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

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

HttpSession session = request.getSession();

//to get all the feedbacks of a particular product

String merchantName = (String)session.getAttribute("merchantName");

for(Merchant m:categoryDao.mercList()) {

if(m.getMerchantUserName().equals(merchantName)) {

productMerchant = m;

break;

}

}

for(Product p:categoryDao.allProdList()) {

Set<Merchant> merchantSet = new HashSet<>();

merchantSet = p.getMerchant();

for(Merchant m:merchantSet) {

if(m.equals(productMerchant)) {

Set<Feedback> fee = p.getFeedback();

for(Feedback f:fee) {

prod.add(p.getProdName());

feed.add(f.getFeedBack());

}

}

}

}

session.setAttribute("Feedbacks", feed);

session.setAttribute("Product",prod);

//to set category while adding a product

session.setAttribute("categoryList", categoryDao.catList());

return "merchantViewPage";

}

//to disallow spring to bind the category object

@InitBinder

public void customBinding(WebDataBinder binder) {

binder.setDisallowedFields(new String[] {"category"});

}

//to create new product

@RequestMapping(value = "/merchantCont.htm", method = RequestMethod.POST)

public ModelAndView addProduct(@ModelAttribute("product") Product product,HttpServletRequest req,ModelMap model,HttpSession session) throws ServletException, ProductException, UserException{

ModelAndView mv=null;

Category productCategory = null;

Merchant productMerchant = null;

//Setting the category of the product

String categoryName = req.getParameter("categoryName");

for(Category c:categoryDao.catList()) {

if(c.getCategoryName().equals(categoryName)) {

productCategory = c;

break;

}

}

product.setCategory(productCategory);

//setting Merchant of the new product

String merchantName = (String)session.getAttribute("merchantName");

for(Merchant m:categoryDao.mercList()) {

if(m.getMerchantUserName().equals(merchantName)) {

productMerchant = m;

break;

}

}

Set merchantSet = new HashSet<>();

Set productSet = new HashSet<>();

productSet.add(product);

merchantSet.add(productMerchant);

productMerchant.setProduct(productSet);

product.setMerchant(merchantSet);

//setting the photo of the product

String localPath = "C:\\Users\\Welcome\\Desktop\\FinalProject\_ProductFiles";

String photoNewName = generateFileName(product.getPhoto());

product.setProdImage(photoNewName);

try {

product.getPhoto().transferTo(new File(localPath,photoNewName));

}catch(Exception e) {

}

try {

int flag = categoryDao.createProduct(product);

if (flag==1) {

return new ModelAndView("productaddedsuccessfully");

}

} catch (ProductException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

return null;

}

private String generateFileName(MultipartFile multipart) {

return new Date().getTime() + "-"+multipart.getOriginalFilename().replace(" ", "-");

}

//to display product form for updating

@RequestMapping(value = "/merchantUpdate.htm", method = RequestMethod.GET)

public ModelAndView updateProductForm(HttpServletRequest req, HttpServletResponse resp,HttpSession session,Product product) throws ServletException{

ModelAndView mv=null;

Product selectedProduct = null;

String productName = req.getParameter("productName");

try {

List<Product> prod = new ArrayList<Product>();

prod = categoryDao.productDetails(productName);

for(Product p:prod) {

if(p.getProdName().equals(productName)) {

selectedProduct = p;

break;

}else {

return new ModelAndView("merchantViewPage");

}

}

session = req.getSession();

session.setAttribute("product", selectedProduct);

return new ModelAndView("updateproduct");

} catch (ProductException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

return null;

}

//to update the product as given by merchant

@RequestMapping(value = "/merchantUpdate.htm", method = RequestMethod.POST)

public ModelAndView updateProduct(Product product,HttpServletRequest req,ModelMap model, HttpSession session) throws ServletException{

ModelAndView mv=null;

product = (Product) session.getAttribute("product");

//product.setProdName(req.getParameter("name"));

product.setProdPrice(req.getParameter("price"));

product.setProdDecript(req.getParameter("desc"));

product.setProdQty(Integer.parseInt(req.getParameter("quantity")));

try {

int flag = categoryDao.updateProduct(product);

if (flag==1) {

return new ModelAndView("merchantViewPage");

}

} catch (ProductException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

return null;

}

}

1. ProductDetailPDFImpl

package com.myproject.product.controller;

import java.util.List;

import java.util.Map;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

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

import org.springframework.web.servlet.ModelAndView;

import org.springframework.web.servlet.view.document.AbstractPdfView;

import com.lowagie.text.Chunk;

import com.lowagie.text.Document;

import com.lowagie.text.Element;

import com.lowagie.text.Font;

import com.lowagie.text.Header;

import com.lowagie.text.Paragraph;

import com.lowagie.text.Phrase;

import com.lowagie.text.pdf.PdfWriter;

import com.myproject.product.dao.CategoryDAO;

import com.myproject.product.exception.ProductException;

import com.myproject.product.pojo.Product;

public class ProductDetailPDFImpl extends AbstractPdfView{

private CategoryDAO categoryDao;

private String selectedCategory;

public ProductDetailPDFImpl(CategoryDAO categoryDao,String selectedCategory) {

this.selectedCategory = selectedCategory;

this.categoryDao = categoryDao;

}

@Override

protected void buildPdfDocument(Map<String, Object> model, Document document, PdfWriter writer,

HttpServletRequest request, HttpServletResponse response) throws Exception {

Font titleFont = new Font(Font.TIMES\_ROMAN, 24, Font.BOLD);

Paragraph title = new Paragraph("List of Products", titleFont );

Font prodTitleFont = new Font(Font.TIMES\_ROMAN, 18, Font.BOLD);

document.add(title);

String cat = selectedCategory;

try {

List<Product> prodList = categoryDao.prodList(cat);

for(Product p: prodList) {

document.add(new Phrase("Name "+p.getProdName(),prodTitleFont)) ;

document.add(new Phrase(" Price "+p.getProdPrice()));

document.add(new Phrase(" Quantity "+String.valueOf(p.getProdQty())));

document.add(new Phrase(" Number of times User Viewed this product "+String.valueOf(p.getUserViewCount())));

document.add( Chunk.NEWLINE );

document.add( Chunk.NEWLINE );

