# E-COMMERCE WEB APPLICATION

**Project Abstract:**

Building a one stop e-commerce web application, wherein sellers can sign up and post their products. Customers can also sign-up as well and order the products of their choice and on successful order placement of a product, confirmation is sent on the email to the user.

**Roles included:**

* Admin – Manage sellers on the website and product categories.
* Seller – Manage their products on the website to be sold and their quantity based on popularity of product.
* Customer – Buy products of their choice from the web app. Also keep items in cart till their approval.

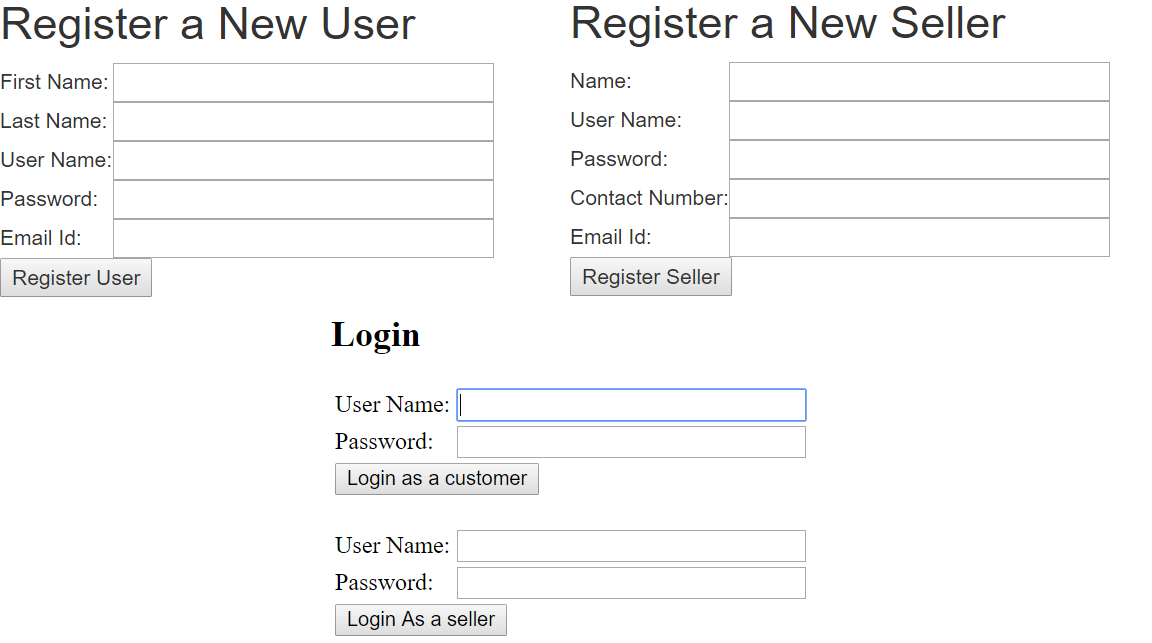
**Scope:**

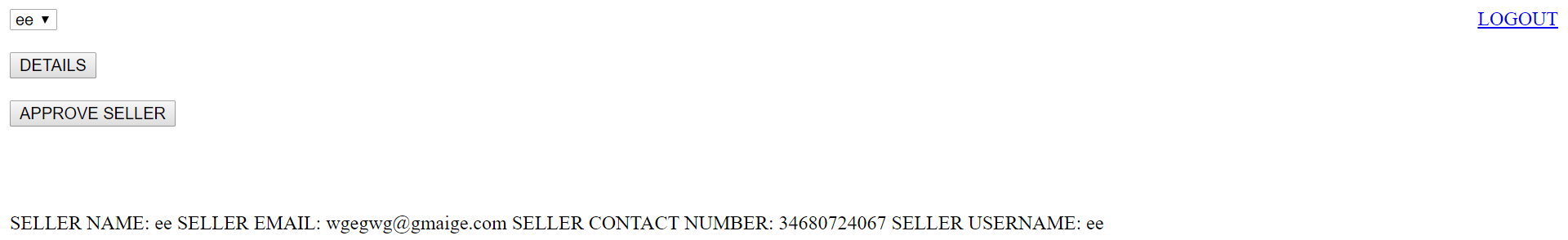
Admin can validate the seller’s authenticity and manage categories and well as seller’s based on their reviews and seller’s popularity based on its products on the web application. Sellers can post different products and manage their inventory based on product’s popularity. Customers on the other side can buy products, and keep orders in their cart till they have made up their mind. A confirmation email will be sent to the user on signing up and on successfully ordering an item from the web application. Can download all pdf’s for previous orders as well.

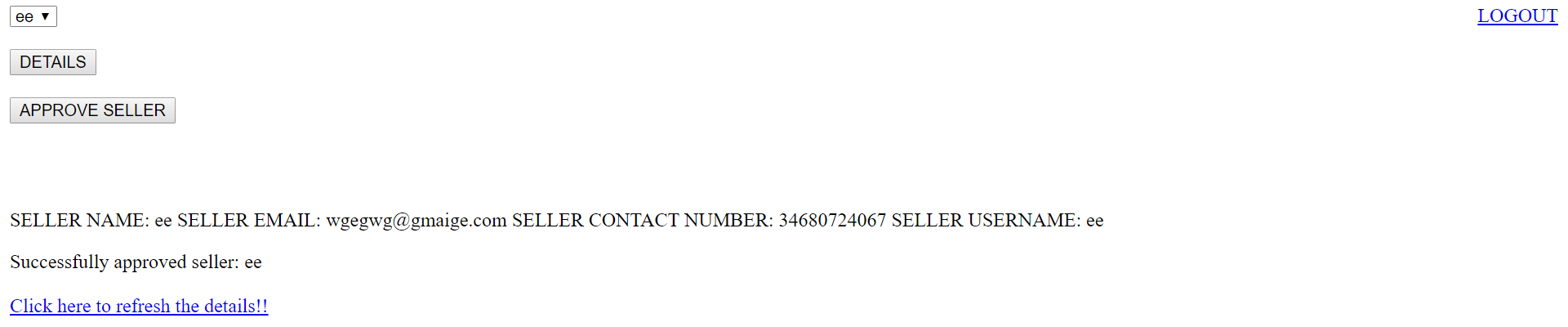
**SUMMARY OF DIFFERENT FUNCTIONALITIES:-**

