

INFO-6250 Web Development Tools & Methods SEC 04 FINAL PROJECT

Name: SUMANTH HAGALAVADI GOPALAKRISHNA

NUID: 001824723

SUMMARY:

Designed and implemented an eCommerce application to the users for online shopping of daily used products. This web application was designed using Spring MVC framework and hibernate. This website will help users to shop online and order the products without physical presence. This application has multiple roles and secure server-side code with data validation in both front and back end.

FUNCTIONALITIES:

- Due to the presence of different roles, application works smoother and easier to implement.
- Admin can create users and assign role according to the requirement.
- Dealers can add the product, price, location and quantity associated to it.
- Users will receive email to confirm the account and verify it accordingly in order to restrict fake email id.
- Admin can view all the details of the products associated with the user and status of the products whether its delivered or shipped to the user.
- Password reminder link is given which will send an email if the user forgets the password at any point of time.
- Dealers will approve the request of products that has been checked and ordered by users.
- Dealers can see the quantity of product and add necessary amount based on customer requirement at any point of time
- Customers can add the product into the cart and view it during any time. Cart items are listed based on customer login.
- Once the customer adds the products to the cart, they can remove it or checkout for the order to be shipped.
- Once the order is placed by customer, dealers will see the checked-out products and approve for shipping and final delivery.

TECHNOLOGIES USED:

Spring MVC (annotated controller and pojo's), Validators, Exception Handling , DAO Pattern, Javax Mail, AJAX, Bootstrap, JQuery, MySql Database, Hibernate, Hibernate Query Language(HQL), Maven, Git.

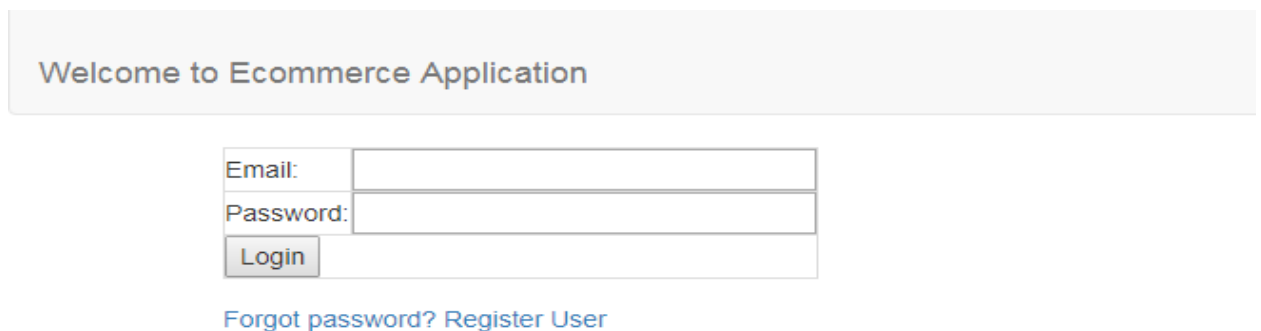
DESCRIPTION OF ROLES:

There are 3 roles which are defined in the application:

1. **Admin:** Used to create monitor the products and create users and assign the role as per the requirement.
2. **ShopOwner:** Used to add products and approve the request sent by customer
3. **Customer:** Used to add, modify and checkout the products

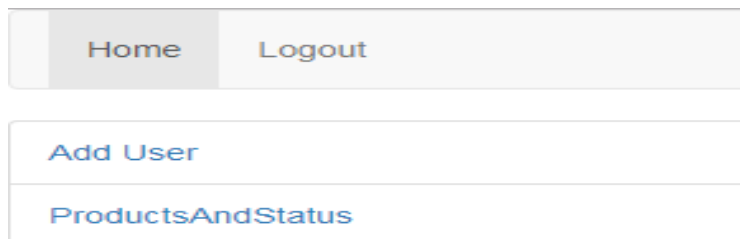
SCREENSHOTS:

1. User Authentication:



A screenshot of a user authentication page. At the top, a light gray banner contains the text "Welcome to Ecommerce Application". Below this, there is a login form with two input fields: "Email:" and "Password:". A "Login" button is positioned below the password field. At the bottom of the form, there are two links: "Forgot password?" and "Register User", both in blue text.

2. Admin Home Page:



A screenshot of an admin home page. At the top, there is a navigation bar with two buttons: "Home" and "Logout". Below this, there is a list of links: "Add User" and "ProductsAndStatus", both in blue text.

3. User creation page for admin:




Create Users for Admin

User Email:

Password:

Select Role Admin ▼

Retype the characters from the picture:



What is BotDetect Java CAPTCHA Library?