}

} catch (ProductException e) {

// TODO Auto-generated catch block

e.printStackTrace();

} } }

## View

1. adminPage

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

pageEncoding="ISO-8859-1"%>

<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>

<%@ page session="true" %>

<%@taglib uri="http://www.springframework.org/tags/form" prefix="form"%>

<!DOCTYPE html>

<html>

<head>

<title>Insert title here</title>

<meta name="viewport" content="width=device-width, initial-scale=1">

<style>

.collapsible {

background-color: #777;

color: white;

cursor: pointer;

padding: 18px;

width: 100%;

border: none;

text-align: left;

outline: none;

font-size: 15px;

}

.active**,** .collapsible:hover {

background-color: #555;

}

.content {

padding: 0 18px;

display: none;

overflow: hidden;

background-color: #f1f1f1;

}

</style>

</head>

<body>

<h2>Welcome Admin ${sessionScope.adminName}!</h2>

<form action="/product/logout.htm"class="form-horizontal" method = "post">

<div class="form-group">

<input type="submit" value="Logout">

</div>

</form>

<br/><br/>

<form action="/product/admin/completeInfo.htm" method="get">

<input type="submit" value="Refresh" />

</form>

<button class="collapsible">Create New Category</button>

<div class="content">

<form action="/product/admin/createCategory.htm"class="form-horizontal" method = "post">

<div class="form-group">

<label for="Category Name" class="col-sm-4 control-label">Please Enter a Category Name:</label>

<div class="col-sm-8">

<input type="text" name="categoryName" value=""/> <br/>

<label for="Category Name" class="col-sm-4 control-label">Please Enter a Category Description:</label>

<div class="col-sm-8">

<input type="text" name="categoryDesc" value=""/>

</div>

<input type="submit" value="Submit">

</div>

</div>

<br/>

</form>

</div>

<button class="collapsible">Show Categories</button>

<div class="content">

<table>

<tr>

<th>Name: </th><th>Description: </th>

</tr>

<c:forEach var="category" items="${sessionScope.categoryList}">

<tr>

<td>${category.categoryName}</td><td>${category.categoryDescript}</td>

</tr>

</c:forEach>

</table>

</div>

<button class="collapsible">Show Customers</button>

<div class="content">

<table>

<tr>

<th>First name: </th><th>Last Name: </th><th>Email ID: </th>

</tr>

<c:forEach var="customer" items="${sessionScope.curtomerList}">

<tr>

<td>${customer.firstName}</td><td>${customer.lastName}</td><td>${customer.emailId}</td>

</tr>

</c:forEach>

</table>

</div>

<button class="collapsible">Show Merchants</button>

<div class="content">

<table>

<tr>

<th>First name: </th><th>Last Name: </th><th>Email ID: </th>

</tr>

<c:forEach var="merc" items="${sessionScope.merchantList}">

<tr>

<td>${merc.firstName}</td><td>${merc.lastName}</td><td>${merc.emailId}</td>

</tr>

</c:forEach>

</table>

</div>

<script>

**var** coll = document.getElementsByClassName("collapsible");

**var** i;

**for** (i = 0; i < coll.length; i++) {

coll[i].addEventListener("click", **function**() {

**this**.classList.toggle("active");

**var** content = **this**.nextElementSibling;

**if** (content.style.display === "block") {

content.style.display = "none";

} **else** {

content.style.display = "block";

}

});

}

</script>

</body>

</html>

1. custViewPage

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

pageEncoding="ISO-8859-1"%>

<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>

<%@page import="com.myproject.product.pojo.\*"%>

<!DOCTYPE html>

<html>

<head>

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

<title>Insert title here</title>

</head>

<body>

<h2>Hello ${sessionScope.customerUserName}</h2>

<h3>List of categories</h3>

<c:forEach var = "cart" items= "${sessionScope.catList}">

<ul>

<li>[<a href="/product/customer/getProduct?selectedCategory=${cart.categoryName}">${cart.categoryName}</a>]&nbsp;&nbsp;</li>

</ul>

</c:forEach>

<br/><br/>

<form action="/product/logout.htm"class="form-horizontal" method = "post">

<div class="form-group">

<input type="submit" value="Logout">

</div>

</form>

</body>

</html>

1. Home

<%@ taglib uri=*"http://java.sun.com/jsp/jstl/core"* prefix=*"c"* %>

<%@ page session=*"false"* %>

<html>

<head>

<title>Home</title>

<style>

**input**[type=submit] {

padding:*5px 15px*;

background:*#E6E6FA*;

border:*0 none*;

cursor:*pointer*;

-webkit-border-radius: *5px*;

border-radius: *5px*;

}

</style>

</head>

<body>

<h1>

Welcome to BestBuyDeals

</h1>

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

Username:<input type=*"text"* name=*"username"* required/><br/>

Password:<input type=*"password"* name=*"password"* required/><br/><br/>

<input type=*"submit"* value=*"Submit"*/>

</form>

<br/><br/>

<form action=*"/product/newRegisterUser"* method=*"get"*>

<input type=*"submit"* value=*"Click to register"* />

</form>

</body>

</html>

1. merchantViewPage

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

pageEncoding="ISO-8859-1"%>

<%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>

<%@ page session="true" %>

<%@taglib uri="http://www.springframework.org/tags/form" prefix="form"%>

<!DOCTYPE html>

<html>

<head>

<meta name="viewport" content="width=device-width, initial-scale=1">

<style>

.collapsible {

background-color: #777;

color: white;

cursor: pointer;

padding: 18px;

width: 100%;

border: none;

text-align: left;

outline: none;

font-size: 15px;

}

.active**,** .collapsible:hover {

background-color: #555;

}

.content {

padding: 0 18px;

display: none;

overflow: hidden;

background-color: #f1f1f1;

