NEU Market Place

By

Deepak Chandwani

NUID: 001822358

Under Professor Yousuf Ozbek

INFO6250 34697 Web Development Tools & Methds SEC 04 - Spring 2018

Contents

Summary

NEU Market Place is a smart e-commerce web app, which is designed for only Northeastern Students (students who has @husky.neu.edu email). We have "*three roles"* in this application, which are Admin, Access Control and Users. Users can sell and purchase products under the specific category.

When selling they are required to pull all the fields, which includes selecting an image file. Then the product will be send to access control for review, Access Control is authorized to approve or reject the product posted by an user.

Previous orders records are maintained for every user. They can view or download the *Excel* file of their ordered products. While placing order the user can view or save the *PDF* invoice which will contain delivery details and products ordered.

Admin can create categories under which the user will post his product along with image. Programmatic security is used to verify the that he is authorized to create new categories.

Features

- **Three Roles**, all have different set of responsibilities
- **Programmatic Security** is used when Admin log in
- When a user is registered an **email** is fired to notify, that registration is successful
- User can **register** by their **husky email only**
- Users can **sell** and **buy products** from this web app.
- When selling products all the fields are mandatory and **checked** and **validated** using **Angular**
- **Image** can be **uploaded** for the products which user want to sell
- **Filters** are used to remove any **special characters** which are intended to hard the database
- **Criteria** is used get the products
- **JQuery** and **AJAX** is used to retrieve the result on the fly when any character is used
- **REST** API is used to get the results
- **PDF** can be viewed and saved of the products ordered
- **Order confirmation email** is fired when an Order is placed successfully
- **Excel can be downloaded** of previous placed orders by the user

Technology Used

- Spring MVC
- Hibernate
- AJAX
- JQuery
- MYSQL
- STS IDE

Description of Roles

Admin

- Admin is responsible to create new categories by which user will filter and shop products from it, it also helps in adding products to users
- Programmatic security is user to Authenticate Admin

Access Control

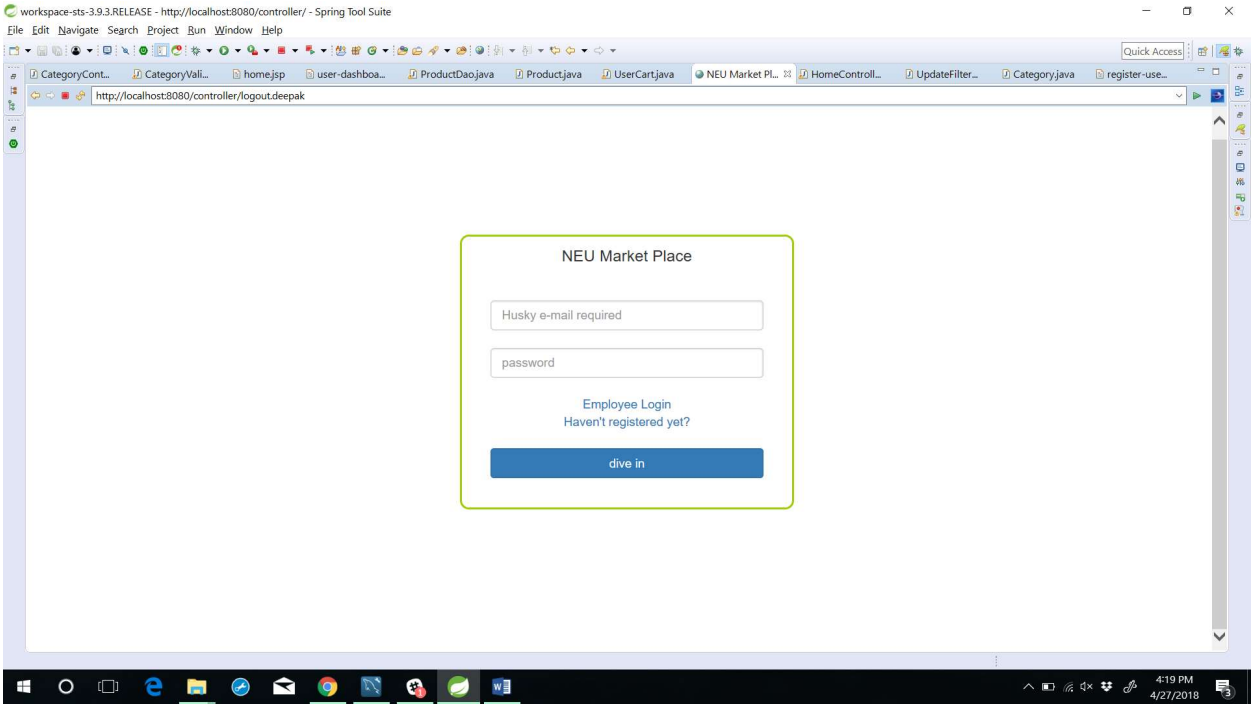
- Access control is responsible for accepting or rejecting the products uploaded by the users.
- If he finds it all okay he will approve
- If he finds anything empty or malicious will reject

User

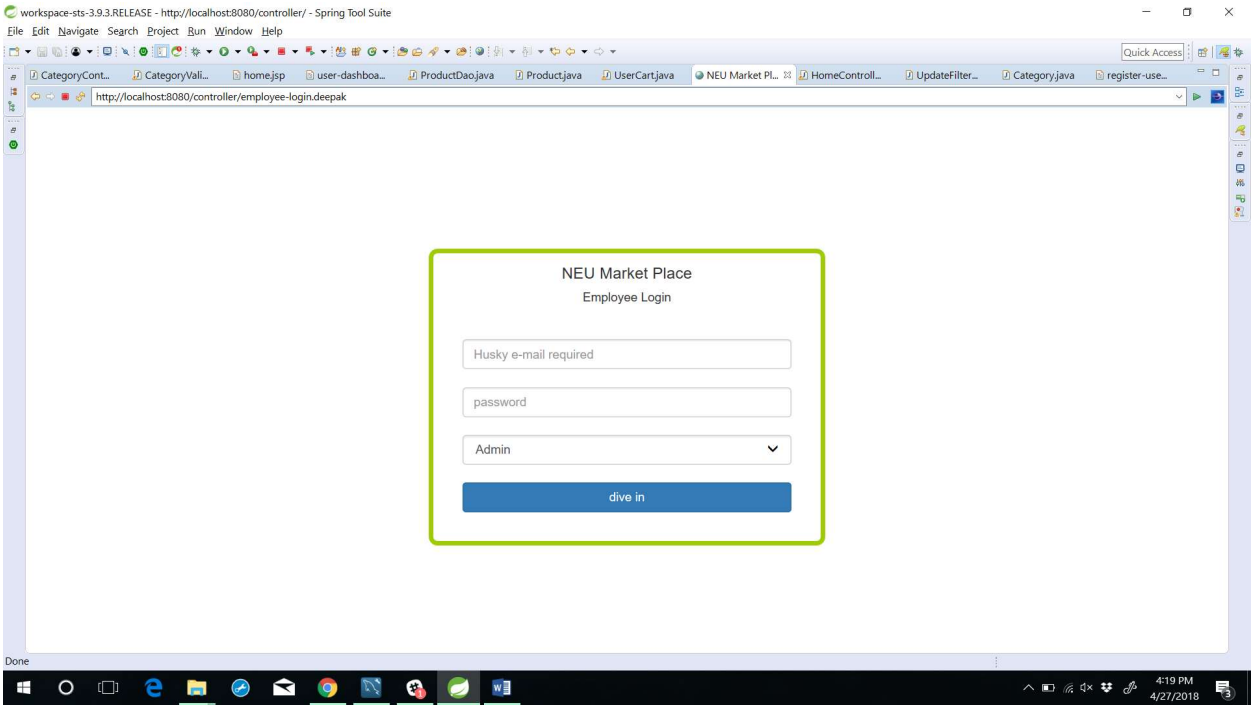
- User is able to register using husky email id only
- Can sell or purchase products
- Can view their previous orders
- can view cart for any previous placed products
- can view/download the PDF of order invoice
- can view/download Excel file of previous placed orders