Login

4. Admin Product status Page:

Home	Logout
------	--------




Product Name	Price	Product Location	Product Status	User Email
"P1"	"3.0"	"I1"	"checked"	"withsumanth@gmail.com"
"p2"	"4.0"	"I2"	"shipped"	"withsumanth@gmail.com"
"p3"	"1.0"	"I3"	"created"	""
"fvd"	"3.0"	"vfd"	"checked"	"withsumanth@gmail.com"

5. Customer User Creation:

User Email:

Password:

Retype the characters from the picture:



What is BotDetect Java CAPTCHA Library?

Login

6. Dealer Home Page:

Home	Logout
------	--------

User Checked Products
Add Products
View Current Products

7. Product Addition Page:

Home	Logout
------	--------

Enter Product Details

Product Name	Price	Product Location

Add Products

8. Currently Added Products:

Home	Logout
------	--------

Product Name	Product Price	Product Location
"P1"	"3.0"	"I1"
"p2"	"4.0"	"I2"
"p3"	"1.0"	"I3"
"fvd"	"3.0"	"vfd"

9. User Product page:

View My Items	Logout	Cart Items2	
---------------	--------	-------------	--

Product Name	Price	Product Location	
"p3"	"1.0"	"l3"	Add to Cart
Checkout			

10. User Checkout Page:

View My Items	Logout
---------------	--------

Product Name	Price	Product Location	
"P1"	"3.0"	"l1"	Remove from Cart
"fvd"	"3.0"	"vfd"	Remove from Cart
Checkout			

11. Checkout Success Page:

Logout

Success

12. Products that need to be shipped by Dealers/Shop Owners:

"P1"	"3.0"	"l1"
"p2"	"4.0"	"l2"
"fvd"	"3.0"	"vfd"
Ship Products		

APPENDIX:

CONTROLLERS:

1. UserController:

```
package com.me.controller;

import java.net.URL;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.Random;

import javax.annotation.PostConstruct;
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.EmailException;
import org.apache.commons.mail.SimpleEmail;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Qualifier;
import org.springframework.stereotype.Controller;
import org.springframework.ui.ModelMap;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.servlet.ModelAndView;

import com.captcha.botdetect.web.servlet.Captcha;
import com.me.dao.ProductDAO;
import com.me.dao.UserDAO;
import com.me.pojo.User;
import com.me.pojo.Product;

@Controller
public class UserController {

    private static final Logger logger =
        LoggerFactory.getLogger(UserController.class);

    @Autowired
```

```
@Qualifier("userDao")
UserDAO userDao;
```

```
@Autowired
@Qualifier("productDao")
ProductDAO productDao;
```

```
@PostConstruct
public void init() {
    User existUser = userDao.checkInitialUser("admin@admin.com");
    if(existUser==null) {
        User u = new User();
        u.setUserEmail("admin@admin.com");
        u.setPassword("1");
        u.setStatus(1);
        u.setRoleName("admin");
        try {
            u = userDao.register(u);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

```
@RequestMapping(value = "/", method = RequestMethod.GET)
public ModelAndView userLoginForm(HttpServletRequest request) {
    HttpSession session = request.getSession();
    User u = (User) session.getAttribute("loggedInUser");
    if(u==null) {
        return new ModelAndView("user-login");
    }else {
        if(u.getRoleName().equals("admin")) {
            return new ModelAndView("admin");
        }else if(u.getRoleName().equals("shopowner")) {
            return new ModelAndView("shop-owner-init");
        }else {
            ModelAndView mv = new ModelAndView();
            List<Product> prodList = new ArrayList();
            prodList = productDao.getAllProducts();
            int checkedProd = productDao.getCheckedProducts();
            mv.addObject("prodList",prodList);
            mv.addObject("checkedProd",checkedProd);
            List<Product> cartProdList = new ArrayList();
            cartProdList = productDao.getUserProducts(u.getId());
            mv.addObject("cartProdList",cartProdList);
            mv.setViewName("customer");
        }
    }
}
```

```

        return mv;
    }
}

}

@RequestMapping(value = "/login.htm", method = RequestMethod.GET)
public String showLoginForm(HttpServletRequest request) {
    HttpSession session = request.getSession();
    User u = (User) session.getAttribute("loggedInUser");
    if(u==null) {
        return "user-login";
    }
    return "user-login";
}

@RequestMapping(value = "/login.htm", method = RequestMethod.POST)
public ModelAndView handleLoginForm(HttpServletRequest request, UserDao
userDao, ModelMap map) {

    HttpSession session = request.getSession();
    String username = request.getParameter("username");
    String password = request.getParameter("password");
    ModelAndView mv = new ModelAndView();
    try {
        User u = userDao.get(username, password);
        if (u != null && u.getStatus() == 1) {
            session.setAttribute("loggedInUser", u);
            if(u.getRoleName().equals("admin")) {
                return new ModelAndView("admin");
            }else if(u.getRoleName().equals("shopowner")) {
                return new ModelAndView("shop-owner-init");
            }else {
                List<Product> prodList = new ArrayList();
                prodList = productDao.getAllProducts();
                int checkedProd =
productDao.getCheckedProducts();
                mv.addObject("prodList",prodList);
                mv.addObject("checkedProd",checkedProd);
                List<Product> cartProdList = new ArrayList();
                cartProdList =
productDao.getUserProducts(u.getId());
                mv.addObject("cartProdList",cartProdList);
                mv.setViewName("customer");
                return mv;
            }
        } else if (u != null && u.getStatus() == 0) {

```