}

</style>

</head>

<body>

<h2>Welcome Back ${sessionScope.merchantName}</h2>

<br/><br/>

<form action="/product/logout.htm"class="form-horizontal" method = "post">

<div class="form-group">

<input type="submit" value="Logout">

</div>

</form>

<p>Please select any of the following option:</p>

<button class="collapsible">Add New Products</button>

<div class="content">

<form:form commandName="product" enctype="multipart/form-data" class="form-horizontal">

<div class="form-group">

<label for="categoryName" class="col-sm-4 control-label">Category Name:</label>

<div class="col-sm-8">

<select name="categoryName">

<c:forEach items="${sessionScope.categoryList}" var="category">

<option value="${category.categoryName}">${category.categoryName}</option>

</c:forEach>

</select>

</div>

</div>

<div class="form-group">

<label for=prodName class="col-sm-4 control-label">Product

Name: </label>

<div class="col-sm-8">

<form:input type="text" class="form-control" id="prodName"

path="prodName" placeholder="Product Name" required="required" />

</div>

</div>

<div class="form-group">

<label for="last" class="col-sm-4 control-label">Product Price:</label>

<div class="col-sm-8">

<form:input type="number" class="form-control" id="prodPrice" path="prodPrice"

placeholder="Product Price" required="required" />

</div>

</div>

<div class="form-group">

<label for="Product Description" class="col-sm-4 control-label">Product Description:</label>

<div class="col-sm-8">

<form:input type="text" class="form-control" id="prodDescript" path="prodDecript"

placeholder="Product Description" required="required" />

</div>

</div>

<div class="form-group">

<label for="Product Quantity" class="col-sm-4 control-label">Product Quantity:</label>

<div class="col-sm-8">

<form:input type="number" class="form-control" id="prodQty" path="prodQty"

placeholder="Product Quantity" required="required" />

</div>

</div>

<div class="form-group">

<label for="photo" class="col-sm-4 control-label">Photo: </label>

<div class="col-sm-4">

<input type="file" id="photo" name="photo" required="required" />

<p class="help-block">Recent photo in JPG format</p>

</div>

</div>

<div class="col-sm-offset-4 col-sm-8">

<input type="submit" class="btn btn-success" value="Add Product" />

</div>

</form:form>

</div>

<button class="collapsible">Update a Product</button>

<div class="content">

<form action="/product/merchant/merchantUpdate.htm" enctype="multipart/form-data" class="form-horizontal">

<div class="form-group">

<label for="Product Name" class="col-sm-4 control-label">Please Enter a Product Name:</label>

<div class="col-sm-8">

<input type="text" name="productName" value="" required/>

</div>

<input type="submit" value="Submit">

</div>

<br/>

</form>

</div>

<button class="collapsible">User Product Reviews</button>

<div class="content">

<c:forEach var="prod" items="${sessionScope.Product}" varStatus="feedback">

Product Name: <c:out value="${prod}"/><br/>

Product FeedBack: <c:out value="${sessionScope.Feedbacks[feedback.index]}"/><br/><br/>

</c:forEach>

</div>

<script>

**var** coll = document.getElementsByClassName("collapsible");

**var** i;

**for** (i = 0; i < coll.length; i++) {

coll[i].addEventListener("click", **function**() {

**this**.classList.toggle("active");

**var** content = **this**.nextElementSibling;

**if** (content.style.display === "block") {

content.style.display = "none";

} **else** {

content.style.display = "block";

}

});

}

</script>

</body>

</html>

1. Register

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

pageEncoding="ISO-8859-1"%>

<!DOCTYPE html>

<html>

<head>

<script>

**function** isPasswordMatch() {

**var** password = $("#txtNewPassword").val();

**var** confirmPassword = $("#txtConfirmPassword").val();

**if** (password != confirmPassword) $("#divCheckPassword").html("Passwords do not match!");

**else** $("#divCheckPassword").html("Passwords match.");

}

$(document).ready(**function** () {

$("#txtConfirmPassword").keyup(isPasswordMatch);

});

</script>

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

<title>Insert title here</title>

<link rel="stylesheet" type="text/css" href="https://stackpath.bootstrapcdn.com/bootstrap/4.2.1/css/bootstrap.min.css">

<!--javascript -->

<script type="text/javascript" src="https://stackpath.bootstrapcdn.com/bootstrap/4.2.1/js/bootstrap.bundle.min.js"></script>

<!-- to make responsive-->

<meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

<style type="text/css">

**body**

{

background-color: #fff8e6;

font-size: 16px;

font-family: Calibri;

}

.wrapper{

text-align :center;

}

.btn{

position:absolute;

top:102%;

}

.myreset{

position:absolute;

top:102%;

left:65%;

}

</style>

</head>

<body>

<form class="col-8" action="register" method="post">

<div class="row form-group">

<div class="col-3">

<label for="gender">User Type:</label>

</div>

<div class="col-5 form-check-inline">

<label class="radio-inline control-label">

<input type="radio" name="usertype" value="Merchant" class="form-check-input" checked>Merchant

</label>

<label class="radio-inline control-label">

<input type="radio" name="usertype" value="Customer" class="form-check-input">Customer

</label>

</div>

</div>

<div class="row form-group">

<div class="col-3">

<label for="firstName">First name:</label>

</div>

<div class="col-5">

<input type="text" name="firstName" placeholder="firstName" class="form-control" required>

</div>

</div>

<div class="row form-group">

<div class="col-3">

<label for="lastName">Last name:</label>

</div>

<div class="col-5">

<input type="text" name="lastName" placeholder="lastName" class="form-control" required>

</div>

</div>

<div class="row form-group">

<div class="col-3">

<label for="email">Email id:</label>

</div>

<div class="col-5">

<input type="email" name="emailId" placeholder="emailid" class="form-control" required>

</div>

</div>

<div>

<p>User type will be appended first with the user id</p>

<div class="row form-group">

<div class="col-3">

<label for="username">User id:</label>

</div>

<div class="col-5">

<input type="text" name="username" placeholder="username" class="form-control" required>