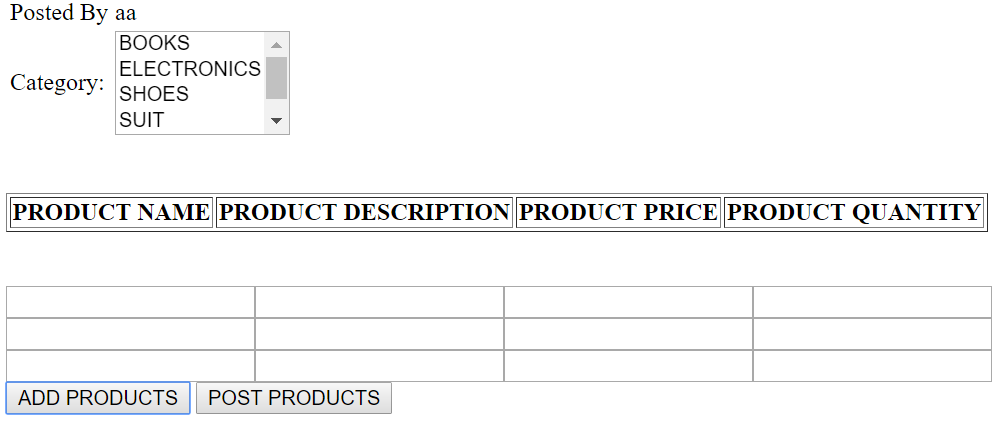
1. CUSTOMERS AND SELLERS CAN SIGN UP ON APPLICATION
2. SIGN UP **EMAILS** ARE SENT TO BOTH
3. SELLER CANNOT LOGIN UNTIL APPROVED BY ADMIN(**AJAX USED**), ADMIN ALSO POSTS THE PRODUCTS CATEGORIES
4. SELLER CAN POST MULTIPLE PRODUCTS(**AJAX USED**) AT A TIME AND FOR MULTIPLE CATEGORIES
5. SELLER CAN PERFORM **CRUD OPERATIONS** ON THE PRODUCTS(**AJAX USED**)
6. USER HAS A BOOTSTRAP HTML5 CSS3 PAGE WITH JAVASCRIPT DESIGNED WHERE HE/SHE CAN VIEW PRODUCTS,ADD PRODUCTS TO CART, BUY THEM, DOWNLOAD ORDER PDF, CHECK ORDER HISTORY AND **DOWNLOAD ANY PREVIOUS ORDER PDF**, SEARCH FOR PRODUCTS(**AJAX USED**) ETC
7. VALIDATIONS FOR ALL INPUT FIELDS ARE DONE USING **BOOTSTRAP VALIDATOR,** AND INBUILT VALIDATOR FOR SPRING FORM
8. ALL ENTITIES HAVE **HIBERNATE ANNOTATIONS** AND **DAO PATTERN** USED FOR SESSION
9. AUTHENTICATION AND AUTHORIZATION PERFORMED USING **INTERCEPTOR**