```

        map.addAttribute("errorMessage", "Please activate your
account to login!");
        return new ModelAndView("error");
    } else {
        map.addAttribute("errorMessage", "Invalid
username/password!");
        return new ModelAndView("error");
    }
} catch (Exception e) {
    e.printStackTrace();
}

return null;

}

```

```

@RequestMapping(value = "/create.htm", method = RequestMethod.GET)
public ModelAndView showCreateForm(HttpServletRequest request) {

```

```

    HttpSession session = request.getSession();
    User u = (User) session.getAttribute("loggedInUser");
    if(u==null) {
        return new ModelAndView("user-create-form");
    } else {
        if(u.getRoleName().equals("admin")) {
            return new ModelAndView("admin");
        } else if(u.getRoleName().equals("shopowner")) {
            return new ModelAndView("shop-owner-init");
        } else {
            ModelAndView mv = new ModelAndView();
            List<Product> prodList = new ArrayList();
            prodList = productDao.getAllProducts();
            int checkedProd = productDao.getCheckedProducts();
            mv.addObject("prodList", prodList);
            mv.addObject("checkedProd", checkedProd);
            List<Product> cartProdList = new ArrayList();
            cartProdList = productDao.getUserProducts(u.getId());
            mv.addObject("cartProdList", cartProdList);
            mv.setViewName("customer");
            return mv;
        }
    }
}

```

```

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

```

```

    public String handleCreateForm(HttpServletRequest request, UserDao userDao,
    ModelMap map) {
        Captcha captcha = Captcha.load(request, "CaptchaObject");
        String captchaCode = request.getParameter("captchaCode");
        HttpSession session = request.getSession();
        User ul = (User) session.getAttribute("loggedInUser");
        if (captcha.validate(captchaCode)) {
            String useremail = request.getParameter("username");
            String password = request.getParameter("password");
            User user = new User();
            user.setUserEmail(useremail);
            user.setPassword(password);
            user.setStatus(0);
            String role = request.getParameter("role");
            if(role==null) {
                user.setRoleName("customer");
            }else {
                user.setRoleName(role);
            }
            try {
                URL url;
                url = new URL(request.getRequestURL().toString());
                String scheme = url.getProtocol();
                String host = url.getHost();
                int port = url.getPort();
                String contextPath = request.getContextPath();
                User u = userDao.register(user);
                Random r= new Random();
                int randomNum1 = r.nextInt(50000);
                int randomNum2 = r.nextInt(50000);
                try {
                    String str =
scheme+"://"+host+": "+port+contextPath+"/validateemail.htm?email=" + useremail +
"&key1="
                                + randomNum1 + "&key2=" +
randomNum2;
                                session.setAttribute("key1", randomNum1);
                                session.setAttribute("key2", randomNum2);
                                sendEmail(useremail,
                                "Click on this link to activate your
account : "+ str);
                                } catch (Exception e) {
                                    System.out.println("Email cannot be sent due to
exception");
                                }
                            } catch (Exception e) {

```

```

        e.printStackTrace();
    }
} else {
    map.addAttribute("errorMessage", "Invalid Captcha!");
    return "user-create-form";
}

return "user-created";
}

@RequestMapping(value = "/forgotpassword.htm", method =
RequestMethod.GET)
public String getForgotPasswordForm(HttpServletRequest request) {

    return "forgot-password";
}

@RequestMapping(value = "/forgotpassword.htm", method =
RequestMethod.POST)
public String handleForgotPasswordForm(HttpServletRequest request, UserDao
 userDao) {

    String useremail = request.getParameter("useremail");
    Captcha captcha = Captcha.load(request, "CaptchaObject");
    String captchaCode = request.getParameter("captchaCode");

    if (captcha.validate(captchaCode)) {
        User user = userDao.get(useremail);
        sendEmail(useremail, "Your password is : " + user.getPassword());
        return "forgot-password-success";
    } else {
        request.setAttribute("captchamsg", "Captcha not valid");
        return "forgot-password";
    }
}

@RequestMapping(value = "user/resendemail.htm", method =
RequestMethod.POST)
public String resendEmail(HttpServletRequest request) {
    HttpSession session = request.getSession();
    String useremail = request.getParameter("username");
    Random rand = new Random();
    int randomNum1 = rand.nextInt(5000000);
    int randomNum2 = rand.nextInt(5000000);
    try {
        URL url;

```