<span id="isE"></span>

</div>

</div>

<div class="row form-group">

<div class="col-3">

<label for="password">Password:</label>

</div>

<div class="col-5">

<input type="password" name="password" placeholder="password" class="form-control" id="txtNewPassword" >

</div>

</div>

<div class="row form-group">

<div class="col-3">

<label for="confirmpassword">Confirm Password:</label>

</div>

<div class="col-5">

<input type="password" name="confirmpassword"

placeholder="confirm password" class="form-control" id="txtConfirmPassword" onChange="isPasswordMatch()">

</div>

</div>

<div id="divCheckPassword"></div>

<div class="row form-group">

<div class="col-3">

<label for="street">Street:</label>

</div>

<div class="col-5">

<input type="text" name="street"

placeholder="street" class="form-control">

</div>

</div>

<div class="row form-group">

<div class="col-3">

<label for="city">City:</label>

</div>

<div class="col-5">

<input type="text" name="city"

placeholder="city" class="form-control">

</div>

</div>

<div class="row form-group">

<div class="col-3">

<label for="state">State:</label>

</div>

<div class="col-5">

<input type="text" name="state"

placeholder="state" class="form-control">

</div>

</div>

<div class="row form-group">

<div class="col-3">

<label for="zip">ZipCode:</label>

</div>

<div class="col-5">

<input type="text" name="zip"

placeholder="zip" class="form-control">

</div>

</div>

<div class="row form-group">

<div class="col-3">

<label for="country">Select Country:</label>

</div>

<div class="col-5">

<select name="Country" class="form-control">

<option value="" selected disabled>Please select</option>

<option value="USA"> USA </option>

<option value="INDIA"> INDIA </option>

<option value="CHINA"> CHINA </option>

</select>

</div>

</div>

<div class="row form-group">

<div class="col-3">

<label for="primePhone">Primary Contact:</label>

</div>

<div class="col-5">

<input type="text" name="primePhone"

placeholder="primePhone" class="form-control">

</div>

</div>

</div>

<div class="row form-group">

<div class="col-3">

<label for="secondPhone">Secondary Contact:</label>

</div>

<div class="col-5">

<input type="text" name="secondPhone"

placeholder="secondPhone" class="form-control">

</div>

</div>

<div class= "wrapper">

<input type="submit" class="btn btn-primary" value="Submit">

<button type="reset" class="myreset btn btn-secondary">Reset</button></div>

</form>

</body>

</html>

1. ProductionDescription

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

pageEncoding="ISO-8859-1"%>

<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>

<%@page import="com.myproject.product.pojo.\*"%>

<!DOCTYPE html>

<html>

<head>

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

<title>Insert title here</title>

</head>

<body>

<h1>List of Products</h1>

<c:set var="product" value= "${sessionScope.prodList}"/>

<b>Name:<c:out value="${product.getProdName()}"/></b><br/><br/>

Price:<c:out value="${product.getProdPrice()}"/><br/>

Description:<c:out value="${product.getProdDecript()}"/><br/>

View Count:<c:out value="${product.getUserViewCount()}"/><br/><br/>

<img src="<c:url value="C:/Users/Welcome/Desktop/FinalProject\_ProductFiles/${product.getProdImage()}"/>" alt="image"

width="200px" height="600px"/><br/><br/>

<br/><br/><h2>Customer Reviews</h2>

<ul>

<c:forEach var="feed" items="${sessionScope.feedList}">

<li><c:out value="${feed.getFeedBack()}"/></li>

</c:forEach>

</ul>

<br/>

<br/>

<h3>Add your feedback</h3>

<form action="/product/customer/addFeedback" method="post">

Enter your feedback:<input type="text" name="feedback"/>

<input type="submit" value="Add"/>

</form>

<form action="/product/customer/categoryList.htm" method="get">

<input type="submit" value="back" />

</form>

</body>

</html>

## POJO class

1. Admin

package com.myproject.product.pojo;

import java.util.Set;

import javax.persistence.CascadeType;

import javax.persistence.Entity;

import javax.persistence.FetchType;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.JoinColumn;

import javax.persistence.NamedQuery;

import javax.persistence.OneToMany;

import javax.persistence.OneToOne;

@NamedQuery(name="authenticateAdmin", query="from Admin where adminUserName=:username and password=:password")

@Entity

public class Admin {

@Id

@GeneratedValue(strategy=GenerationType.IDENTITY)

private long adminId;

private String firstName;

private String lastName;

private String emailId;

private String adminUserName;

private String password;

@OneToMany(fetch = FetchType.EAGER, cascade = CascadeType.ALL)

private Set<Category> category;

@OneToMany(fetch = FetchType.EAGER, cascade = CascadeType.ALL)

private Set<Customer> customer;

@OneToMany(fetch = FetchType.EAGER, cascade = CascadeType.ALL)

private Set<Merchant> merchant;

@OneToOne(fetch = FetchType.LAZY, cascade = CascadeType.ALL)

@JoinColumn(name="addrssId")

private Address adminAddress;

//@OneToMany(mappedBy="adminPhone",fetch = FetchType.LAZY, cascade = CascadeType.ALL)

//private Set<Phone> contact;

public Admin() {

}

public String getEmailId() {

return emailId;

}

public void setEmailId(String emailId) {

this.emailId = emailId;

}

public String getAdminUserName() {

return adminUserName;

}

public void setAdminUserName(String adminUserName) {

this.adminUserName = adminUserName;

}

public String getPassword() {

return password;

}

public void setPassword(String password) {

this.password = password;

}

public long getAdminId() {

return adminId;

}

public void setAdminId(long adminId) {

this.adminId = adminId;

}

public String getFirstName() {

return firstName;

}

public void setFirstName(String firstName) {

this.firstName = firstName;

}

public String getLastName() {

return lastName;

}

public void setLastName(String lastName) {

this.lastName = lastName;

}

public Set<Category> getCategory() {

return category;

}

public void setCategory(Set<Category> category) {

this.category = category;

}

public Set<Customer> getCustomer() {

return customer;

}

public void setCustomer(Set<Customer> customer) {

this.customer = customer;