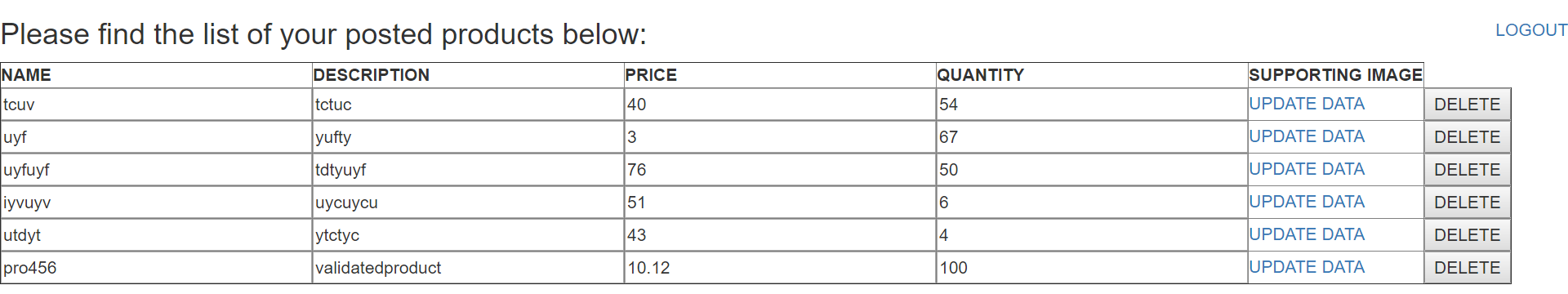
## KEY SCREENSHOTS

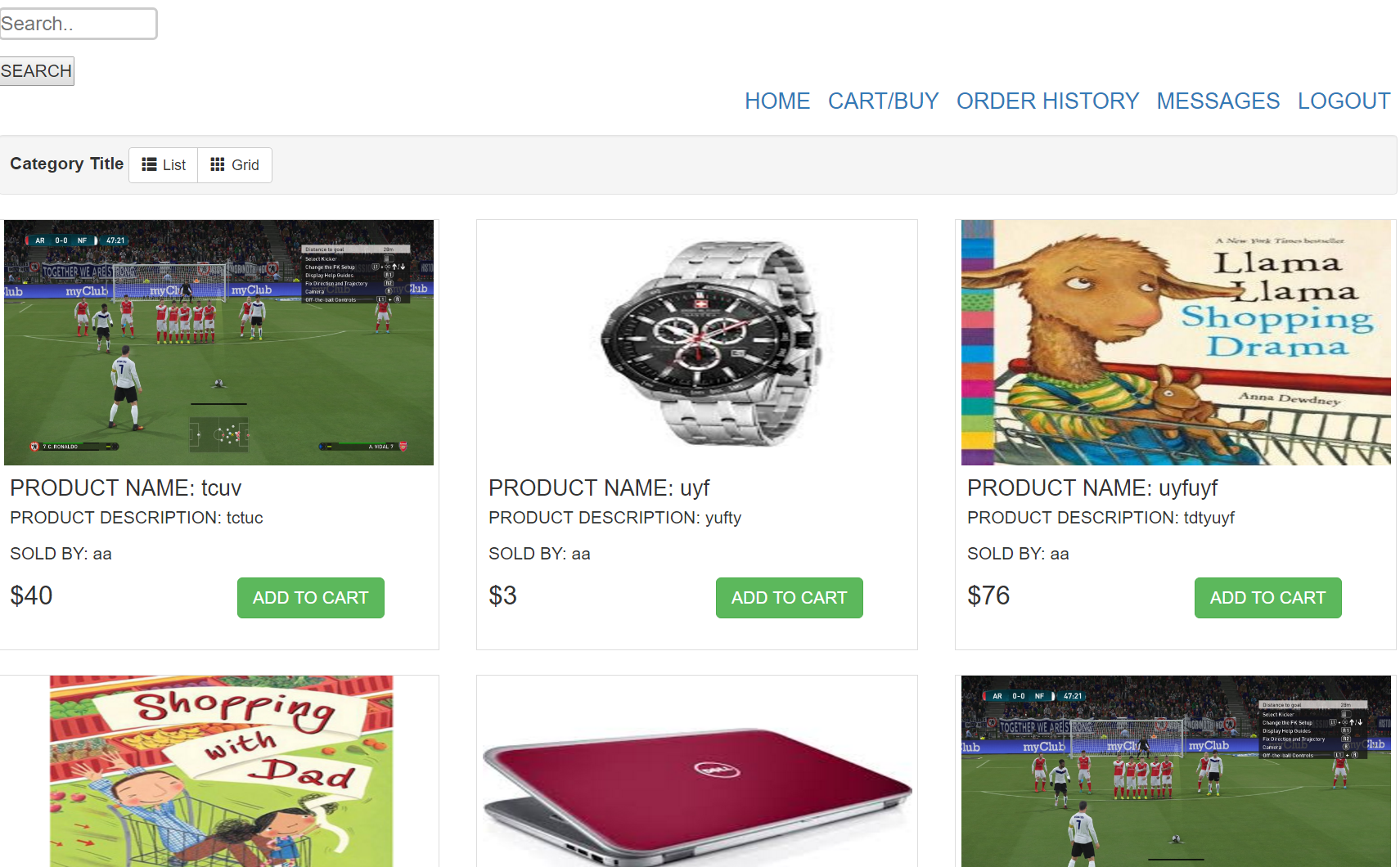


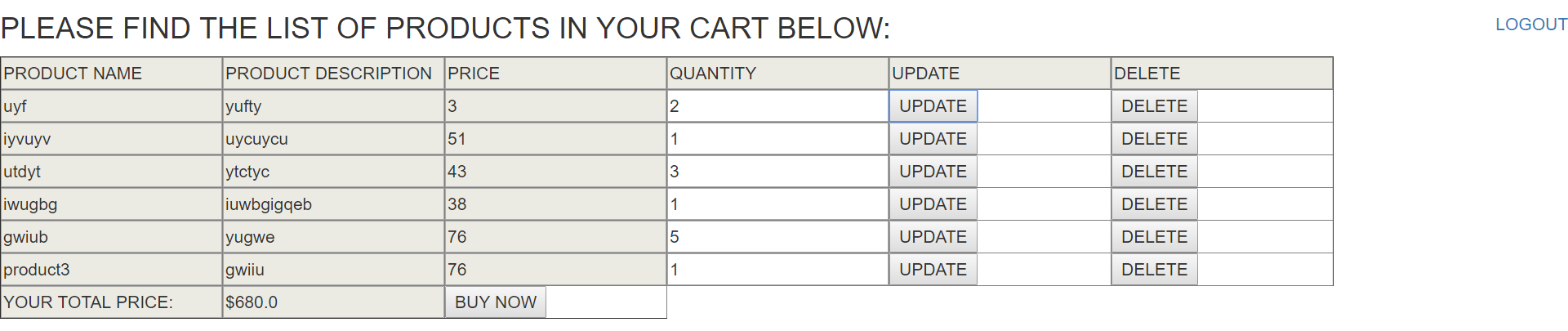


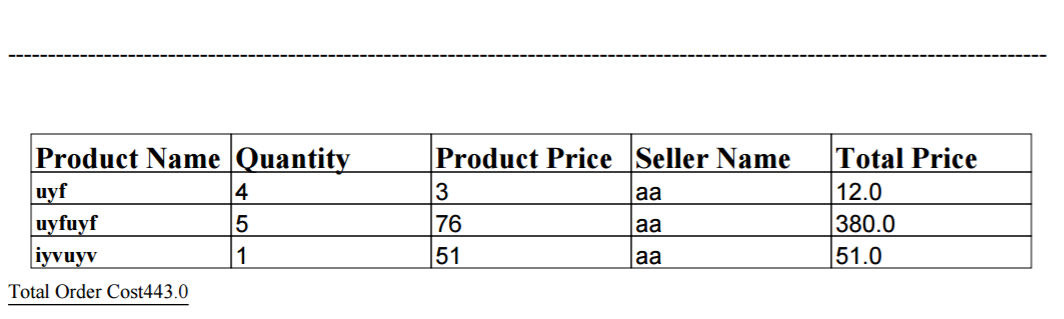
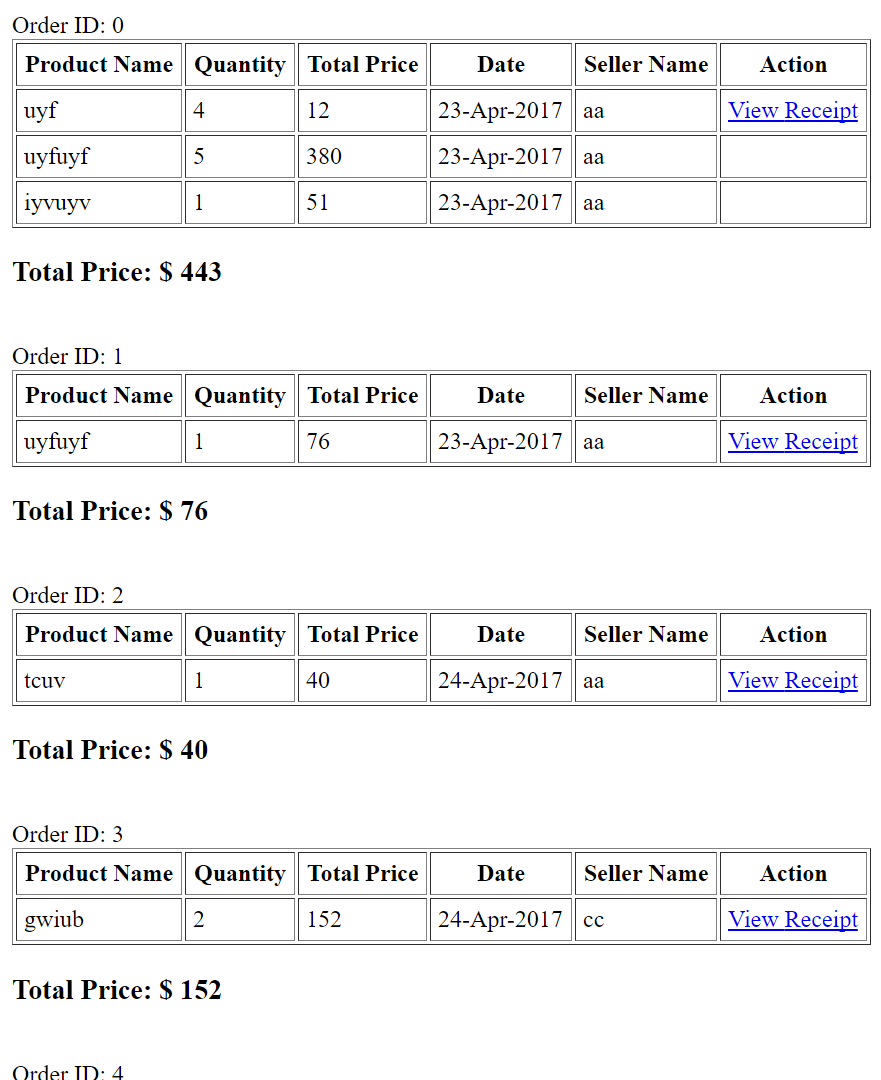


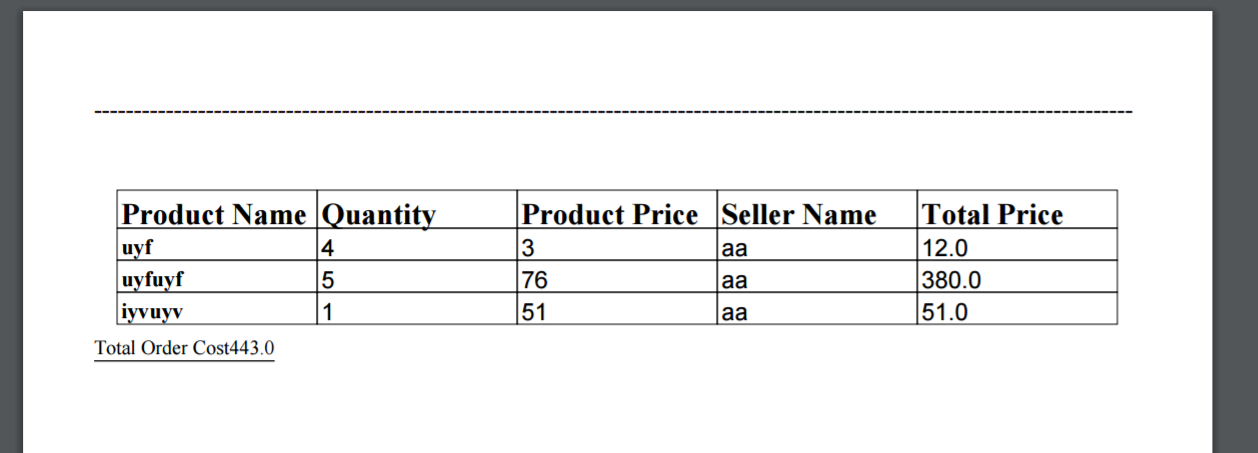












## CONTROLLER CODES

1. ADMIN CONTROLLER

package com.me.controller;

import java.util.ArrayList;

import javax.servlet.http.HttpServletRequest;

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

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

import org.springframework.stereotype.Controller;

import org.springframework.web.bind.WebDataBinder;

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

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

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

import org.springframework.web.servlet.ModelAndView;

import com.me.dao.AdminDAO;

import com.me.dao.SellerDAO;

import com.me.pojo.Seller;

import com.me.validator.AdminValidator;

import com.me.validator.SellerValidator;

@Controller

@RequestMapping("/admin/\*")

public class AdminController {

@Autowired

@Qualifier("adminDao")