```

        url = new URL(request.getRequestURL().toString());
        String scheme = url.getProtocol();
        String host = url.getHost();
        int port = url.getPort();
        String contextPath = request.getContextPath();
        String str =
scheme+"://"+host+": "+port+contextPath+"/validateemail.htm?email=" + useremail +
"&key1=" + randomNum1
                                + "&key2=" + randomNum2;
        session.setAttribute("key1", randomNum1);
        session.setAttribute("key2", randomNum2);
        sendEmail(useremail,
                                "Click on this link to activate your account : "+ str);
    } catch (Exception e) {
        System.out.println("Email cannot be sent");
    }

    return "user-created";
}

public void sendEmail(String useremail, String message) {
    try {
        Email email = new SimpleEmail();
        email.setHostName("smtp.googlemail.com");
        email.setSmtpport(465);
        email.setAuthenticator(new
DefaultAuthenticator("contactapplication2018@gmail.com", "springmvc"));
        email.setSSLonConnect(true);
        email.setFrom("no-reply@msis.neu.edu"); // This user email does
not
        // exist
        email.setSubject("Password Reminder");
        email.setMsg(message); // Retrieve email from the DAO and send
this
        email.addTo(useremail);
        email.send();
    } catch (EmailException e) {
        System.out.println("Email cannot be sent");
    }
}

@RequestMapping(value = "validateemail.htm", method = RequestMethod.GET)
public String validateEmail(HttpServletRequest request, UserDao userDao,
ModelMap map) {

```

```

        HttpSession session = request.getSession();
        String email = request.getParameter("email");
        int key1 = Integer.parseInt(request.getParameter("key1"));
        int key2 = Integer.parseInt(request.getParameter("key2"));
        System.out.println(session.getAttribute("key1") );
        System.out.println(session.getAttribute("key2") );

        if ((Integer)(session.getAttribute("key1")) == key1 &&
            ((Integer)session.getAttribute("key2"))== key2) {
            try {
                System.out.println("HI_____");
                boolean updateStatus = userDao.updateUser(email);
                if (updateStatus) {
                    return "user-login";
                } else {
                    return "error";
                }
            } catch (Exception e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
        } else {
            map.addAttribute("errorMessage", "Link expired , generate new
link");

            map.addAttribute("resendLink", true);
            return "error";
        }

        return "user-login";

    }

}

```

2. AdminController:

```
package com.me.controller;

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

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpSession;

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.servlet.ModelAndView;

import com.me.dao.AdminDAO;
import com.me.dao.ProductDAO;
import com.me.pojo.Product;
import com.me.pojo.User;

@Controller
public class AdminController {

    @Autowired
    @Qualifier("adminDao")
    AdminDAO adminDao;

    @RequestMapping(value = "/registerAdminUser.htm", method =
RequestMethod.GET)
    public ModelAndView showAdminLoginForm(HttpServletRequest request) {
        HttpSession session = request.getSession();
        User u = (User) session.getAttribute("loggedInUser");
        if (u == null) {
            return new ModelAndView("user-login");
        }
        return new ModelAndView("user-create-form", "admin", "admin");
    }

    @RequestMapping(value = "/prodAndStatus.htm", method =
RequestMethod.GET)
    public ModelAndView showProductsAdmin(HttpServletRequest request) {
        HttpSession session = request.getSession();
        User u = (User) session.getAttribute("loggedInUser");
        if (u == null) {
```

```

        return new ModelAndView("user-login");
    }
    List<Product> checkedList = new ArrayList();
    ModelAndView mv = new ModelAndView();
    checkedList = adminDao.getAllProducts();
    mv.addObject("checkedList", checkedList);
    mv.setViewName("prodAndStatus");
    return mv;
}

@RequestMapping(value = "/logout.htm", method = RequestMethod.GET)
public String logOut(HttpServletRequest request) {
    HttpSession session = request.getSession();
    session.invalidate();
    return "user-login";
}

@RequestMapping(value = "/admin.htm", method = RequestMethod.GET)
public ModelAndView adminMain(HttpServletRequest request) {
    HttpSession session = request.getSession();
    User u = (User) session.getAttribute("loggedInUser");
    if (u == null) {
        return new ModelAndView("user-login");
    }
    return new ModelAndView("admin");
}
}

```

3. **ProductController:**

```

package com.me.controller;

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

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpSession;

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

```

```

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

import com.me.dao.ProductDAO;
import com.me.pojo.Product;
import com.me.pojo.User;

@Controller
public class ProductController {

    @Autowired
    @Qualifier("productDao")
    ProductDAO productDao;