}

public Set<Merchant> getMerchant() {

return merchant;

}

public void setMerchant(Set<Merchant> merchant) {

this.merchant = merchant;

}

public Address getAdminAddress() {

return adminAddress;

}

public void setAdminAddress(Address adminAddress) {

this.adminAddress = adminAddress;

}

/\*public Set<Phone> getContact() {

return contact;

}

public void setContact(Set<Phone> contact) {

this.contact = contact;

}\*/

}

Address

package com.myproject.product.pojo;

import javax.persistence.CascadeType;

import javax.persistence.Entity;

import javax.persistence.FetchType;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.OneToOne;

@Entity

public class Address {

public Address() {

}

@Id

@GeneratedValue(strategy=GenerationType.IDENTITY)

private long addressId;

private String street;

private String city;

private String state;

private String country;

private String zip;

@OneToOne(mappedBy="adminAddress",fetch = FetchType.LAZY, cascade = CascadeType.ALL)

private Admin admin;

@OneToOne(mappedBy="customerAddress",fetch = FetchType.LAZY, cascade = CascadeType.ALL)

private Customer customer;

@OneToOne(mappedBy="merchantAddress",fetch = FetchType.LAZY, cascade = CascadeType.ALL)

private Merchant merchant;

public Long getAddressId() {

return addressId;

}

public void setAddressId(long addressId) {

this.addressId = addressId;

}

public String getStreet() {

return street;

}

public void setStreet(String street) {

this.street = street;

}

public String getCity() {

return city;

}

public void setCity(String city) {

this.city = city;

}

public String getState() {

return state;

}

public void setState(String state) {

this.state = state;

}

public String getCountry() {

return country;

}

public void setCountry(String country) {

this.country = country;

}

public String getZip() {

return zip;

}

public void setZip(String zip) {

this.zip = zip;

}

public Admin getAdmin() {

return admin;

}

public void setAdmin(Admin admin) {

this.admin = admin;

}

public Customer getCustomer() {

return customer;

}

public void setCustomer(Customer customer) {

this.customer = customer;

}

public Merchant getMerchant() {

return merchant;

}

public void setMerchant(Merchant merchant) {

this.merchant = merchant;

}

}

Category

package com.myproject.product.pojo;

import java.util.HashSet;

import java.util.Set;

import javax.persistence.CascadeType;

import javax.persistence.Entity;

import javax.persistence.FetchType;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.JoinColumn;

import javax.persistence.JoinTable;

import javax.persistence.ManyToMany;

import javax.persistence.OneToMany;

@Entity

public class Category {

public Category() {

}

@Id

@GeneratedValue(strategy=GenerationType.IDENTITY)

private long categoryId;

private String categoryName;

private String categoryDescript;

@OneToMany(mappedBy="category",fetch = FetchType.LAZY, cascade = CascadeType.ALL)

// @JoinColumn(name="prodId")

private Set<Product> product;

@ManyToMany(cascade = CascadeType.ALL,fetch = FetchType.LAZY)

@JoinTable(name = "category\_customer\_table", joinColumns = {

@JoinColumn(name = "categoryID", nullable = false, updatable = false)},

inverseJoinColumns = {

@JoinColumn(name ="custId")

})

private Set<Customer> customer = new HashSet<Customer>();

public long getCategoryId() {

return categoryId;

}

public String getCategoryName() {

return categoryName;

}

public void setCategoryName(String categoryName) {

this.categoryName = categoryName;

}

public void setCategoryId(long categoryId) {

this.categoryId = categoryId;

}

public String getCategoryDescript() {

return categoryDescript;

}

public void setCategoryDescript(String categoryDescript) {

this.categoryDescript = categoryDescript;

}

public Set<Product> getProduct() {

return product;

}

public void setProduct(Set<Product> product) {

this.product = product;

}

}

Customer

package com.myproject.product.pojo;

import java.util.Set;

import javax.persistence.CascadeType;

import javax.persistence.Entity;

import javax.persistence.FetchType;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.JoinColumn;

import javax.persistence.OneToMany;

import javax.persistence.OneToOne;

import javax.persistence.ManyToMany;

import javax.persistence.NamedQuery;

@NamedQuery(name="authenticateCustomer", query="from Customer where custUserName=:username")

@Entity

public class Customer {

public Customer() {

}

@Id

@GeneratedValue(strategy=GenerationType.IDENTITY)

private long custId;

private String firstName;

private String lastName;

private int visitCount;

private String emailId;

private String custUserName;

private String password;

@OneToOne(fetch = FetchType.LAZY, cascade = CascadeType.ALL)

@JoinColumn(name="addrssId")

private Address customerAddress;

//@OneToMany(mappedBy="customerPhone",fetch = FetchType.LAZY, cascade = CascadeType.ALL)

//private Set<Phone> contact;

@ManyToMany(mappedBy="customer",fetch = FetchType.LAZY, cascade = CascadeType.ALL)

private Set<Category> category;

public long getCustId() {

return custId;

}

public void setCustId(long custId) {

this.custId = custId;

}

public String getEmailId() {

return emailId;

}

public void setEmailId(String emailId) {

this.emailId = emailId;

}

public String getCustUserName() {

return custUserName;

}

public void setCustUserName(String custUserName) {

this.custUserName = custUserName;

}

public String getPassword() {

return password;

}

public void setPassword(String password) {

this.password = password;

}

public String getFirstName() {

return firstName;

}

public void setFirstName(String firstName) {

this.firstName = firstName;

}

public String getLastName() {

return lastName;

}

public void setLastName(String lastName) {

this.lastName = lastName;

}

public int getVisitCount() {

return visitCount;

}

public void setVisitCount(int visitCount) {

this.visitCount = visitCount;

}

public Address getCustomerAddress() {

return customerAddress;

}

public void setCustomerAddress(Address customerAddress) {

this.customerAddress = customerAddress;

}

/\*public Set<Phone> getContact() {

return contact;

}

public void setContact(Set<Phone> contact) {

this.contact = contact;

}\*/

public Set<Category> getCategory() {

return category;