Screenshots

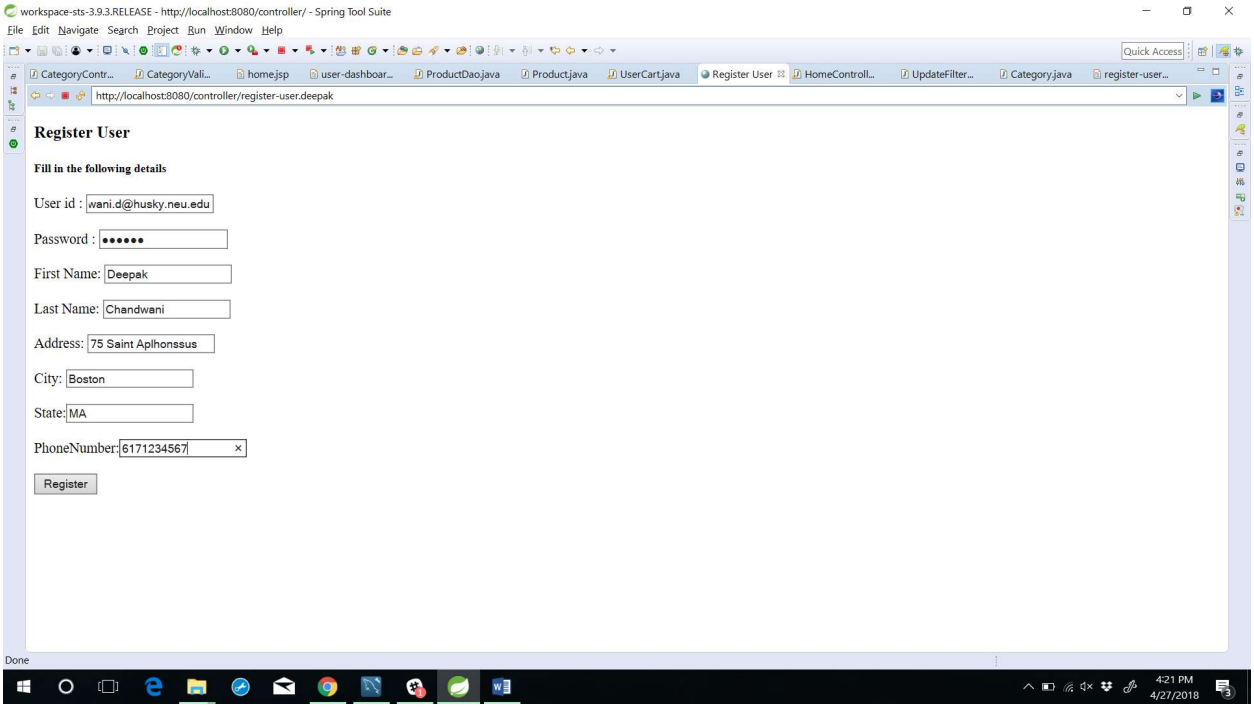
Home Page (User Login)