AdminDAO adminDao;

@Autowired

@Qualifier("adminValidator")

AdminValidator adminvalidator;

@InitBinder

private void initBinder(WebDataBinder binder) {

binder.setValidator(adminvalidator);

}

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

protected ModelAndView ValidateSellers() throws Exception {

System.out.println("Validate Sellers by Admin clicked!");

ArrayList<Seller> sellerList = adminDao.getNewSellerList();

return new ModelAndView("validate-sellers","sellerlist",sellerList);

}

@RequestMapping(value = "/admin/admin-home.htm", method = RequestMethod.GET)

protected ModelAndView AdminHomePage(HttpServletRequest request) throws Exception {

request.getSession().setAttribute("role", "admin");

return new ModelAndView("admin-home");

}

}

1. AJAX DATA PROVIDER

package com.me.controller;

import java.util.ArrayList;

import java.util.List;

import javax.servlet.http.HttpServletRequest;

import org.hibernate.HibernateException;

import org.hibernate.Query;

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

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

import org.springframework.stereotype.Controller;

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

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

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

import com.fasterxml.jackson.core.JsonProcessingException;

import com.fasterxml.jackson.databind.ObjectMapper;

import com.me.dao.DAO;

import com.me.dao.ProductDAO;

import com.me.dao.UserDAO;

import com.me.exception.ProductException;

import com.me.exception.UserException;

import com.me.pojo.Cart;

import com.me.pojo.Product;

import com.me.pojo.Seller;

import com.me.pojo.User;

/\*\*

\* Servlet implementation class AjaxDataProvider

\*/

@Controller

public class AjaxDataProvider extends DAO {

public AjaxDataProvider() {

}

@Autowired

@Qualifier("userDao")

UserDAO userDAO;

@Autowired

@Qualifier("productDao")

ProductDAO productDAO;

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

@ResponseBody

public String getAttr(HttpServletRequest request) {

ObjectMapper mapper = new ObjectMapper();

Long sellername = Long.parseLong(request.getParameter("seller"));

System.out.println("\*\*\*\*RETRIEVING FROM AJAX: " + sellername);

ArrayList<Seller> sellerList;

Seller seller1 = new Seller();

System.out.println();

try{

String sql = "from Seller where id = :someAttr";

Query query = getSession().createQuery(sql);

long sellername1 = sellername;

System.out.println(String.valueOf(sellername1));

query.setLong("someAttr", sellername1);

System.out.println("query for selecting seller details from db ran in ajax controller!!");

sellerList = (ArrayList<Seller>)query.list();

for(Seller seller:sellerList){

seller1 = seller;

}

}catch (HibernateException e) {

rollback();

System.out.println("Error retreiving seller list - AJAX CONTROLLER!!");

}

for(Product p : seller1.getProducts()){

System.out.println("Products list!!");

System.out.println(p.getName());

}

//String jsonInString = gson.toJson(seller1);

String jsonInString="";

try {

jsonInString = mapper.writeValueAsString(seller1);

} catch (JsonProcessingException e) {

// TODO Auto-generated catch block

System.out.println("Exception converting object to JSON:\t"+e.getMessage());

}

System.out.println("JSON OUTPUT:"+jsonInString);

System.out.println("Seller name retrieved is:\t"+seller1.getName());

return jsonInString;

}

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

@ResponseBody

public String getSellerApproved(HttpServletRequest request) {

//Gson gson = new Gson();

ObjectMapper mapper = new ObjectMapper();

String sellername = request.getParameter("seller");

System.out.println("\*\*\* RETRIEVING FROM AJAX: " + sellername);

ArrayList<Seller> sellerList;

Seller seller1 = new Seller();

try{

begin();

String sql = "update Seller set status = :status where id = :id";

Query query = getSession().createQuery(sql);

long sellername1 = Long.parseLong(sellername);

query.setLong("id", sellername1);

query.setString("status", "Approved");

query.executeUpdate();

System.out.println("query for updating seller details in db ran in ajax controller!!");

commit();

String sql1 = "from Seller where id = :someAttr";

Query query1 = getSession().createQuery(sql1);

//long sellername1 = Long.parseLong(sellername);

System.out.println(sellername1);

query1.setLong("someAttr", sellername1);

System.out.println("query for selecting seller details from db ran in ajax controller!!");

sellerList = (ArrayList<Seller>)query1.list();

for(Seller seller:sellerList){

seller1 = seller;

}

}catch (HibernateException e) {

rollback();

System.out.println("Error retreiving seller list - AJAX CONTROLLER!!");

}

System.out.println("Seller name retrieved is:\t"+seller1.getName());

//String jsonInString = gson.toJson(seller1);

String jsonInString="";

try {

jsonInString = mapper.writeValueAsString(seller1);

} catch (JsonProcessingException e) {

// TODO Auto-generated catch block

System.out.println("Exception converting object to JSON:\t"+e.getMessage());

}

return jsonInString;

}

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

@ResponseBody

public String deleteProductFromDb(HttpServletRequest request) {

Long productid = Long.parseLong(request.getParameter("product"));

System.out.println("\*\*\*\* PRODUCT-ID RETRIEVING FROM AJAX: " + productid);

try{

begin();

String sql1 = "delete from category\_product\_table where productID =:productid";

Query query1 = getSession().createSQLQuery(sql1);

query1.setLong("productid", productid);

int result = query1.executeUpdate();

String sql = "delete Product where productID =:productid";

Query query = getSession().createQuery(sql);

query.setLong("productid", productid);

int result1 = query.executeUpdate();

System.out.println("Resultant Rows:\t"+result);

System.out.println("Resultant Rows:\t"+result1);

commit();

close();

System.out.println("query for deleting product from db ran in ajax controller!!");

}catch (HibernateException e) {

rollback();

System.out.println("Error deleting products! - AJAX CONTROLLER!!");

}

return "success";

}

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

@ResponseBody

public String addtoCart(HttpServletRequest request) {

System.out.println("INSIDE ADDTOCART CONTROLLER!");

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

System.out.println(u.getFirstName());

Long userid = u.getPersonID();

Long pid = Long.parseLong(request.getParameter("productid"));

System.out.println("\*\*\*\*RETRIEVING FROM SESSION: " +userid);

try{

begin();

System.out.println("inside CARTDAO - AJAXDATAPROVIDER!!!");

Query q2 = getSession().createQuery("from Cart where productID= :pid and userID= :uid");

q2.setLong("pid", pid);

q2.setLong("uid", userid);

Cart cart = (Cart) q2.uniqueResult();

if(cart!=null){

begin();

int quantity = cart.getQuantity();

quantity = quantity + 1;

Query q3 = getSession().createQuery("update Cart set quantity=:quantity where productID= :pid and userID= :uid");

q3.setLong("pid", pid);

q3.setInteger("quantity", quantity);

q3.setLong("uid", userid);

int result = q3.executeUpdate();

System.out.println("result of update cart:\t"+result);

commit();

}else{

Query q1 = getSession().createQuery("from Product where productID= :pid");

q1.setLong("pid", pid);

Product product = (Product) q1.uniqueResult();

System.out.println("PRODUCT RETRIEVED IS:\t"+product.getName());

System.out.println("USER RETRIEVED IS:\t"+u.getFirstName());

Cart cart1 = new Cart();

cart1.setUser(u);

cart1.setProduct(product);

cart1.setQuantity(1);

getSession().save(cart1);

commit();

}

}catch (HibernateException e) {

rollback();

System.out.println("Error for CART - AJAX CONTROLLER!!");

}

return "incart";

}

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

@ResponseBody

public String updateproductlist(HttpServletRequest request) {

String text = request.getParameter("text");