}

public void setCategory(Set<Category> category) {

this.category = category;

}

}

Feedback

package com.myproject.product.pojo;

import javax.persistence.CascadeType;

import javax.persistence.Entity;

import javax.persistence.FetchType;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.JoinColumn;

import javax.persistence.ManyToOne;

@Entity

public class Feedback {

public Feedback() {

}

@Id

@GeneratedValue(strategy=GenerationType.IDENTITY)

private long feedId;

private String feedBack;

@ManyToOne(fetch = FetchType.EAGER, cascade = CascadeType.ALL)

@JoinColumn(name="prodId")

private Product product;

@ManyToOne(fetch = FetchType.EAGER, cascade = CascadeType.ALL)

@JoinColumn(name="custId")

private Customer customer;

public long getFeedId() {

return feedId;

}

public void setFeedId(long feedId) {

this.feedId = feedId;

}

public String getFeedBack() {

return feedBack;

}

public void setFeedBack(String feedBack) {

this.feedBack = feedBack;

}

public Product getProduct() {

return product;

}

public void setProduct(Product product) {

this.product = product;

}

public Customer getCustomer() {

return customer;

}

public void setCustomer(Customer customer) {

this.customer = customer;

}

}

Merchant

package com.myproject.product.pojo;

import java.util.HashSet;

import java.util.Set;

import javax.persistence.CascadeType;

import javax.persistence.Entity;

import javax.persistence.FetchType;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.JoinColumn;

import javax.persistence.ManyToMany;

import javax.persistence.NamedQuery;

import javax.persistence.OneToMany;

import javax.persistence.OneToOne;

@NamedQuery(name="authenticateMerchant", query="from Merchant where merchantUserName=:username")

@Entity

public class Merchant {

public Merchant() {

}

@Id

@GeneratedValue(strategy=GenerationType.IDENTITY)

private long merchantId;

private String firstName;

private String lastName;

private String emailId;

private String merchantUserName;

private String password;

@ManyToMany(mappedBy="merchant",fetch = FetchType.LAZY, cascade = CascadeType.ALL)

private Set<Product> product = new HashSet<Product>();;

@OneToOne(fetch = FetchType.LAZY, cascade = CascadeType.ALL)

@JoinColumn(name="addrssId")

private Address merchantAddress;

//@OneToMany(mappedBy="merchantPhone",fetch = FetchType.LAZY, cascade = CascadeType.ALL)

//private Set<Phone> contact;

public long getMerchantId() {

return merchantId;

}

public void setMerchantId(long merchantId) {

this.merchantId = merchantId;

}

public String getFirstName() {

return firstName;

}

public void setFirstName(String firstName) {

this.firstName = firstName;

}

public String getLastName() {

return lastName;

}

public String getEmailId() {

return emailId;

}

public void setEmailId(String emailId) {

this.emailId = emailId;

}

public String getMerchantUserName() {

return merchantUserName;

}

public void setMerchantUserName(String merchantUserName) {

this.merchantUserName = merchantUserName;

}

public String getPassword() {

return password;

}

public void setPassword(String password) {

this.password = password;

}

public void setLastName(String lastName) {

this.lastName = lastName;

}

public Set<Product> getProduct() {

return product;

}

public void setProduct(Set<Product> product) {

this.product = product;

}

public Address getMerchantAddress() {

return merchantAddress;

}

public void setMerchantAddress(Address merchantAddress) {

this.merchantAddress = merchantAddress;

}

/\*public Set<Phone> getContact() {

return contact;

}

public void setContact(Set<Phone> contact) {

this.contact = contact;

}\*/

}

Product

package com.myproject.product.pojo;

import java.util.HashSet;

import java.util.Set;

import javax.persistence.CascadeType;

import javax.persistence.Entity;

import javax.persistence.FetchType;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.JoinColumn;

import javax.persistence.JoinTable;

import javax.persistence.ManyToMany;

import javax.persistence.ManyToOne;

import javax.persistence.NamedQueries;

import javax.persistence.NamedQuery;

import javax.persistence.OneToMany;

import javax.persistence.Transient;

import org.springframework.web.multipart.MultipartFile;

@Entity

public class Product {

public Product() {

}

@Id

@GeneratedValue(strategy=GenerationType.IDENTITY)

private long prodId;

private String prodName;

private String prodPrice;

private String prodImage;

@Transient

private MultipartFile photo;

private String prodDecript;

private int prodQty;

@ManyToOne(fetch = FetchType.LAZY, cascade = CascadeType.ALL)

@JoinColumn(name="categoryId")

private Category category;

@ManyToMany(cascade = CascadeType.ALL,fetch = FetchType.LAZY)

@JoinTable(name = "Product\_Merchant\_table", joinColumns = {

@JoinColumn(name = "prodId", nullable = false, updatable = false)

},

inverseJoinColumns = {

@JoinColumn(name ="merchantId")

})

private Set<Merchant> merchant = new HashSet<Merchant>();

@OneToMany(mappedBy="product",fetch = FetchType.LAZY, cascade = CascadeType.ALL)

private Set<Feedback> feedback;

private int userViewCount;

public long getProdId() {

return prodId;

}

public void setProdId(long prodId) {

this.prodId = prodId;

}

public int getProdQty() {

return prodQty;

}

public void setProdQty(int prodQty) {

this.prodQty = prodQty;

}

public String getProdName() {

return prodName;

}

public void setProdName(String prodName) {

this.prodName = prodName;

}

public String getProdPrice() {

return prodPrice;

}

public void setProdPrice(String prodPrice) {

this.prodPrice = prodPrice;

}

public String getProdImage() {

return prodImage;

}

public void setProdImage(String prodImage) {

this.prodImage = prodImage;

}

public String getProdDecript() {

return prodDecript;

}

public void setProdDecript(String prodDecript) {

this.prodDecript = prodDecript;

}

public Category getCategory() {

return category;

}

public void setCategory(Category category) {

this.category = category;

}

public Set<Merchant> getMerchant() {

return merchant;

}

public void setMerchant(Set<Merchant> merchant) {

this.merchant = merchant;