    @RequestMapping(value = "/user-shop-checkout.htm", method =
RequestMethod.GET)
    protected ModelAndView checkedProducts(HttpServletRequest request) throws
Exception {
        HttpSession session = request.getSession();
        User u = (User) session.getAttribute("loggedInUser");
        if (u == null) {
            return new ModelAndView("user-login");
        }
        List<User> userList = new ArrayList();
        ModelAndView mv = new ModelAndView();
        userList = productDao.getAllUsers();
        mv.addObject("userList", userList);
        mv.setViewName("user-shop-checkout");
        return mv;
    }

    @RequestMapping(value = "/addShopProducts.htm", method =
RequestMethod.GET)
    protected String viewProduct(HttpServletRequest request) throws Exception {
        HttpSession session = request.getSession();
        User u = (User) session.getAttribute("loggedInUser");
        if (u == null) {
            return "user-login";
        }
        return "addShopProducts";
    }

    @RequestMapping(value = "/shopOwner.htm", method = RequestMethod.GET)
    protected String shopMain(HttpServletRequest request) throws Exception {
        HttpSession session = request.getSession();
        User u = (User) session.getAttribute("loggedInUser");

```



```

        if (u == null) {
            return "user-login";
        }
        return "shop-owner-init";
    }

```

```

@RequestMapping(value = "/viewshopproduct.htm", method =
RequestMethod.GET)
protected ModelAndView addProducts(HttpServletRequest request) throws
Exception {
    HttpSession session = request.getSession();
    User u = (User) session.getAttribute("loggedInUser");
    if (u == null) {
        return new ModelAndView("user-login");
    }
    List<Product> checkedList = new ArrayList();
    ModelAndView mv = new ModelAndView();
    checkedList = productDao.findProductList();
    mv.addObject("checkedList", checkedList);
    mv.setViewName("viewshopproduct");
    return mv;
}

```

```

@RequestMapping(value = "/sproducts.htm", method = RequestMethod.POST)
protected ModelAndView numberOfProducts(HttpServletRequest request)
throws Exception {
    HttpSession session = request.getSession();
    User u = (User) session.getAttribute("loggedInUser");
    if (u == null) {
        return new ModelAndView("user-login");
    }
    return new ModelAndView("sproducts", "noOfProducts",
request.getParameter("noOfProducts"));
}

```

```

@RequestMapping(value = "/afteraddproducts", method =
RequestMethod.POST)
protected ModelAndView afterAddProducts(HttpServletRequest request) throws
Exception {
    HttpSession session = request.getSession();
    User u = (User) session.getAttribute("loggedInUser");
    if (u == null) {
        return new ModelAndView("user-login");
    }
    String noOfProd = request.getParameter("noOfProducts");
    // HttpSession session = request.getSession();

```

```

        // User loggedIn = (User) session.getAttribute("loggedInUser");
        int noOfProducts = Integer.parseInt(noOfProd);
        for (int i = 0; i < noOfProducts; i++) {
            Product products = new Product();
            products.setProductName(request.getParameter("productName"
+ (i + 1)));
            products.setProdLocation(request.getParameter("prodLocation" +
(i + 1)));
            products.setPrice(Float.parseFloat(request.getParameter("price"
+ (i + 1))));
            products.setProdStatus("created");
            productDao.register(products);
        }
        return new ModelAndView("shop-owner-init", "", "");
    }

```

```

    @RequestMapping(value = "/viewConfirmedProd", method =
RequestMethod.POST)
    protected ModelAndView toConfirmProd(HttpServletRequest request) throws
Exception {
        HttpSession session = request.getSession();
        User u = (User) session.getAttribute("loggedInUser");
        if (u == null) {
            return new ModelAndView("user-login");
        }
        String email = request.getParameter("productList");
        List<Product> prodList = productDao.getProducts(email);
        session.setAttribute("prodList", prodList);
        ModelAndView mv = new ModelAndView();
        mv.addObject("prodList", prodList);
        mv.setViewName("viewConfirmedProd");
        return mv;
    }

```

```

    @RequestMapping(value = "/shop-owner-init", method = RequestMethod.POST)
    protected String afterConfirm(HttpServletRequest request) throws Exception {
        HttpSession session = request.getSession();
        User u = (User) session.getAttribute("loggedInUser");
        if (u == null) {
            return "user-login";
        }
        List<Product> prodList = (List<Product>) session.getAttribute("prodList");
        for (Product p : prodList) {
            productDao.updateStatus(p);
        }
        return "shop-owner-init";
    }

```

```
}  
}
```

4. UserCartController:

```
package com.me.controller;  
  
import java.util.ArrayList;  
import java.util.List;  
  
import javax.servlet.http.HttpServletRequest;  
import javax.servlet.http.HttpSession;  
  
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.web.bind.annotation.RequestMapping;  
import org.springframework.web.bind.annotation.RequestMethod;  
import org.springframework.web.bind.annotation.RequestParam;  
import org.springframework.web.bind.annotation.ResponseBody;  
import org.springframework.web.servlet.ModelAndView;  
  
import com.me.dao.CartDAO;  
import com.me.dao.ProductDAO;  
import com.me.pojo.Product;  
import com.me.pojo.User;  
  