System.out.println("Searched value is:\t"+text);

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

try {

productList = productDAO.get(text);

} catch (ProductException e1) {

System.out.println("+++++++"+e1.getMessage());

}

System.out.println("SIZE OF RETREIVED LIST:\t"+productList.size());

try{

request.getSession().setAttribute("productlist", productList);

}catch (HibernateException e) {

rollback();

System.out.println("Error retreiving seller list - AJAX CONTROLLER!!");

}

System.out.println("STORED IN SESSION:\t"+productList);

String jsonInString="abc";

return jsonInString;

}

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

@ResponseBody

public String updateproductlistfilter(HttpServletRequest request) {

String text = request.getParameter("text");

System.out.println("Searched value is:\t"+text);

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

try {

productList = productDAO.list();

} catch (ProductException e1) {

System.out.println("+++++++"+e1.getMessage());

}

System.out.println("SIZE OF RETREIVED LIST:\t"+productList.size());

try{

request.getSession().setAttribute("productlist", productList);

}catch (HibernateException e) {

rollback();

System.out.println("Error retreiving product list filter - AJAX CONTROLLER!!");

}

//System.out.println("STORED IN SESSION:\t"+productList);

String jsonInString="abc";

return jsonInString;

}

}

1. CATEGORY CONTROLLER

package com.me.controller;

import javax.servlet.http.HttpServletRequest;

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

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

import org.springframework.stereotype.Controller;

import org.springframework.validation.BindingResult;

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.RequestMapping;

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

import org.springframework.web.servlet.ModelAndView;

import com.me.dao.CategoryDAO;

import com.me.pojo.Category;

import com.me.validator.CategoryValidator;

import com.me.exception.CategoryException;

@Controller

public class CategoryController {

@Autowired

@Qualifier("categoryValidator")

CategoryValidator categoryValidator;

@Autowired

@Qualifier("categoryDao")

CategoryDAO categoryDAO;

@InitBinder

private void initBinder(WebDataBinder binder) {

binder.setValidator(categoryValidator);

}

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

protected ModelAndView AdminAddCategories() throws Exception {

System.out.println("Admin home Add Categories clicked!");

return new ModelAndView("category-add-form","category",new Category());

}

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

public ModelAndView addCategory(@ModelAttribute("category") Category category, BindingResult result,HttpServletRequest Request) throws Exception {

System.out.println("Inside Add Category Controller!!!");

categoryValidator.validate(category, result);

if (result.hasErrors()) {

return new ModelAndView("category-add-form", "category", category);

}

try {

category = categoryDAO.create(category.getTitle());

} catch (CategoryException e) {

System.out.println(e.getMessage());

return new ModelAndView("error", "errorMessage", "error while login");

}

Request.getSession().setAttribute("success", "Successfully added Category!!");

return new ModelAndView("category-add-form", "category", new Category());

}

}

1. LINK CONTROLLER

package com.me.controller;

import java.util.List;

import java.util.Map;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpSession;

import org.apache.commons.mail.DefaultAuthenticator;

import org.apache.commons.mail.Email;

import org.apache.commons.mail.SimpleEmail;

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

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

import org.springframework.http.MediaType;

import org.springframework.stereotype.Controller;

import org.springframework.validation.BindingResult;

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.RequestMapping;

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

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

import org.springframework.web.servlet.ModelAndView;

import com.me.exception.UserException;

import com.me.pojo.Cart;

import com.me.pojo.Order;

import com.me.pojo.Product;

import com.me.pojo.User;

import com.me.validator.UserValidator;

import com.me.dao.CartDAO;

import com.me.dao.ProductDAO;

import com.me.dao.UserDAO;

@Controller

public class LinkController {

@Autowired

@Qualifier("userDao")

UserDAO userDao;

@Autowired

@Qualifier("cartDao")

CartDAO cartDao;

@Autowired

@Qualifier("productDao")

ProductDAO productDao;

@Autowired

@Qualifier("userValidator")

UserValidator validator;

@InitBinder

private void initBinder(WebDataBinder binder) {

binder.setValidator(validator);

}

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

protected ModelAndView backtohome(HttpServletRequest request) throws Exception {

System.out.println("homepage clicked!");

return new ModelAndView("home");

}

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

public ModelAndView redirectLogout(HttpServletRequest request) {

HttpSession session = request.getSession();

session.invalidate();

System.out.println("-----Session INVALIDATED!!-----");

return new ModelAndView("home");

}

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

protected ModelAndView LoginPageTranfer(HttpServletRequest request) throws Exception {

System.out.println("Login/SignUp clicked!");

List<Product> productlist = productDao.list();

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

System.out.println("NO OF PRODUCTS TO SHOW FOR USER:\t"+productlist.size());

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

return new ModelAndView("login");

}

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

protected ModelAndView RegisterPageTransferUser() throws Exception {

System.out.println("Register clicked!");

return new ModelAndView("register","user",new User());

}

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

protected ModelAndView CartPageTransfer(HttpServletRequest request) throws Exception {

System.out.println("Cart/Buy clicked!");

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

List<Cart> cartlist = cartDao.list(user.getPersonID());

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

double totalprice=0;

for(Cart cart:cartlist){

int quantity = cart.getQuantity();

double price = Double.parseDouble(cart.getProduct().getPrice());

double temp = quantity\*price;

totalprice = totalprice + temp;

}

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

return new ModelAndView("cart");

}

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

protected ModelAndView OrderHistoryPage(HttpServletRequest request) throws Exception {

System.out.println("OrderHistory clicked!");

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

Map<Integer,List<Order>> ordermap = cartDao.listordermap(user.getPersonID());

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

return new ModelAndView("orderhistory");

}

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

@ResponseBody

protected String UpdateCartDb(HttpServletRequest request) throws Exception {

System.out.println("Update in Cart clicked!!!");

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

Long pid = Long.parseLong(request.getParameter("pid"));

int quantity = Integer.parseInt(request.getParameter("quantity"));

cartDao.updatedb(pid, user.getPersonID(), quantity);

List<Cart> cartlist = cartDao.list(user.getPersonID());

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

double totalprice=0;

for(Cart cart:cartlist){

int quantity1 = cart.getQuantity();

double price = Double.parseDouble(cart.getProduct().getPrice());

double temp = quantity1\*price;

totalprice = totalprice + temp;

}

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

return String.valueOf(totalprice);

}

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

@ResponseBody

protected String DeleteCartValueDb(HttpServletRequest request) throws Exception {

System.out.println("Delete in Cart clicked!!!");

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

Long pid = Long.parseLong(request.getParameter("pid"));

cartDao.deleteindb(pid, user.getPersonID());

List<Cart> cartlist = cartDao.list(user.getPersonID());

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

double totalprice=0;

for(Cart cart:cartlist){

int quantity1 = cart.getQuantity();

double price = Double.parseDouble(cart.getProduct().getPrice());

double temp = quantity1\*price;

totalprice = totalprice + temp;

}

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

return String.valueOf(totalprice);

}

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

protected ModelAndView loginUserGet(HttpServletRequest request) throws Exception {

return loginUser(request);

}

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

protected ModelAndView loginUser(HttpServletRequest request) throws Exception {

HttpSession session = (HttpSession) request.getSession();

try {

User u = null;

User u1 = null;

System.out.println("loginUser -POST METHOD OF LOGIN!!");

if(request.getSession().getAttribute("user")==null){

u1 = userDao.get(request.getParameter("username"), request.getParameter("password"));

if(u1 == null){

System.out.println("UserName/Password does not exist");

session.setAttribute("errorMessage", "UserName/Password does not exist");

return new ModelAndView ("error");

}

}

if(request.getSession().getAttribute("user")!=null){

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

}else if(u1!=null){

u = u1;