}

public Set<Feedback> getFeedback() {

return feedback;

}

public void setFeedback(Set<Feedback> feedback) {

this.feedback = feedback;

}

public int getUserViewCount() {

return userViewCount;

}

public void setUserViewCount(int userViewCount) {

this.userViewCount = userViewCount;

}

public MultipartFile getPhoto() {

return photo;

}

public void setPhoto(MultipartFile photo) {

this.photo = photo;

}

}

## Exception class

1. Feedback class

**package** com.myproject.product.exception;

**public** **class** FeedbackException **extends** Exception{

**public** FeedbackException(String message)

{

**super**("FeedbackException-"+message);

}

**public** FeedbackException(String message, Throwable cause)

{

**super**("FeedbackException-"+message,cause);

}

}

1. Product Exception

**package** com.myproject.product.exception;

**public** **class** ProductException **extends** Exception {

**public** ProductException(String message)

{

**super**("ProductException-"+message);

}

**public** ProductException(String message, Throwable cause)

{

**super**("ProductException-"+message,cause);

}

}

1. UserException

**package** com.myproject.product.exception;

**public** **class** UserException **extends** Exception {

**public** UserException(String message)

{

**super**("UserException-"+message);

}

**public** UserException(String message, Throwable cause)

{

**super**("UserException-"+message,cause);

}

}

## DAO class

1. Main Dao

package com.myproject.product.dao;

import java.util.logging.Level;

import java.util.logging.Logger;

import org.hibernate.HibernateException;

import org.hibernate.Session;

import org.hibernate.SessionFactory;

import org.hibernate.cfg.Configuration;

public class DAO {

private static final Logger log = Logger.getAnonymousLogger();

private static final ThreadLocal sessionThread = new ThreadLocal();

private static final SessionFactory sessionFactory = new Configuration().configure("hibernate.cfg.xml").buildSessionFactory();

protected DAO() {

}

public static Session getSession()

{

Session session = (Session) DAO.sessionThread.get();

if (session == null)

{

session = sessionFactory.openSession();

DAO.sessionThread.set(session);

}

return session;

}

protected void begin() {

getSession().beginTransaction();

}

protected void commit() {

getSession().getTransaction().commit();

}

protected void rollback() {

try {

getSession().getTransaction().rollback();

} catch (HibernateException e) {

log.log(Level.WARNING, "Cannot rollback", e);

}

try {

getSession().close();

} catch (HibernateException e) {

log.log(Level.WARNING, "Cannot close", e);

}

DAO.sessionThread.set(null);

}

public static void close() {

getSession().close();

DAO.sessionThread.set(null);

}

}

UserDao

package com.myproject.product.dao;

import java.util.List;

import org.hibernate.Criteria;

import org.hibernate.HibernateException;

import org.hibernate.Query;

import org.hibernate.criterion.Criterion;

import org.hibernate.criterion.Disjunction;

import org.hibernate.criterion.LogicalExpression;

import org.hibernate.criterion.Restrictions;

import org.mindrot.jbcrypt.BCrypt;

import com.myproject.product.pojo.\*;

import com.myproject.product.exception.UserException;

public class UserDAO extends DAO {

//Declaring a default constructor

public UserDAO(){

}

//Admin registration

public int registerAdmin(Admin admin) throws UserException {

int register = 0;

try {

begin();

getSession().save(admin);

commit();

register =1;

return register;

} catch (HibernateException e) {

rollback();

throw new UserException("Exception while creating user: " + e.getMessage());

}

}

//Customer registration

public int registerCustomer(Customer customer) throws UserException {

int register = 0;

try {

begin();

getSession().save(customer);

commit();

register =1;

return register;

} catch (HibernateException e) {

rollback();

throw new UserException("Exception while creating user: " + e.getMessage());

}

}

//Merchant registration

public int registerMerchant(Merchant merchant) throws UserException {

int register = 0;

try {

begin();

getSession().save(merchant);

commit();

register =1;

return register;

} catch (HibernateException e) {

rollback();

throw new UserException("Exception while creating user: " + e.getMessage());

}

}

//Authenticate User

public String authenticateUser(String username, String password) throws UserException{

String result = "notAUser";

begin();

if (username.startsWith("merc")) {

Merchant merch = null;

Query query = getSession().getNamedQuery("authenticateMerchant");

query.setParameter("username", username);

//query.setParameter("password", password);

merch = (Merchant) query.uniqueResult();

if (merch == null || !BCrypt.checkpw(password, merch.getPassword()))

return result;

else {

result = username;

}

}

else if (username.startsWith("cust")) {

Query query = getSession().getNamedQuery("authenticateCustomer");

query.setParameter("username", username);

//query.setParameter("password", password);

Customer cust = (Customer) query.uniqueResult();

if (cust == null || !BCrypt.checkpw(password, cust.getPassword()))

return result;

else {

result = username;

}

}

else {

Query query = getSession().getNamedQuery("authenticateAdmin");

query.setParameter("username", username);

query.setParameter("password", password);

Admin admin = (Admin) query.uniqueResult();

if (admin == null)

return result;

else {

result = username;

}

}

commit();

return result;

}

//Get all list of users

public List<Customer> custList() throws UserException {

try {

begin();

Query q = getSession().createQuery("from Customer");

List<Customer> list = q.list();

commit();

return list;

} catch (HibernateException e) {

rollback();

throw new UserException("Could not list the Customer", e);

}

}

//Get all list of merchant

public List<Merchant> merchList() throws UserException {

try {

begin();

Query q = getSession().createQuery("from Merchant");

List<Merchant> list = q.list();

commit();

return list;

} catch (HibernateException e) {

rollback();

throw new UserException("Could not list the Merchant", e);

}

}

//Get detail of a merchant

public List<Merchant> merchDetails(String merchName) throws UserException {

try {

begin();

Criteria mercDetail = getSession().createCriteria(Merchant.class);

Criterion merchFName = Restrictions.like("firstName",merchName);

Criterion merchLName = Restrictions.like("lastName",merchName);