@Controller  
public class UserCartController {  
  
    @Autowired  
    @Qualifier("cartDao")  
    CartDAO cartDao;  
  
    @RequestMapping(value = "/cartChange.htm", method = RequestMethod.POST,  
        produces = MediaType.APPLICATION_JSON_VALUE)  
    public @ResponseBody String submit(HttpServletRequest request,  
        @RequestParam("name") String name, @RequestParam("price") String price,  
        @RequestParam("location") String location) {  
        HttpSession session = request.getSession();  
        User u = (User) session.getAttribute("loggedInUser");  
        if(u==null) {  
            return "user-login";  
        }  
    }  
}
```

```

        name = name.replaceAll("^\\|\\$", "");
        price = price.replaceAll("^\\|\\$", "");
        location = location.replaceAll("^\\|\\$", "");
        double d = Double.parseDouble(price);
        System.out.println(d);
        User loggedInUser = (User) session.getAttribute("loggedInUser");
        cartDao.updateCheckedProduct(name,price,location,loggedInUser);
        return null;
    }

    @RequestMapping(value = "/removeCart.htm", method =
RequestMethod.POST, produces = MediaType.APPLICATION_JSON_VALUE)
    public @ResponseBody String removeCart(HttpServletRequest request,
@RequestParam("name") String name,@RequestParam("price") String price,
@RequestParam("location") String location) {
        HttpSession session = request.getSession();
        User u = (User) session.getAttribute("loggedInUser");
        if(u==null) {
            return "user-login";
        }
        name = name.replaceAll("^\\|\\$", "");
        price = price.replaceAll("^\\|\\$", "");
        location = location.replaceAll("^\\|\\$", "");
        double d = Double.parseDouble(price);
        User loggedInUser = (User) session.getAttribute("loggedInUser");
        cartDao.updateRemovedProduct(name,price,location,loggedInUser);
        return null;
    }

    @RequestMapping(value = "/checkout", method = RequestMethod.POST)
    protected ModelAndView afterAddProducts(HttpServletRequest request) throws
Exception {
        HttpSession session = request.getSession();
        User ul = (User) session.getAttribute("loggedInUser");
        if(ul==null) {
            return new ModelAndView("user-login");
        }
        List<Product> checkedList = new ArrayList();
        ModelAndView mv = new ModelAndView();
        User u = (User) session.getAttribute("loggedInUser");
        checkedList = cartDao.findCheckedList(u);
        mv.addObject("checkedList",checkedList);
        mv.setViewName("checkout");
        return mv;
    }

```

```

    @RequestMapping(value = "/submitProduct", method = RequestMethod.POST)
    protected ModelAndView afterSubmitProducts(HttpServletRequest request)
    throws Exception {
        HttpSession session = request.getSession();
        User ul = (User) session.getAttribute("loggedInUser");
        if(ul==null) {
            return new ModelAndView("user-login");
        }
        List<Product> submittedList = new ArrayList();
        ModelAndView mv = new ModelAndView();
        User u = (User) session.getAttribute("loggedInUser");
        submittedList = cartDao.findCheckedList(u);
        for(Product p:submittedList) {
            cartDao.updateConfirmedStatus(p);
        }
        mv.setViewName("submitProduct");
        return mv;
    }

```

```

    @RequestMapping(value = "/viewMyItems.htm", method =
    RequestMethod.GET)
    protected ModelAndView viewMyProducts(HttpServletRequest request) throws
    Exception {
        HttpSession session = request.getSession();
        User ul = (User) session.getAttribute("loggedInUser");
        if(ul==null) {
            return new ModelAndView("user-login");
        }
        List<Product> submittedList = new ArrayList();
        ModelAndView mv = new ModelAndView();
        User u = (User) session.getAttribute("loggedInUser");
        submittedList = cartDao.findShippedList(u);
        mv.addObject("prodList",submittedList);
        mv.setViewName("viewMyItems");
        return mv;
    }
}

```

POJO Classes

1. User

```
package com.me.pojo;

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

@Entity
@Table(name = "userTable")
public class User {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name="id", unique = true, nullable = false)
    private long id;

    @Column(name = "userEmail")
    private String userEmail;

    @Column(name = "password")
    private String password;

    @Column(name="status")
    private int status;

    @Column(name="roleName")
    private String roleName;

    public User() {

    }

    public long getId() {
        return id;
    }

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

    public String getRoleName() {
```

```

        return roleName;
    }

    public void setRoleName(String roleName) {
        this.roleName = roleName;
    }

    public String getUserEmail() {
        return userEmail;
    }

    public void setUserEmail(String userEmail) {
        this.userEmail = userEmail;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }

    public int getStatus() {
        return status;
    }

    public void setStatus(int status) {
        this.status = status;
    }
}

```

2. Product