}

System.out.println("USER ---------------------------\t"+u.getFirstName());

session.setAttribute("user", u);

//Create userhome.jsp in views

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

System.out.println("USER ROLE SET TO USER!!");

return new ModelAndView("user-home");

} catch (UserException e) {

System.out.println("Exception: " + e.getMessage());

session.setAttribute("errorMessage", "error while login");

return new ModelAndView ("error");

}

}

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

protected ModelAndView registerNewUser(HttpServletRequest request, @ModelAttribute("user") User user, BindingResult result) throws Exception {

System.out.println("Inside register controller!!!");

validator.validate(user, result);

System.out.println("Checking for user existance");

boolean flag = userDao.checkIfExists(user);

System.out.println("Checked for user existance!!");

if(!flag){

System.out.println("Same user found!!");

return new ModelAndView("register", "user", user);

}

if (result.hasErrors()) {

return new ModelAndView("register", "user", user);

}

System.out.print("registerNewUser");

String receiverEmail = user.getEmail().toString();

User u = null;

try {

u = userDao.register(user);

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

Email email = new SimpleEmail();

email.setSmtpPort(465);

email.setAuthenticator(new DefaultAuthenticator("chintankoticha@gmail.com", "dtvc4\_{koticha"));

email.setHostName("smtp.gmail.com");//if a server is capable of sending email.

email.setSSL(true);//setSSLOnConnect(true);

email.setFrom("chintankoticha@gmail.com");

email.setSubject("SUCCESSFUL SIGN UP!!!");

email.setMsg("This is system generated mail, do not reply to this email.\n\n"+"\nYOUR USERNAME IS:\t"+user.getUsername()+"\nKINDLY LOGIN TO THE WEBSITE!!");

email.addTo(receiverEmail);

email.setTLS(true);//startTLS.enable.true

email.send();

} catch (UserException e) {

System.out.println("Exception: " + e.getMessage());

return new ModelAndView("error", "errorMessage", "error while login");

} catch (Exception e) {

System.out.println("Exception e:" + e.getMessage());

}

System.out.println("EMAIL SENT TO NEW USER!!");

return new ModelAndView("user-home", "user", u);

}

}

1. MYVIEW CONTROLLER

package com.me.controller;

import java.awt.Color;

import java.util.List;

import java.util.Map;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

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

import com.lowagie.text.Chunk;

import com.lowagie.text.Document;

import com.lowagie.text.Font;

import com.lowagie.text.FontFactory;

import com.lowagie.text.Paragraph;

import com.lowagie.text.Phrase;

import com.lowagie.text.pdf.PdfPCell;

import com.lowagie.text.pdf.PdfPTable;

import com.lowagie.text.pdf.PdfWriter;

import com.me.pojo.Order;

public class MyView extends AbstractPdfView{

@Override

protected void buildPdfDocument(Map<String, Object> model, Document document, PdfWriter pdfwriter, HttpServletRequest request,

HttpServletResponse response) throws Exception {

Font helvetica\_18\_blue = new Font(Font.HELVETICA, 18, Font.BOLDITALIC, Color.BLUE);

Paragraph title = new Paragraph("Order Details", helvetica\_18\_blue);

Phrase firstPhrase = new Phrase("Details of the Order");

@SuppressWarnings("unchecked")

List<Order> list = (List<Order>)model.get("orderlist");

document.add(new Paragraph("----------------------------------------------------------------------------------------------------------------------------------"));

document.add(new Paragraph(""));

document.add(new Paragraph(""));

document.add(new Paragraph(""));

document.add(Chunk.NEWLINE);

PdfPTable table = new PdfPTable(5);

table.setTotalWidth(500);

table.setLockedWidth(true);

table.setWidths(new float[]{1, 1, 1, 1, 1});

PdfPCell cell;

table.addCell(new Paragraph("Product Name",FontFactory.getFont(FontFactory.TIMES, 15,Font.BOLD)));

table.addCell(new Paragraph("Quantity",FontFactory.getFont(FontFactory.TIMES, 15,Font.BOLD)));

table.addCell(new Paragraph("Product Price",FontFactory.getFont(FontFactory.TIMES, 15,Font.BOLD)));

table.addCell(new Paragraph("Seller Name",FontFactory.getFont(FontFactory.TIMES, 15,Font.BOLD)));

table.addCell(new Paragraph("Total Price",FontFactory.getFont(FontFactory.TIMES, 15,Font.BOLD)));

double totalprice = 0;

for(Order order:list)

{

table.addCell(new Paragraph(order.getProduct().getName(),FontFactory.getFont(FontFactory.TIMES, 11,Font.BOLD)));

table.addCell(String.valueOf(order.getQuantity()));

table.addCell(String.valueOf(order.getProduct().getPrice()));

table.addCell(String.valueOf((order.getProduct().getSeller().getName())));

table.addCell(String.valueOf((order.getQuantity()) \* Double.parseDouble((order.getProduct().getPrice()))));

totalprice = totalprice + (order.getQuantity()) \* Double.parseDouble((order.getProduct().getPrice()));

}

document.add(table);

Paragraph totalPrice = new Paragraph("Total Order Cost"+totalprice,FontFactory.getFont(FontFactory.TIMES, 10,Font.UNDERLINE));

document.add(totalPrice);

}

}

1. MYVIEW ORDER HISTORY

package com.me.controller;

import java.awt.Color;

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.beans.factory.annotation.Qualifier;

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

import com.lowagie.text.Chunk;

import com.lowagie.text.Document;

import com.lowagie.text.Font;

import com.lowagie.text.FontFactory;

import com.lowagie.text.Paragraph;

import com.lowagie.text.Phrase;

import com.lowagie.text.pdf.PdfPCell;

import com.lowagie.text.pdf.PdfPTable;

import com.lowagie.text.pdf.PdfWriter;

import com.me.dao.CartDAO;

import com.me.pojo.Order;

public class MyViewOrderHistory extends AbstractPdfView{

@Autowired

@Qualifier("cartDao")

CartDAO cartDao;

@Override

protected void buildPdfDocument(Map<String, Object> model, Document document, PdfWriter pdfwriter, HttpServletRequest request,

HttpServletResponse response) throws Exception {

Font helvetica\_18\_blue = new Font(Font.HELVETICA, 18, Font.BOLDITALIC, Color.BLUE);

Paragraph title = new Paragraph("Order Details", helvetica\_18\_blue);

Phrase firstPhrase = new Phrase("Details of the Order");

@SuppressWarnings("unchecked")

List<Order> list = (List<Order>)model.get("orderlist");

document.add(new Paragraph("----------------------------------------------------------------------------------------------------------------------------------"));

document.add(new Paragraph(""));

document.add(new Paragraph(""));

document.add(new Paragraph(""));

document.add(Chunk.NEWLINE);

PdfPTable table = new PdfPTable(5);

table.setTotalWidth(500);

table.setLockedWidth(true);

table.setWidths(new float[]{1, 1, 1, 1, 1});

PdfPCell cell;

table.addCell(new Paragraph("Product Name",FontFactory.getFont(FontFactory.TIMES, 15,Font.BOLD)));

table.addCell(new Paragraph("Quantity",FontFactory.getFont(FontFactory.TIMES, 15,Font.BOLD)));

table.addCell(new Paragraph("Product Price",FontFactory.getFont(FontFactory.TIMES, 15,Font.BOLD)));

table.addCell(new Paragraph("Seller Name",FontFactory.getFont(FontFactory.TIMES, 15,Font.BOLD)));

table.addCell(new Paragraph("Total Price",FontFactory.getFont(FontFactory.TIMES, 15,Font.BOLD)));

double totalprice = 0;

for(Order order:list)