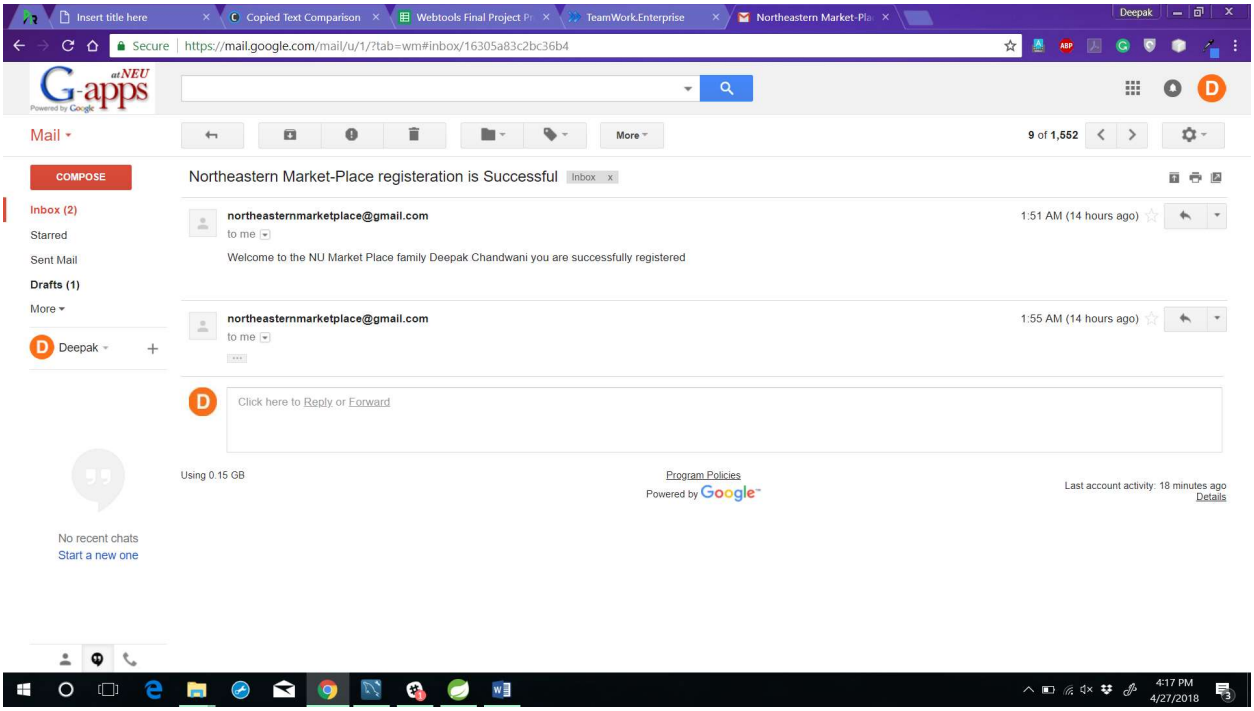
Employee Login



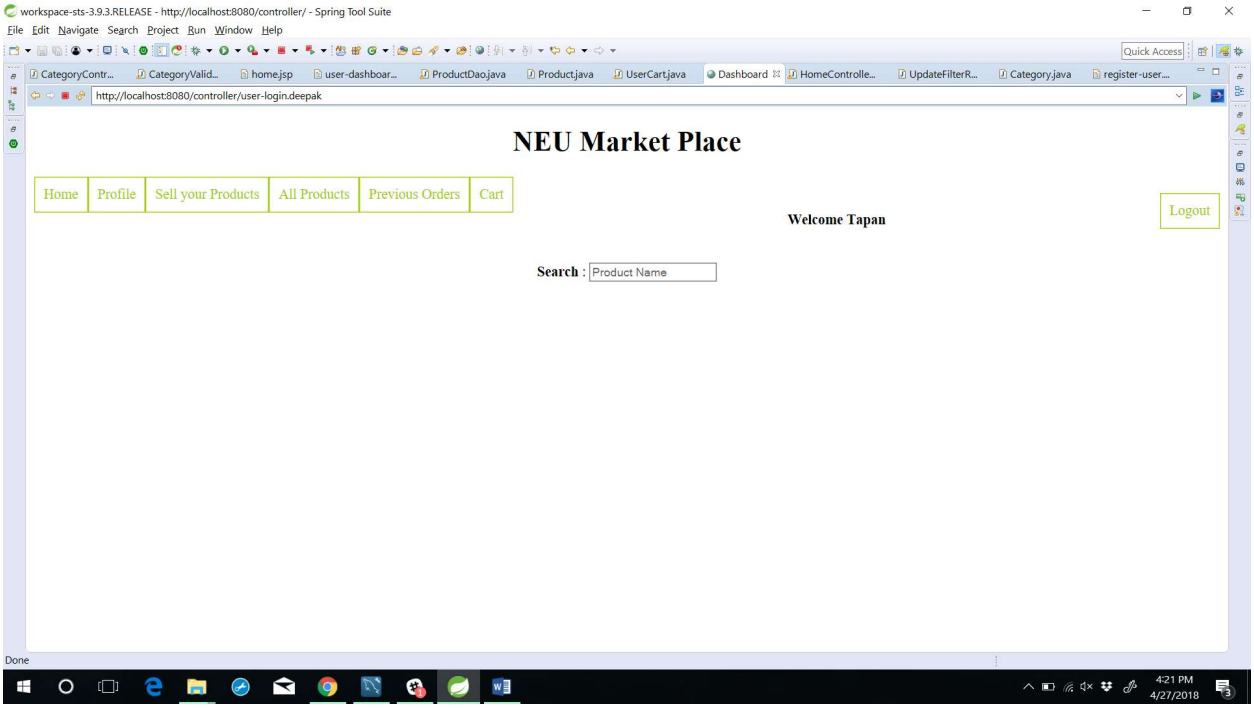
User Register Form



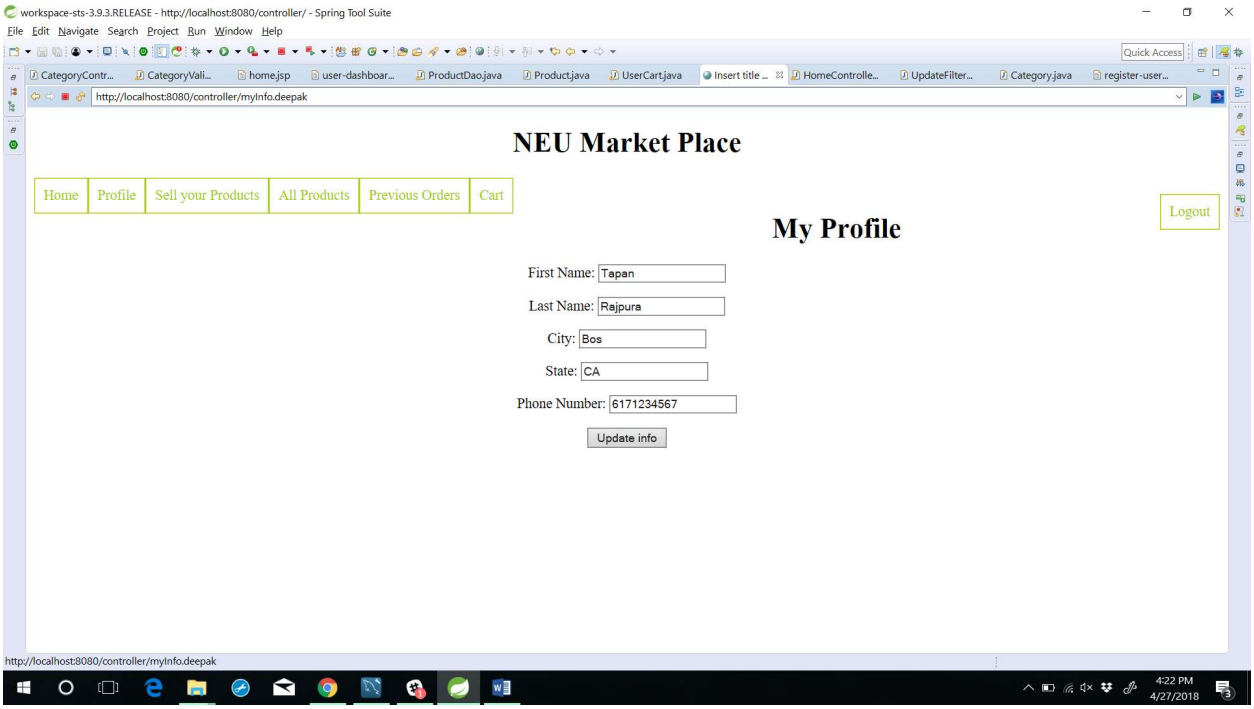
Successfully Registered



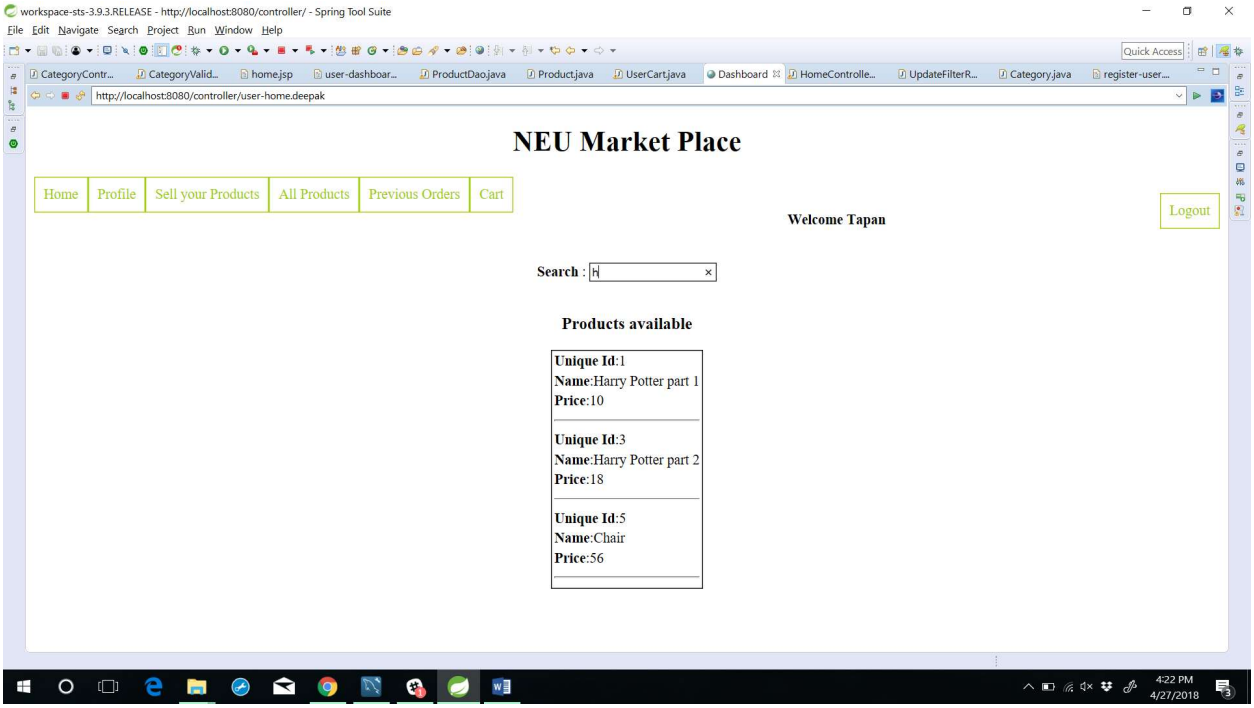
User Home Page



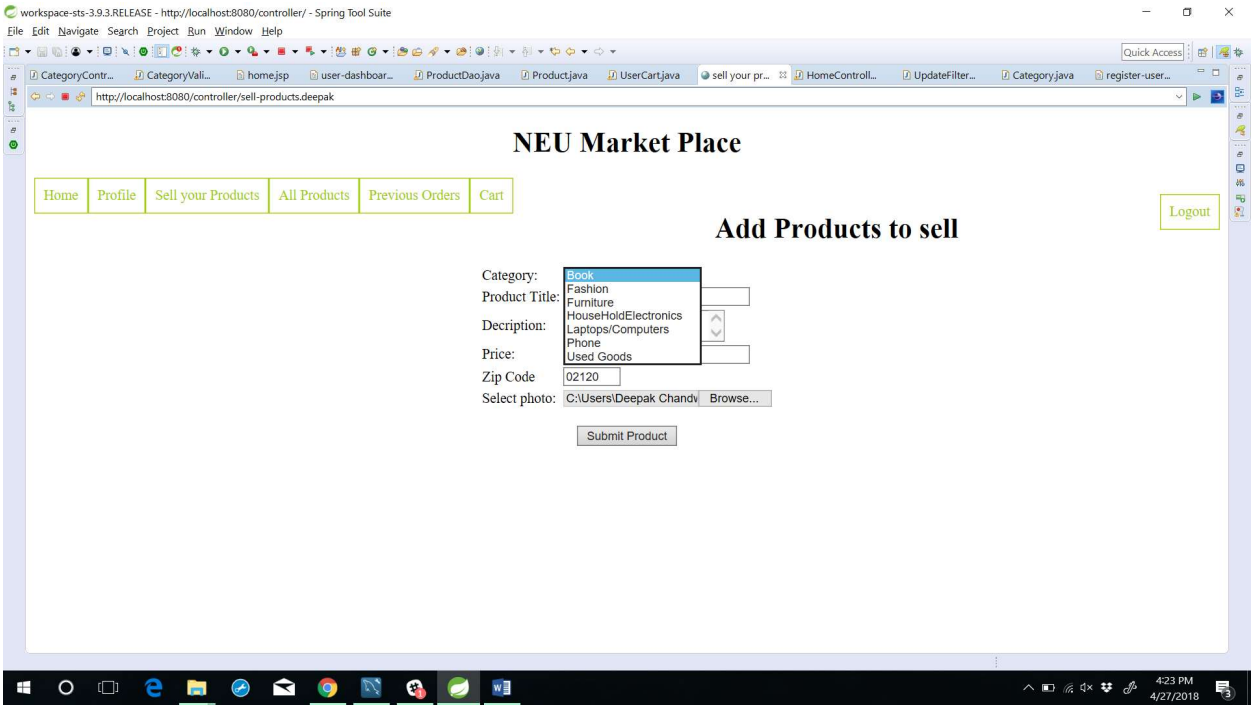
User Profile to view and update



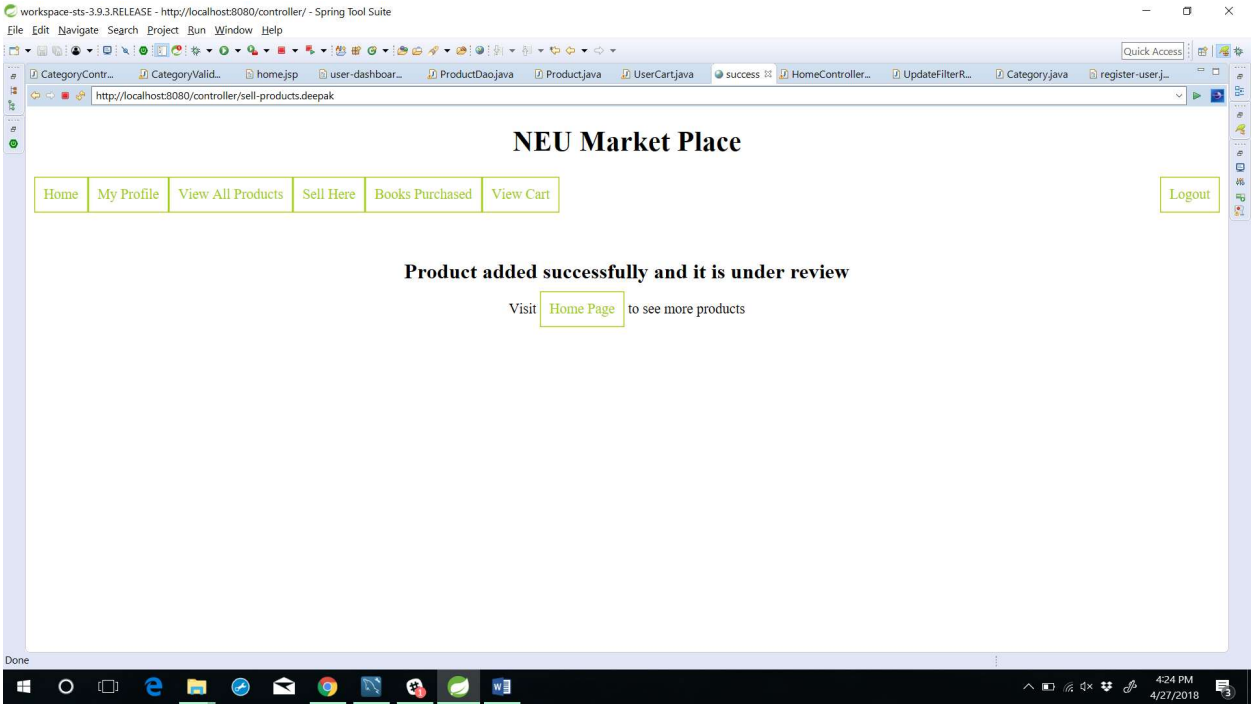
Search Products



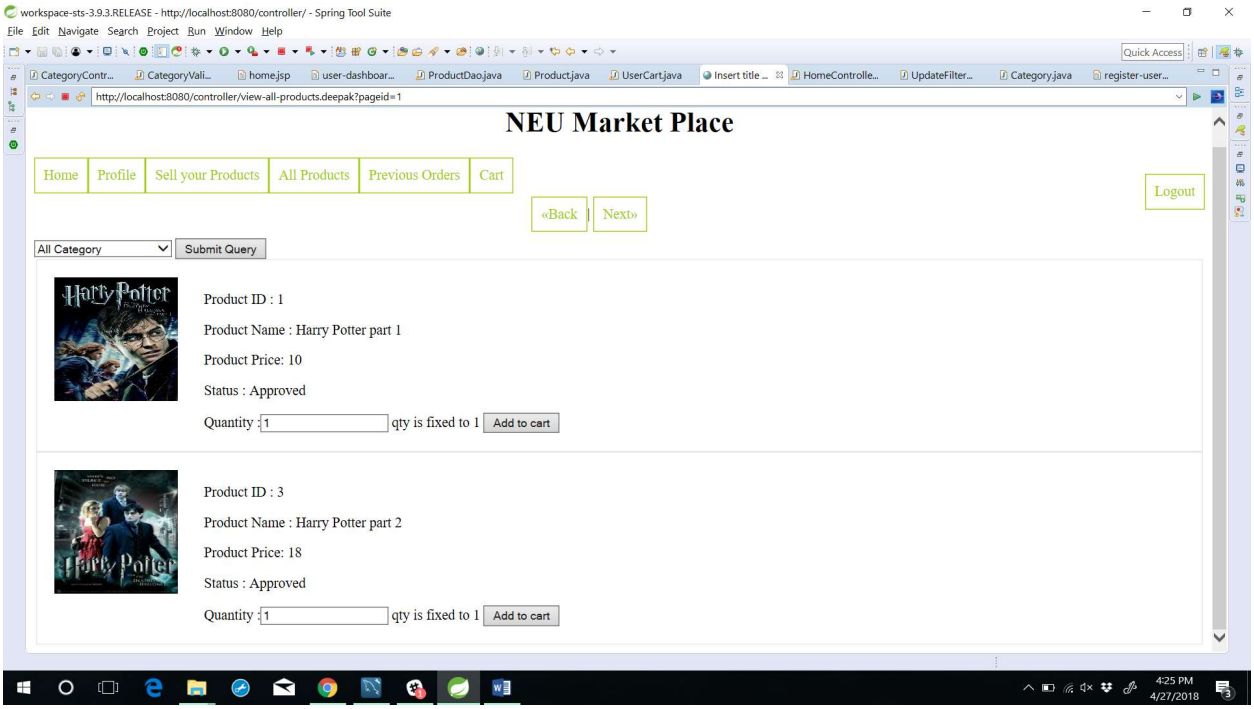
Sell Your Products



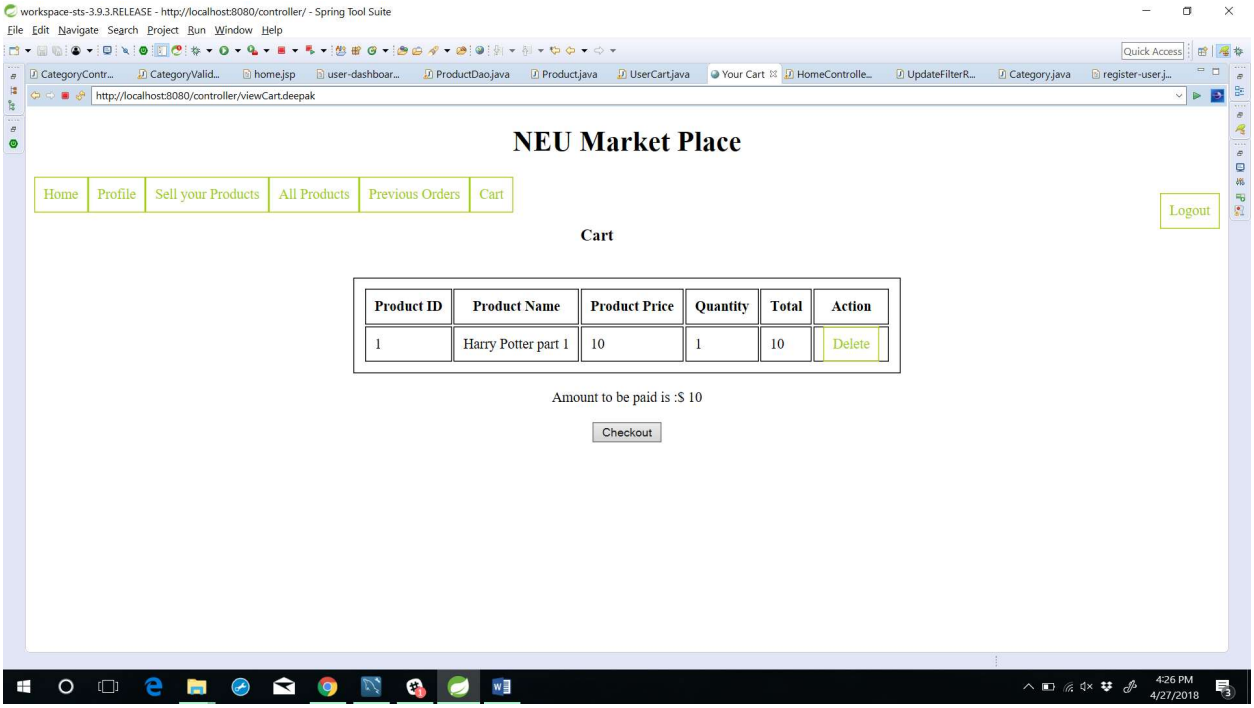
New Product Created



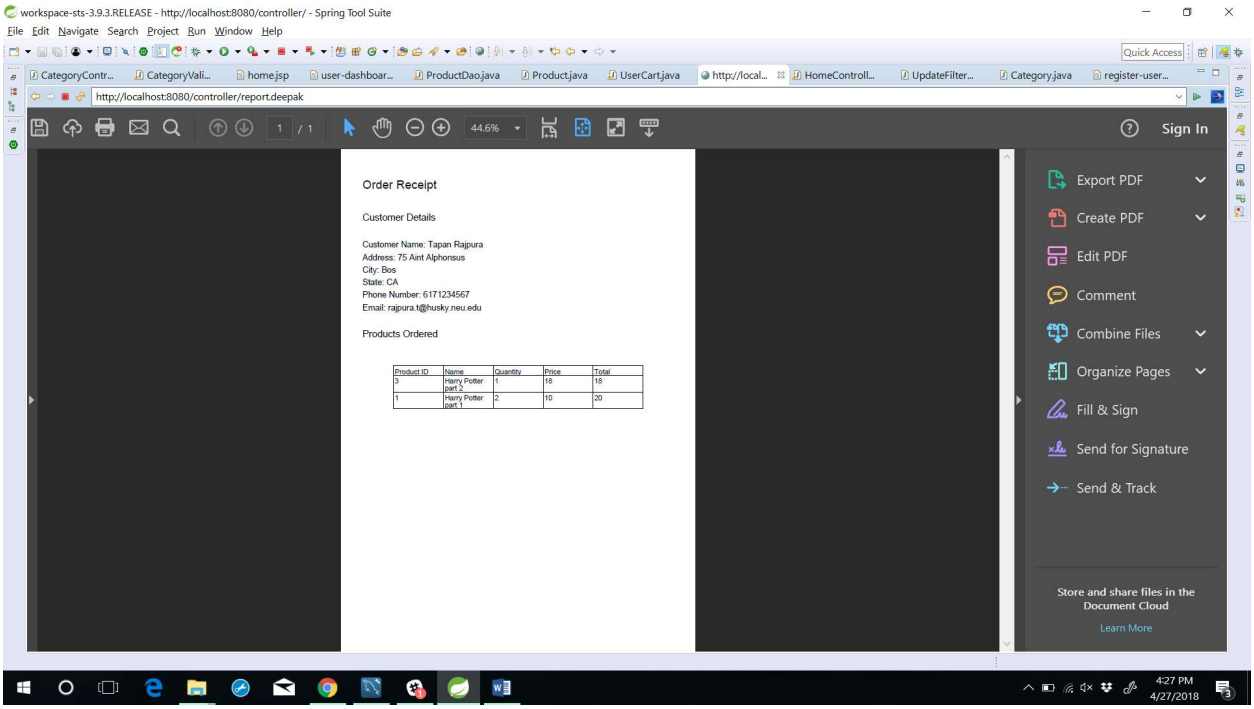
Approved Products for Sale



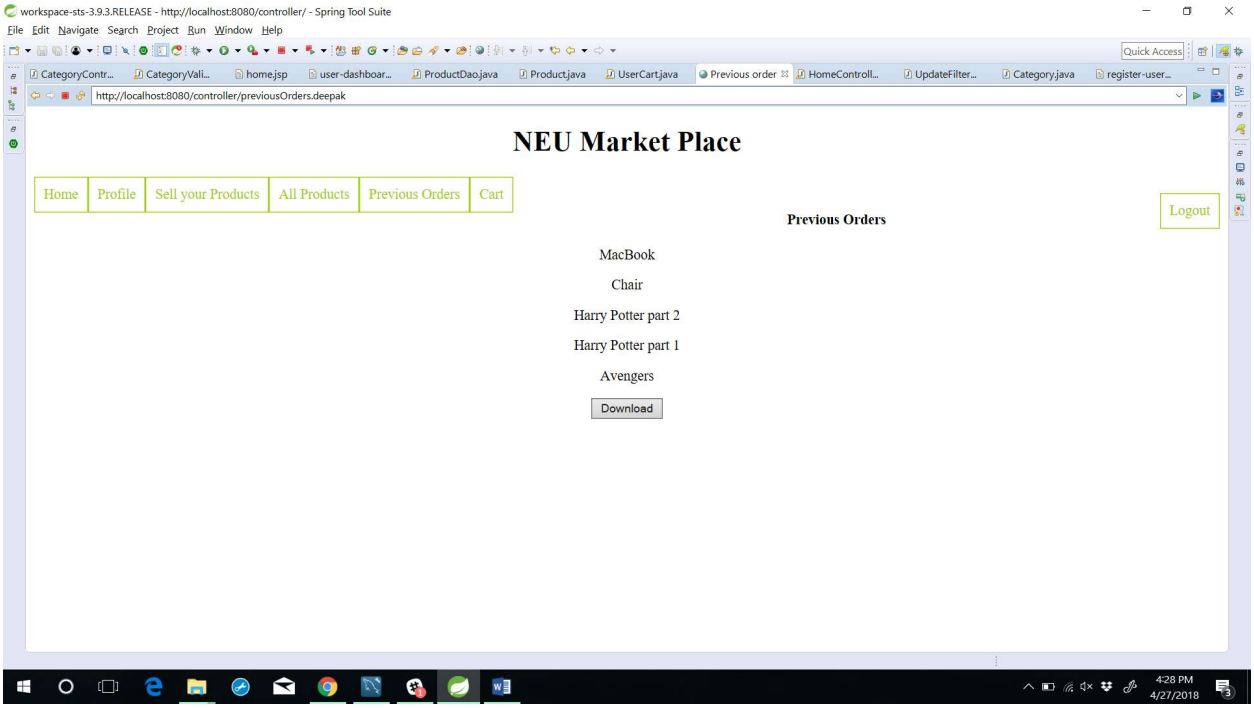
User Cart



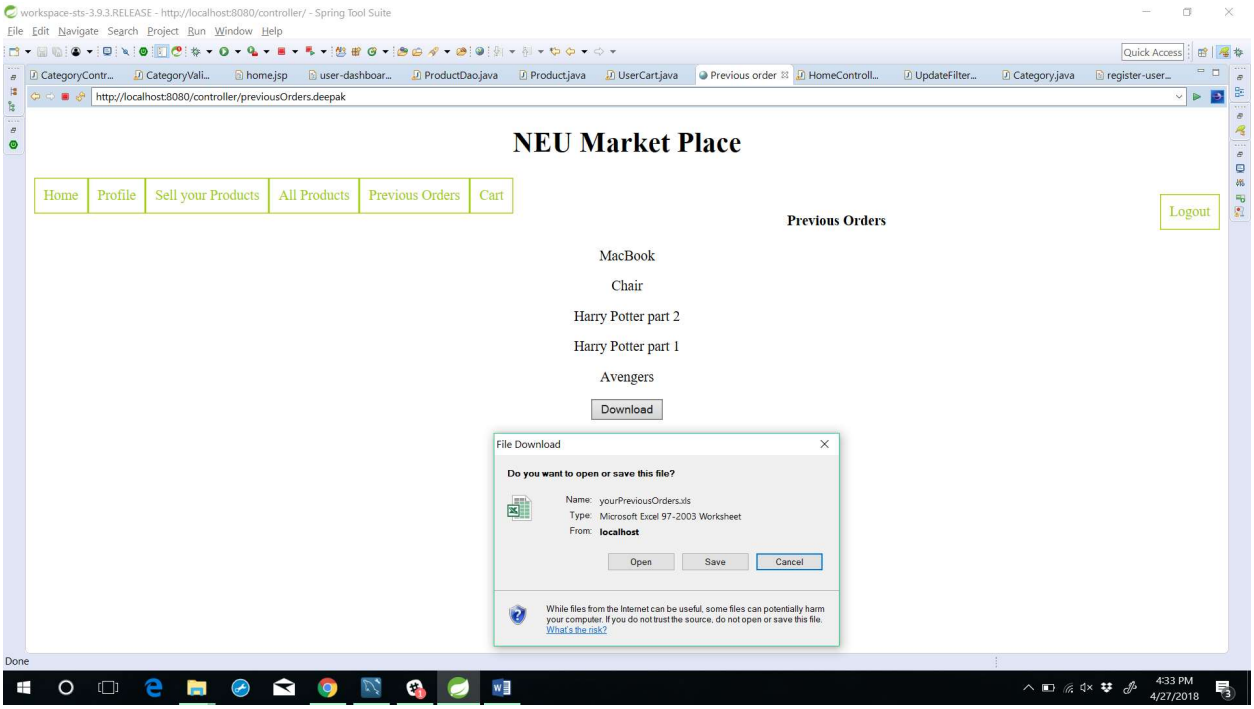
PDF Invoice



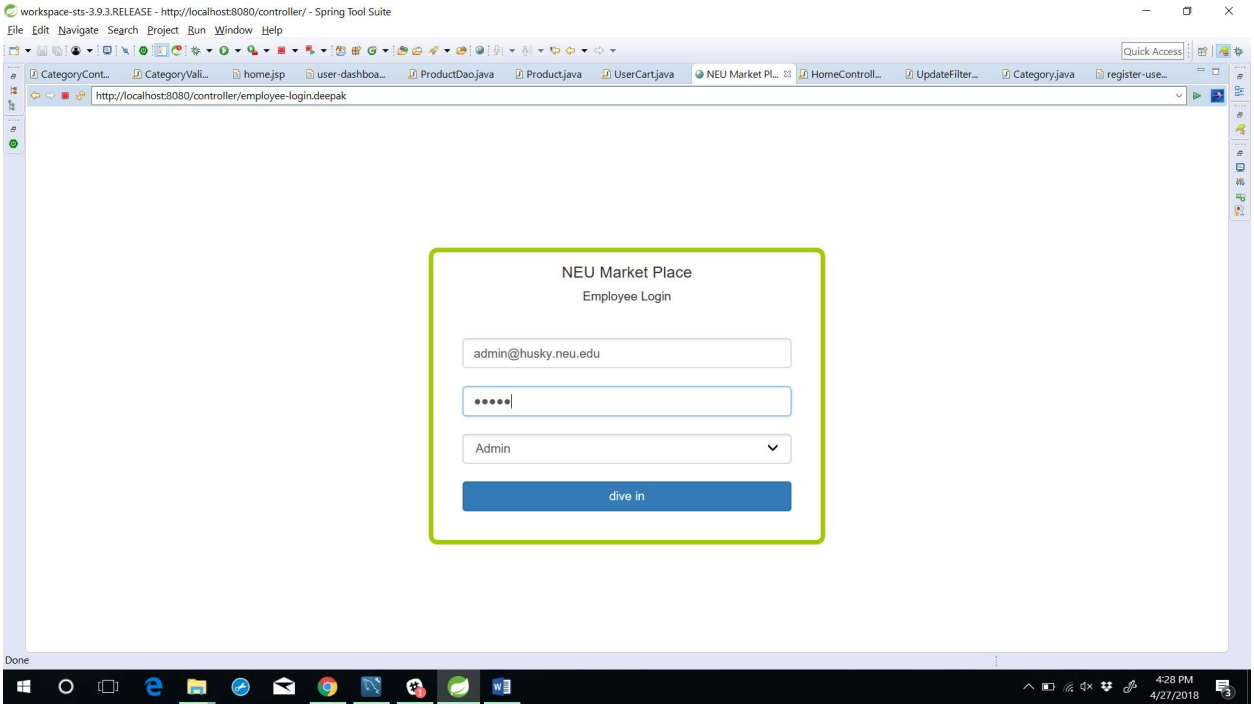
Previous Orders



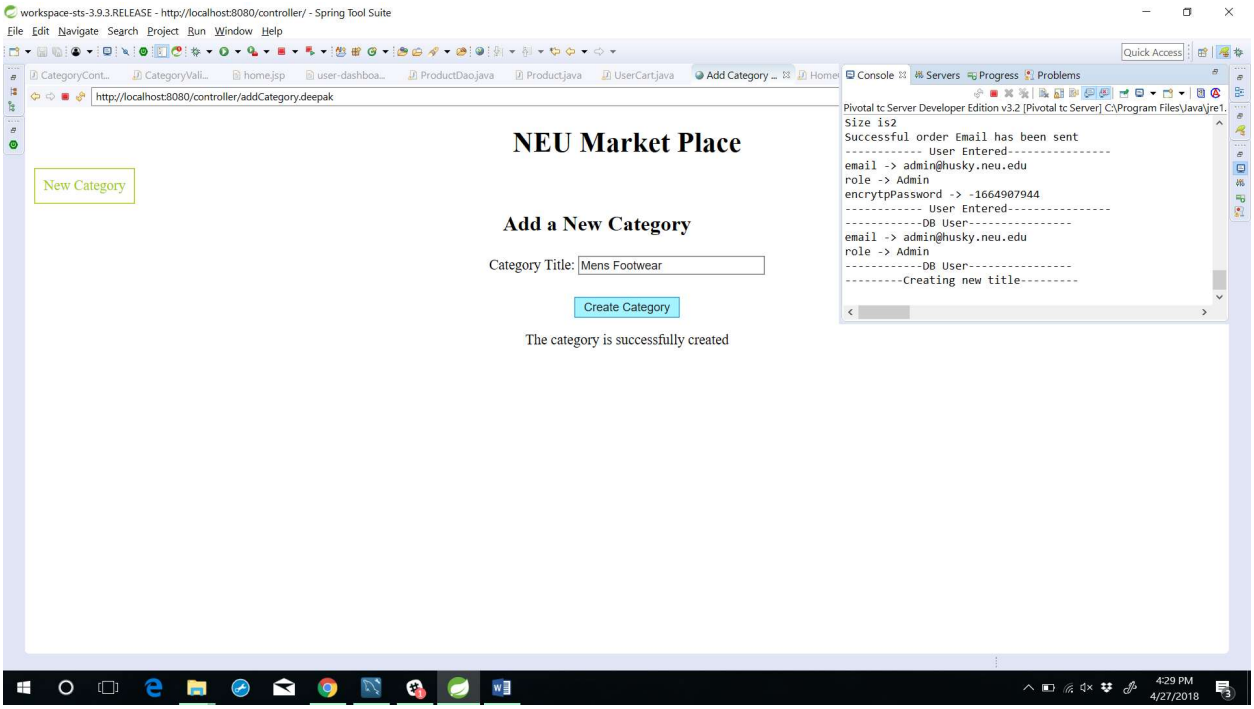
Excel Report



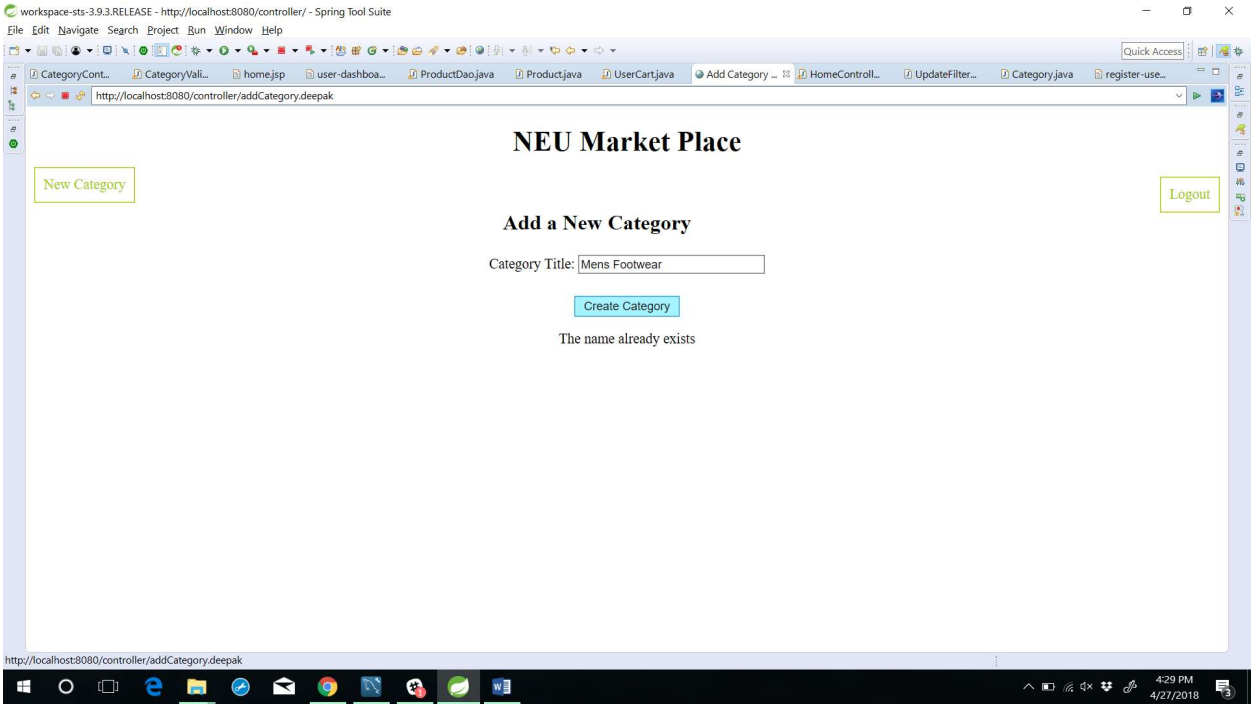
Admin Login



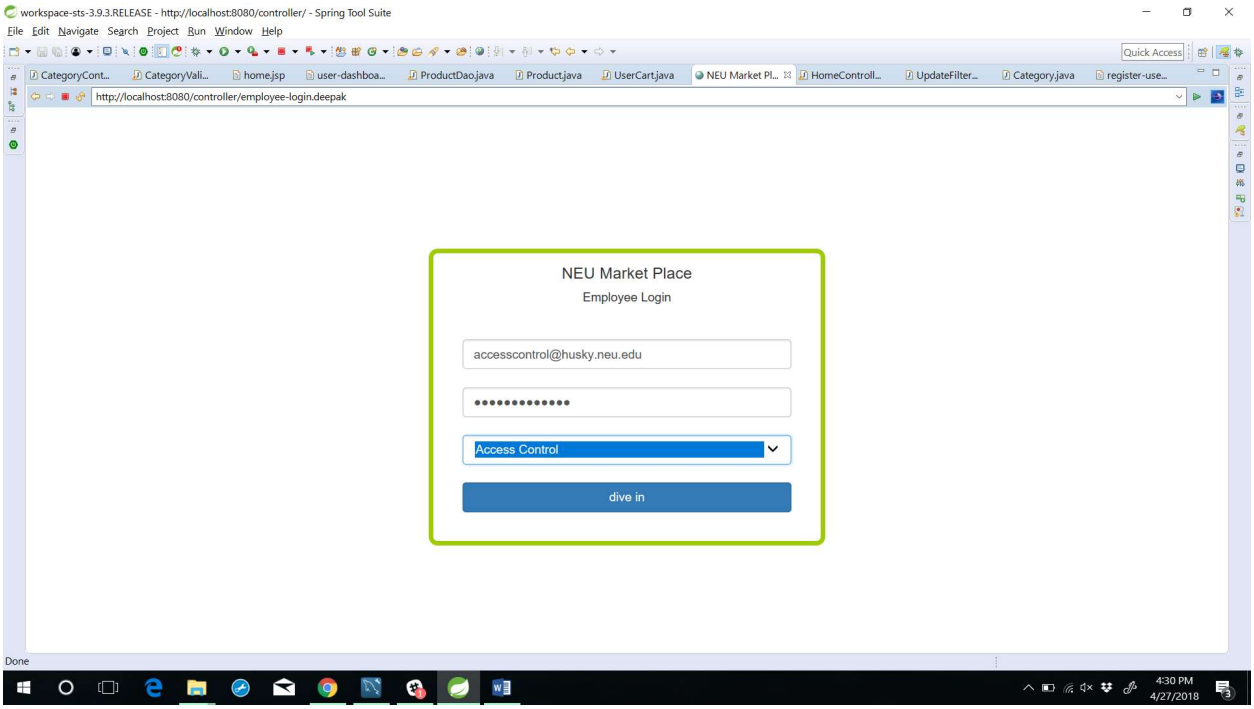
New Category Created



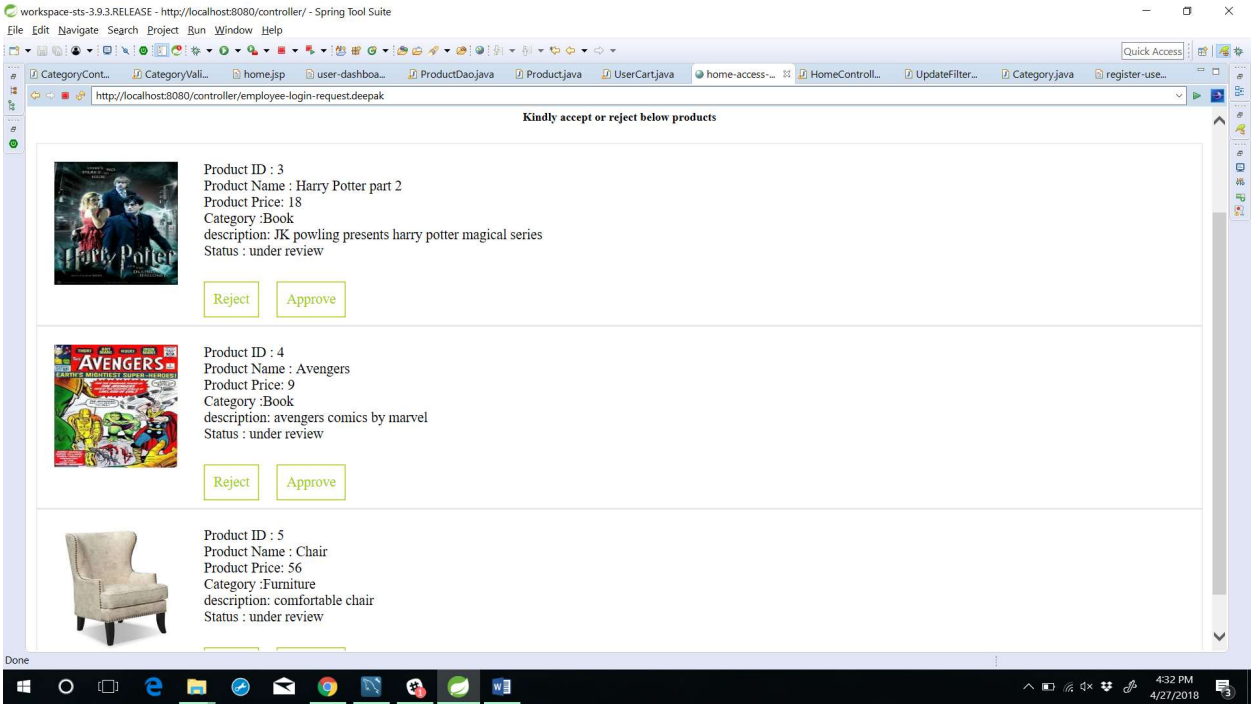
Validation in Category



Access Control Login



Access Control Home Page



APPENDIX

Controller Code

Home Controller

```
@Controller
public class HomeController {

	@Autowired
	Employee employee;

	@Autowired
	UserDao userDao;