Criterion merchUserName = Restrictions.like("merchantUserName",merchName);

Disjunction orExp = Restrictions.disjunction();

orExp.add(merchFName);orExp.add(merchLName);orExp.add(merchUserName);

mercDetail.add(orExp);

List<Merchant> list = mercDetail.list();

commit();

return list;

} catch (HibernateException e) {

rollback();

throw new UserException("Could not list the Merchant", e);

}

}

//Get detail of a users

public List<Customer> userDetails(String userName) throws UserException {

try {

begin();

Criteria userDetail = getSession().createCriteria(Customer.class);

Criterion custFName = Restrictions.like("firstName",userName);

Criterion custLName = Restrictions.like("lastName",userName);

Criterion custUserName = Restrictions.like("custUserName",userName);

Disjunction orExp = Restrictions.disjunction();

orExp.add(custFName);orExp.add(custLName);orExp.add(custUserName);

userDetail.add(orExp);

List<Customer> list = userDetail.list();

commit();

return list;

} catch (HibernateException e) {

rollback();

throw new UserException("Could not list the Customer", e);

}

}

//Get customer details

public Customer getCustDetail(String custName) throws UserException {

try {

begin();

Criteria crit = getSession().createCriteria(Customer.class);

Criterion userName = Restrictions.eq("custUserName",custName);

crit.add(userName);

Customer cust = (Customer) crit.uniqueResult();

commit();

return cust;

} catch (HibernateException e) {

rollback();

throw new UserException("Could not list the Customer", e);

}

}

}

Category Dao

package com.myproject.product.dao;

import java.util.List;

import org.hibernate.Criteria;

import org.hibernate.HibernateException;

import org.hibernate.Query;

import org.hibernate.criterion.Criterion;

import org.hibernate.criterion.Disjunction;

import org.hibernate.criterion.Order;

import org.hibernate.criterion.Restrictions;

import com.myproject.product.pojo.\*;

import com.myproject.product.exception.FeedbackException;

import com.myproject.product.exception.ProductException;

import com.myproject.product.exception.UserException;

public class CategoryDAO extends DAO{

public CategoryDAO(){

}

//Add new category

public int createCategory(Category cat) throws ProductException {

int register = 0;

try {

begin();

getSession().save(cat);

commit();

register =1;

return register;

} catch (HibernateException e) {

rollback();

throw new ProductException("Exception while creating user: " + e.getMessage());

}

}

//Add new Product

public int createProduct(Product prod) throws ProductException {

int register = 0;

try {

begin();

getSession().merge(prod);

commit();

register =1;

return register;

} catch (HibernateException e) {

rollback();

throw new ProductException("Exception while creating user: " + e.getMessage());

}

}

//List all category

public List<Category> catList() throws ProductException {

try {

begin();

Query q = getSession().createQuery("from Category");

List<Category> list = q.list();

commit();

return list;

} catch (HibernateException e) {

rollback();

throw new ProductException("Could not list the Category", e);

}

}

//List all products under a category

public List<Product> prodList(String catName) throws ProductException {

try {

begin();

Criteria prodList = getSession().createCriteria(Product.class);

Criteria catList = prodList.createCriteria("category");

catList.add(Restrictions.eq("categoryName",catName));

prodList.addOrder(Order.asc("prodPrice"));

prodList.addOrder(Order.desc("userViewCount"));

List<Product> result = prodList.list();

commit();

return result;

} catch (HibernateException e) {

rollback();

throw new ProductException("Could not list the Catrgory", e);

}

}

//Get detail of a product

public List<Product> productDetails(String prodName) throws ProductException {

try {

begin();

Criteria prodDetail = getSession().createCriteria(Product.class);

Criterion productName = Restrictions.like("prodName",prodName);

prodDetail.add(productName);

prodDetail.setMaxResults(10);

List<Product> list = prodDetail.list();

commit();

return list;

} catch (HibernateException e) {

rollback();

throw new ProductException("Could not list the Product", e);

}

}

//Update a product

public int updateProduct(Product prod) throws ProductException {

int register = 0;

try {

begin();

getSession().merge(prod);

commit();

register =1;

return register;

} catch (HibernateException e) {

rollback();

throw new ProductException("Exception while creating user: " + e.getMessage());

}

}

//Get description of one product

public Product productDescript(String prodName) throws ProductException {

try {

begin();

Criteria prodDetail = getSession().createCriteria(Product.class);

Criterion productName = Restrictions.eq("prodName",prodName);

prodDetail.add(productName);

//prodDetail.setMaxResults(10);

Product p = (Product) prodDetail.uniqueResult();

commit();

return p;

} catch (HibernateException e) {

rollback();

throw new ProductException("Could not list the Product", e);

}

}

//Add feedback to a product

public int addFeedBack(Feedback fd) throws ProductException {

int register = 0;

try {

begin();

getSession().merge(fd);

commit();

register =1;

return register;

} catch (HibernateException e) {

rollback();

throw new ProductException("Exception while creating user: " + e.getMessage());

}

}

//Get feedback of a product

public List<Feedback> getFeedback(String selectedProduct) throws FeedbackException {

try {

begin();

Criteria feedList = getSession().createCriteria(Feedback.class);

Criteria prod = feedList.createCriteria("product");

prod.add(Restrictions.eq("prodName",selectedProduct));

List<Feedback> result = feedList.list();

commit();

return result;

} catch (HibernateException e) {

rollback();

throw new FeedbackException("Exception while creating user: " + e.getMessage());

}

}

//List all Products

public List<Product> allProdList() throws ProductException {

try {

begin();

Query q = getSession().createQuery("from Product");

List<Product> list = q.list();

commit();

return list;

} catch (HibernateException e) {

rollback();

throw new ProductException("Could not list the Merchant", e);

}

}

//List all Merchants

public List<Merchant> mercList() throws UserException {

try {

begin();

Query q = getSession().createQuery("from Merchant");

List<Merchant> list = q.list();

commit();

return list;

} catch (HibernateException e) {

rollback();

throw new UserException("Could not list the Merchant", e);

}

}

}