```

package com.me.pojo;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.ManyToOne;
import javax.persistence.Table;

@Entity
@Table(name = "product")

```

```
public class Product {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name="pid", unique = true, nullable = false)
    private long pid;

    private String productName;
    private double price;
    private String prodStatus;
    private String prodLocation;

    @ManyToOne
    User user;

    public String getProdLocation() {
        return prodLocation;
    }

    public void setProdLocation(String prodLocation) {
        this.prodLocation = prodLocation;
    }

    public String getProdStatus() {
        return prodStatus;
    }

    public void setProdStatus(String prodStatus) {
        this.prodStatus = prodStatus;
    }

    public long getPid() {
        return pid;
    }

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

    public String getProductName() {
        return productName;
    }

    public void setProductName(String productName) {
        this.productName = productName;
    }
}
```



```

    public double getPrice() {
        return price;
    }

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

    public User getUser() {
        return user;
    }

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

```

DAO Classes

1. AdminDAO

```

package com.me.dao;

import java.util.List;

import org.hibernate.HibernateException;
import org.hibernate.Query;

import com.me.pojo.Product;

public class AdminDAO extends DAO{
    public List<Product> getAllProducts() {
        try {
            begin();
            Query q = getSession().createQuery("from Product");
            List<Product> prodList = q.list();
            close();
            return prodList;

        } catch (HibernateException e) {
            System.out.println("Error getting Product List " + e.getMessage());
        }
        return null;
    }
}

```

2. CartDAO:

```
package com.me.dao;

import java.util.List;

import org.hibernate.HibernateException;
import org.hibernate.Query;

import com.me.pojo.Product;
import com.me.pojo.User;

public class CartDAO extends DAO {

    public void updateCheckedProduct(String name, String price, String location,
    User u) {
        try {
            begin();
            Query q = getSession().createQuery(
                "from Product where productName =
:productName and price=:price and prodLocation=:prodLocation");
            q.setString("productName", name);
            q.setDouble("price", Double.parseDouble(price));
            q.setString("prodLocation", location);
            Product p = (Product) q.uniqueResult();
            if (p != null) {
                p.setProdStatus("checked");
                p.setUser(u);
                getSession().update(p);
                commit();
            }
            close();
        } catch (HibernateException e) {
            rollback();
            System.out.println("Error updating Product List " +
e.getMessage());
        }
    }

    public void updateRemovedProduct(String name, String price, String location,
    User u) {
        try {
            begin();
            Query q = getSession().createQuery(
                "from Product where productName =
:productName and price=:price and prodLocation=:prodLocation");
```

```

        q.setString("productName", name);
        q.setDouble("price", Double.parseDouble(price));
        q.setString("prodLocation", location);
        Product p = (Product) q.uniqueResult();
        if (p != null) {
            p.setProdStatus("created");
            p.setUser(null);
            getSession().update(p);
            commit();
        }
        close();
    } catch (HibernateException e) {
        rollback();
        System.out.println("Error updating Product List " +
e.getMessage());
    }
}

    public List<Product> findCheckedList(User u) {
        try {
            begin();
            Query q = getSession().createQuery("from Product where
prodStatus=:prodStatus and user=:user");
            q.setString("prodStatus", "checked");
            q.setParameter("user", u);
            List<Product> pList = q.list();
            close();
            return pList;
        } catch (HibernateException e) {
            rollback();
            System.out.println("Error updating Product List " +
e.getMessage());
        }
        return null;
    }

    public void updateConfirmedStatus(Product p) {
        try {
            begin();
            p.setProdStatus("confirmed");
            getSession().update(p);
            commit();
            close();
        } catch (HibernateException e) {
            rollback();

```

```

        System.out.println("Error updating Product List " +
e.getMessage());
    }

    }

    public List<Product> findShippedList(User u) {
        try {
            begin();
            Query q = getSession().createQuery("from Product where
prodStatus=:prodStatus and user=:user");
            q.setString("prodStatus", "shipped");
            q.setParameter("user", u);
            List<Product> pList = q.list();
            close();
            return pList;
        } catch (HibernateException e) {
            rollback();
            System.out.println("Error updating Product List " +
e.getMessage());
        }
        return null;
    }
}

```

3. **ProductDAO:**

```

package com.me.dao;

import java.util.List;

import org.hibernate.HibernateException;
import org.hibernate.Query;

import com.me.pojo.Product;
import com.me.pojo.User;

public class ProductDAO extends DAO {

    public ProductDAO() {

    }

    public Product register(Product p) throws Exception {
        try {
            begin();

```

```

        getSession().save(p);
        commit();
        close();
        return p;