{

table.addCell(new Paragraph(order.getProduct().getName(),FontFactory.getFont(FontFactory.TIMES, 11,Font.BOLD)));

table.addCell(String.valueOf(order.getQuantity()));

table.addCell(String.valueOf(order.getProduct().getPrice()));

table.addCell(String.valueOf((order.getProduct().getSeller().getName())));

table.addCell(String.valueOf((order.getQuantity()) \* Double.parseDouble((order.getProduct().getPrice()))));

totalprice = totalprice + (order.getQuantity()) \* Double.parseDouble((order.getProduct().getPrice()));

}

document.add(table);

Paragraph totalPrice = new Paragraph("Total Order Cost"+totalprice,FontFactory.getFont(FontFactory.TIMES, 10,Font.UNDERLINE));

document.add(totalPrice);

}

}

1. PRODUCT CONTROLLER

package com.me.controller;

import java.io.File;

import java.util.Arrays;

import java.util.List;

import javax.servlet.http.HttpServletRequest;

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

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

import org.springframework.stereotype.Controller;

import org.springframework.ui.Model;

import org.springframework.util.AutoPopulatingList;

import org.springframework.validation.BindingResult;

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.multipart.commons.CommonsMultipartFile;

import org.springframework.web.servlet.ModelAndView;

import com.me.dao.CategoryDAO;

import com.me.dao.ProductDAO;

import com.me.dao.SellerDAO;

import com.me.exception.CategoryException;

import com.me.exception.ProductException;

import com.me.exception.SellerException;

import com.me.pojo.Product;

import com.me.pojo.Seller;

import com.me.validator.ProductValidator;

import com.me.pojo.Category;

@Controller

public class ProductController {

@Autowired

@Qualifier("categoryDao")

CategoryDAO categoryDAO;

@Autowired

@Qualifier("productDao")

ProductDAO productDao;

@Autowired

@Qualifier("sellerDao")

SellerDAO sellerDao;

@Autowired

@Qualifier("productValidator")

ProductValidator productvalidator;

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

public String addProducts(@ModelAttribute Seller seller, Model model,HttpServletRequest request) throws Exception {

String hiddensellerid = request.getSession().getAttribute("hiddensellerid").toString();

System.out.println("Hidden seller ID:\t"+hiddensellerid);

seller = sellerDao.get(Integer.parseInt(hiddensellerid));

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

request.getSession().setAttribute("categories", categoryDAO.list());

request.getSession().setAttribute("postedby",seller.getName());

return productCreate(seller,model,true);

}

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

public ModelAndView updateProducts(@ModelAttribute Product product,Model model,HttpServletRequest request) throws Exception {

String hiddensellerid = request.getSession().getAttribute("hiddensellerid").toString();

System.out.println("Hidden seller ID:\t"+hiddensellerid);

List<Product> products = productDao.list(Long.parseLong(hiddensellerid));

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

request.getSession().setAttribute("noofproducts", products.size());

return new ModelAndView("UpdateProducts","products",products);

}

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

public ModelAndView viewandupdateproducts(@ModelAttribute Product product,Model model,HttpServletRequest request) throws Exception {

String productid = request.getParameter("id");

System.out.println("Product ID for that row:\t"+productid);

//List<Product> products = productDao.list(Long.parseLong(hiddensellerid));

Product p = productDao.get(Long.parseLong(productid));

return new ModelAndView("viewandupdateproduct","product",p);

}

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

public ModelAndView viewandupdateproducts1(@ModelAttribute Product product,Model model,HttpServletRequest request) throws Exception {

String productid = request.getParameter("id");

System.out.println("Product ID for the update function:\t"+productid);

//List<Product> products = productDao.list(Long.parseLong(hiddensellerid));

//Product p = productDao.get(Long.parseLong(productid));

String name = request.getParameter("name");

String description = request.getParameter("description");

String price = request.getParameter("price");

String quantity = request.getParameter("quantity");

//File Operations

CommonsMultipartFile photoInMemory = product.getPhoto();

String fileName = photoInMemory.getOriginalFilename();

System.out.println(fileName);

File localFile = new File("C:\\SpringProjects\\FinalProject\\src\\main\\webapp\\user\\resources\\images\\", fileName);

photoInMemory.transferTo(localFile);

System.out.println("File is stored at" + localFile.getPath());

product.setFilename(fileName);

product.setName(name);

product.setDescription(description);

product.setPrice(price);

product.setQuantity(quantity);

product.setId(Long.parseLong(productid));

product = productDao.update(product);

System.out.println("Updated product!!");

request.getSession().setAttribute("productupdate", "PRODUCT DETAILS SUCCESSFULLY UPDATED!!");

return new ModelAndView("viewandupdateproduct","product",product);

}

private String productCreate(Seller seller,Model model,boolean init)

{

System.out.println("productCreate method--1");

if(init)

{

AutoPopulatingList<Product> add= new AutoPopulatingList<Product>(Product.class);

add.add(new Product());

System.out.println("productCreate method--2");

seller.setProducts(add);

}

System.out.println("productCreate method--3");

model.addAttribute("type","create");

return "/products-add-form";

}

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

protected ModelAndView registerNewUser(HttpServletRequest request,@ModelAttribute Seller seller, BindingResult bindingResult, Model model) {

System.out.println("Product Add Post method--1");

try{

System.out.println("Product Add Post method--3");

String hiddensellerid = request.getSession().getAttribute("hiddensellerid").toString();

System.out.println("NUMBER OF PRODUCTS:\t"+seller.getProducts().size());

Seller s = sellerDao.get(Integer.parseInt(hiddensellerid));

System.out.println("\*\*\*\*\*\*Seller for which products are added:\t"+s.getId());

String[] categories = request.getParameterValues("categories");

for(String s1:categories){

System.out.println(s1);

}

for(Product p:seller.getProducts()){

p.setSeller(s);

//System.out.println(p.getQuantity());

System.out.println("Inside every products!!");

String string = p.getDescription();

//System.out.println(string);

String str2 = p.getName();

String str3 = p.getPrice();

String str4 = p.getQuantity();

CommonsMultipartFile str5 = p.getPhoto();

System.out.println("PHOTO KA PATH:\t"+str5); // filename

List<String> list = Arrays.asList(string.split(","));

List<String> list1 = Arrays.asList(str2.split(","));

List<String> list2 = Arrays.asList(str3.split(","));

List<String> list3 = Arrays.asList(str4.split(","));

//List<String> list4 = Arrays.asList(str5.split(","));

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

Product p1 = new Product();

p1.setName(list1.get(i));

p1.setPrice(list2.get(i));

p1.setQuantity(list3.get(i));

p1.setDescription(list.get(i));

//FILE OPERATIONS FOR IMAGE UPLOAD ON SERVER

p1.setSeller(s);

p1 = productDao.create(p1);

for(int i1=0;i1<categories.length;i1++){

Category c = categoryDAO.get(categories[i1]);

c.getProducts().add(p1);

categoryDAO.update(c);

}

System.out.println("Upodated Category-Product bridge table!!");

System.out.println("Created a product!!");

}

}

System.out.println("Created products and updated bridge table!!!");

return new ModelAndView("success");

}

catch (ProductException e)

{

System.out.println("Exception: " + e.getMessage());

return new ModelAndView("error", "errorMessage", "error while login");

}

catch (SellerException e) {

System.out.println("Exception: " + e.getMessage());

return new ModelAndView("error", "errorMessage", "error while updating seller on products addition!!");

} catch (CategoryException e) {

System.out.println("Exception: " + e.getMessage());

return new ModelAndView("error", "errorMessage", "error while updating category bridge table on products addition!!");