	@Autowired
	User user;

	@Autowired
	Product product;

	@Autowired
	ProductDao productDao;

	@Autowired
	CartDao	cartDao;

	@Autowired
	CategoryDAO categoryDao;


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

	/**
	 * Simply selects the home view to render by returning its name.
	 */

	@RequestMapping(value = "/home.deepak", method = RequestMethod.GET)
	public String home(Locale locale, Model model) {
		return "home";
	}




	@RequestMapping(value = "/register-user.deepak", method = RequestMethod.GET)
	public String registerUser(Locale locale, Model model) {
		return "register-user";
	}

	@RequestMapping(value = "/user-home.deepak", method = RequestMethod.GET)
	public String userHome(Locale locale, Model model) {
		return "user-dashboard";
	}
```

```
	@RequestMapping(value = "/employee-login.deepak", method = RequestMethod.GET)
	public String employeeLogin(Locale locale, Model model) {
		return "employee-login";
	}

	@RequestMapping(value = "/dashboard.deepak", method = RequestMethod.GET)
	public String userDashboard(Locale locale, Model model) {
		return "user-dashboard";
	}

	@RequestMapping(value = "/rejectProduct.deepak", method = RequestMethod.GET)
	public ModelAndView rejectProduct(HttpServletRequest request) {
		//int result = 0;
		List<Product>result =
productDao.rejectProduct(Integer.parseInt(request.getParameter("id")));
		if(result.size() > 0)
			return new ModelAndView("home-access-control",
"underReviewedProductsList", result);
		else {
			return new ModelAndView("home-access-control",
"underReviewedProductsList", result);
		}
	}

	@RequestMapping(value = "/approveProduct.deepak", method = RequestMethod.GET)
	public ModelAndView approveProduct(HttpServletRequest request) {
		//int result = 0;
		List<Product> result =
productDao.approveProduct(Integer.parseInt(request.getParameter("id")));
		if(result.size() > 0)
			return new ModelAndView("home-access-control",
"underReviewedProductsList", result);
		else {
			return new ModelAndView("home-access-control",
"underReviewedProductsList", result);
		}

	}


	@RequestMapping(value= "/employee-login-request.deepak", method =
RequestMethod.POST)
	protected ModelAndView handleRequestInternal(
		HttpServletRequest request,
		HttpServletResponse response) throws Exception {




	 ModelAndView mv = null;
		HttpSession mySession = request.getSession(true);
	 //String action = request.getParameter("action");