    } catch (HibernateException e) {
        rollback();
        throw new Exception("Exception while creating product: " +
e.getMessage());
    }
}

public List<Product> getAllProducts() {
    try {
        begin();
        Query q = getSession().createQuery("from Product where
prodStatus=:prodStatus");
        q.setString("prodStatus", "created");
        List<Product> prodList = q.list();
        close();
        return prodList;

    } catch (HibernateException e) {
        System.out.println("Error getting Product List " + e.getMessage());
    }
    return null;
}

public List<Product> getUserProducts(long userId) {
    try {
        begin();
        Query q = getSession().createQuery("from Product where
id=:userId");
        q.setLong("userId", userId);
        List<Product> prodList = q.list();
        close();
        return prodList;
    } catch (HibernateException e) {
        System.out.println("Error getting Product List " + e.getMessage());
    }
    return null;
}

public int getCheckedProducts() {
    try {
        begin();

```

```

        Query q = getSession().createQuery("select count(*) from Product
where prodStatus=:prodStatus");
        q.setString("prodStatus", "checked");
        Long count = (Long) q.uniqueResult();
        close();
        return Long.valueOf(count).intValue();
    } catch (HibernateException e) {
        System.out.println("Error getting Product List " + e.getMessage());
    }
    return 0;
}

```

```

public List<Product> findProductList() {
    try {
        begin();
        Query q = getSession().createQuery("from Product");
        List<Product> pList = q.list();
        close();
        return pList;
    } catch (HibernateException e) {
        rollback();
        System.out.println("Error updating Product List " +
e.getMessage());
    }
    return null;
}

```

```

public List<User> getAllUsers() {
    try {
        begin();
        Query q = getSession().createQuery("select distinct user from
Product where prodStatus=:prodStatus");
        q.setString("prodStatus", "confirmed");
        List<User> pList = q.list();
        close();
        return pList;
    } catch (HibernateException e) {
        rollback();
        System.out.println("Error updating Product List " +
e.getMessage());
    }
    return null;
}

```

```

public List<Product> getProducts(String email) {
    try {

```

```

        begin();
        Query q = getSession().createQuery("from User where
userEmail=:userEmail");
        q.setString("userEmail", email);
        User u = (User) q.uniqueResult();
        close();
        begin();
        Query q1 = getSession().createQuery("from Product where
user=:user");
        q1.setParameter("user", u);
        List<Product> pList = q1.list();
        return pList;
    } catch (HibernateException e) {
        rollback();
        System.out.println("Error updating Product List " +
e.getMessage());
    }
    return null;
}

    public void updateStatus(Product p) {
        try {
            begin();
            p.setProdStatus("shipped");
            getSession().update(p);
            commit();
            close();
        } catch (HibernateException e) {
            rollback();
            System.out.println("Error updating Product List " +
e.getMessage());
        }
    }
}

```

4. **UserDAO:**

```

package com.me.dao;

import org.hibernate.HibernateException;
import org.hibernate.Query;

import com.me.pojo.User;

public class UserDAO extends DAO {

```

```

public UserDAO() {
}

public User get(String userEmail, String password) throws Exception {
    try {
        begin();
        Query q = getSession().createQuery("from User where userEmail =
:useremail and password = :password");
        q.setString("useremail", userEmail);
        q.setString("password", password);
        User user = (User) q.uniqueResult();
        close();
        return user;
    } catch (HibernateException e) {
        rollback();
        throw new Exception("Could not get user " + userEmail, e);
    }
}

public User get(String userEmail) {
    try {
        begin();
        Query q = getSession().createQuery("from User where userEmail =
:useremail");
        q.setString("useremail", userEmail);
        User user = (User) q.uniqueResult();
        close();
        return user;
    } catch (Exception e) {
        System.out.println(e.getMessage());
        rollback();
    }
    return null;
}

public User register(User u) throws Exception {
    try {
        begin();
        System.out.println("inside DAO");
        getSession().save(u);
        commit();
        close();
        return u;
    }
}

```



```

        } catch (HibernateException e) {
            rollback();
            throw new Exception("Exception while creating user: " +
e.getMessage());
        }
    }

    public boolean updateUser(String email) throws Exception {
        try {
            begin();
            Query q = getSession().createQuery("from User where userEmail =
:useremail");

            q.setString("useremail", email);
            User user = (User) q.uniqueResult();
            if (user != null) {
                user.setStatus(1);
                getSession().update(user);
                commit();
                close();
                return true;
            } else {
                return false;
            }
        }

        } catch (HibernateException e) {
            rollback();
            throw new Exception("Exception while creating user: " +
e.getMessage());
        }
    }

    public User checkInitialUser(String email) {
        try {
            begin();
            Query q = getSession().createQuery("from User where userEmail =
:useremail");

            q.setString("useremail", email);
            User user = (User) q.uniqueResult();
            close();
            return user;
        } catch (HibernateException e) {
            rollback();
            System.out.println("Could not get user " + email +
e.getMessage());
        }
    }

```

```
        return null;
    }
}
```