}

}

}

1. SELLER LINK CONTROLLER

package com.me.controller;

import java.util.List;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpSession;

import org.apache.commons.mail.DefaultAuthenticator;

import org.apache.commons.mail.Email;

import org.apache.commons.mail.SimpleEmail;

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

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

import org.springframework.stereotype.Controller;

import org.springframework.validation.BindingResult;

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.RequestMapping;

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

import org.springframework.web.context.request.RequestScope;

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

import org.springframework.web.servlet.ModelAndView;

import com.me.dao.SellerDAO;

import com.me.exception.SellerException;

import com.me.pojo.Product;

import com.me.pojo.Seller;

import com.me.validator.SellerValidator;

import com.me.dao.CategoryDAO;

@Controller

@RequestMapping("/seller/\*")

public class SellerLinkController {

@Autowired

@Qualifier("sellerDao")

SellerDAO sellerDao;

@Autowired

@Qualifier("categoryDao")

CategoryDAO categoryDAO;

@Autowired

@Qualifier("sellerValidator")

SellerValidator sellervalidator;

@InitBinder

private void initBinder(WebDataBinder binder) {

binder.setValidator(sellervalidator);

}

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

protected ModelAndView RegisterPageTransferSeller() throws Exception {

System.out.println("Register clicked!");

return new ModelAndView("seller-register","seller",new Seller());

}

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

protected ModelAndView registerNewSeller(HttpServletRequest request, @ModelAttribute("seller") Seller seller, BindingResult result) throws Exception {

System.out.println("Inside register controller for Seller!!!");

sellervalidator.validate(seller, result);

if (result.hasErrors()) {

return new ModelAndView("seller-register", "seller", seller);

}

System.out.println("registerNewSeller");

String receiverEmail = seller.getEmailAddress();

Seller s = null;

try {

s = sellerDao.register(seller);

request.getSession().setAttribute("seller", s);

Email email = new SimpleEmail();

email.setSmtpPort(465);

email.setAuthenticator(new DefaultAuthenticator("chintankoticha@gmail.com", "dtvc4\_{koticha"));

email.setHostName("smtp.gmail.com");//if a server is capable of sending email.

email.setSSL(true);//setSSLOnConnect(true);

email.setFrom("chintankoticha@gmail.com");

email.setSubject("SUCCESSFUL SIGN UP!!!");

email.setMsg("This is system generated mail, do not reply to this email.\n\n"+"\nYOUR USERNAME IS:\t"+seller.getUsername()+"\n"

+ "KINDLY LOGIN TO THE WEBSITE ONCE YOU APPROVED BY THE ADMIN!!");

email.addTo(receiverEmail);

email.setTLS(true);//startTLS.enable.true

email.send();

} catch (SellerException e) {

System.out.println("Exception: " + e.getMessage());

return new ModelAndView("error", "errorMessage", "error while login");

} catch (Exception e) {

System.out.println("Exception e:" + e.getMessage());

}

System.out.println("EMAIL SENT TO NEW SELLER!!");

return new ModelAndView("seller-register", "message", "Successfully registered, just wait for our team to approve you before can login!!");

}

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

protected ModelAndView loginSeller(HttpServletRequest request) throws Exception {

ModelAndView mv = new ModelAndView();

HttpSession session = (HttpSession) request.getSession();

try {

System.out.println("Inside LoginSeller Link Controller!!");

Seller u = sellerDao.get(request.getParameter("sellerusername"), request.getParameter("sellerpassword"));

if(u == null){

System.out.println("UserName/Password does not exist");

session.setAttribute("errorMessage", "UserName/Password does not exist");

return new ModelAndView ("error");

}

if(u.getStatus().equalsIgnoreCase("Pending")){

return new ModelAndView("login", "messagelogin", "PLEASE WAIT FOR OUR TEAM TO APPROVE YOU BEFORE YOU CAN LOGIN!!");

}else{

session.setAttribute("seller", u);

request.getSession().setAttribute("role", "seller");

SessionScope ss = new SessionScope();

RequestScope rs = new RequestScope();

ss.remove("messagelogin");

rs.remove("messagelogin");

mv.addObject("seller",u);

System.out.println("Adding to hidden field:\t"+u.getId());

request.getSession().setAttribute("hiddensellerid", u.getId());

mv.setViewName("seller-home");

return mv;

}

} catch (SellerException e) {

System.out.println("Exception: " + e.getMessage());

session.setAttribute("errorMessage", "error while login");

return new ModelAndView ("error");

}

}

}

1. USER CONTROLLER

package com.me.controller;

import java.text.DateFormat;

import java.text.SimpleDateFormat;

import java.util.ArrayList;

import java.util.Date;

import java.util.List;

import javax.servlet.http.HttpServletRequest;

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

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

import org.springframework.stereotype.Controller;

import org.springframework.web.bind.WebDataBinder;

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

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.me.dao.CartDAO;

import com.me.dao.ProductDAO;

import com.me.dao.UserDAO;

import com.me.pojo.Cart;

import com.me.pojo.Order;

import com.me.pojo.Product;

import com.me.pojo.User;

import com.me.validator.UserValidator;

@Controller

public class UserController {

@Autowired

@Qualifier("userDao")

UserDAO userDao;

@Autowired

@Qualifier("cartDao")

CartDAO cartDao;

@Autowired

@Qualifier("productDao")

ProductDAO productDao;

@Autowired

@Qualifier("userValidator")

UserValidator validator;

@InitBinder

private void initBinder(WebDataBinder binder) {

binder.setValidator(validator);

}

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

protected ModelAndView BuyProducts(HttpServletRequest request) throws Exception {

System.out.println("BUYPAGE clicked!");

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

@SuppressWarnings("unchecked")

List<Cart> cartlist = (List<Cart>) request.getSession().getAttribute("cartlist");

int max = cartDao.getmax();

System.out.println("MAX ORDERNUMBER:\t"+max);

if(max==0){

max = 1;

}else{

max = max+1;

}

List<Order> orderlist = new ArrayList<Order>();

for(Cart c:cartlist){

Order od = new Order();

DateFormat dateFormat = new SimpleDateFormat("yyyy/MM/dd");

Date date = new Date();

System.out.println(dateFormat.format(date));

od.setCreationDate(date);

od.setProduct(c.getProduct());

int quantity = cartDao.get(c.getProduct(),user.getPersonID());

od.setQuantity(quantity);

od.setUser(user);

od.setId(max);

cartDao.createOrder(od);

//Storing to show in pdf

orderlist.add(od);

cartDao.deleteindb(c.getProduct().getId(), user.getPersonID());

}

request.getSession().setAttribute("currentorderlist", orderlist);

//System.out.println("NO OF PRODUCTS TO SHOW FOR USER:\t"+productlist.size());

return new ModelAndView("buysuccess");

}

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

public ModelAndView pdfview(HttpServletRequest request) throws Exception

{

System.out.println("In Order pdf method");

View view = new MyView();

System.out.println("In Order pdf method");

@SuppressWarnings("unchecked")

List<Order> orderlist = (List<Order>)request.getSession().getAttribute("currentorderlist");

System.out.println("In Order pdf method");

System.out.println("orderId"+orderlist.size());

return new ModelAndView(view,"orderlist",orderlist);

}

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

public ModelAndView pdfviewOrderHistoryPage(HttpServletRequest request) throws Exception

{

System.out.println("In Order pdf pdfviewOrderHistoryPage method");

View view = new MyViewOrderHistory();

System.out.println("In Order pdf method");

int orderidpdf = Integer.parseInt(request.getParameter("id"));

@SuppressWarnings("unchecked")

List<Order> orderlist = cartDao.getOrderList(orderidpdf);

System.out.println("In Order pdf method");

System.out.println("orderId"+orderlist.size());

return new ModelAndView(view,"orderlist",orderlist);

}

}