	 String userName = request.getParameter("employee-email");
	 String pas = request.getParameter("employee-password");
	 String role = request.getParameter("employee-role");
```

```
        Employee employeeCredentials = new Employee();
        employeeCredentials.setUserName(userName);
        employeeCredentials.setPassword(encrytpPassword);
        employeeCredentials.setRole(role);

        //boolean success = false;
        Employee employeeResult = EmployeeDAO.checkEmployee(employeeCredentials);

        try {
        if(!(employeeResult.equals(null)))
        {


             mySession.setAttribute("employee", employeeResult);
             if("Admin".equalsIgnoreCase(employeeResult.getRole()))
             {

                   //-----------------Programmatic Security-----------------
                   String authorization = request.getHeader("Authorization");
                if (authorization == null) {
                   askForPassword(response);
                   }
                else
                {
                            String userInfo = authorization.substring(5).trim();
                            //System.out.println("authorization ->"+new
String(Base64.getDecoder().decode(authorization)));
                            String nameAndPassword = new
String(Base64.getDecoder().decode(userInfo));
                            int index = nameAndPassword.indexOf(":");
                            String user = nameAndPassword.substring(0, index);
                            String password = nameAndPassword.substring(index+1);

                      if(user.equalsIgnoreCase("admin@husky.neu.edu")){

                            mv = new ModelAndView("home-admin");
                             }
                      else {
                                   mv = new ModelAndView("notAuthorized");
                             }
                }
             }else if("Access Control".equalsIgnoreCase(employeeResult.getRole()))
             {
                   List<Product> productsList =
productDao.getUnderReviewedProducts();
                   System.out.println("------------productList size----------------
");
                   System.out.println("productListSize -> "+productsList.size());
                   System.out.println("------------productList size----------------
");
                   mv = new ModelAndView("home-access-control",
"underReviewedProductsList", productsList);
             }
```

```
 }else
 {
      mv = new ModelAndView("employee-login", "invalidCredentials", "true");
 }
 }catch(NullPointerException e)
 {
      System.out.println("Nullpointer Exception "+ e.toString());
      mv = new ModelAndView("employee-login", "invalidCredentials", "true");
 }

 return mv;
}

private void askForPassword(HttpServletResponse response) {
      // SC_UNAUTHORIZED is 401
      response.setStatus(response.SC_UNAUTHORIZED);
      response.setStatus(response.SC_UNAUTHORIZED);
      response.setHeader
      ("WWW-Authenticate",
      "BASIC realm BASIC realm=\"numarketplace.com\"");
      }



@RequestMapping(value= "user-login.deepak", method = RequestMethod.POST)
protected ModelAndView loginUser(
     HttpServletRequest request,
     HttpServletResponse response) throws Exception {
 ModelAndView mv = null;

 String userid = request.getParameter("user-email");
 String password = request.getParameter("user-password");
 System.out.println("userEmail -> "+userid);
 System.out.println("password -> "+password);

 //String pass = encPas(password);
 String encrytpPassword = MyEncryption.encPas(password);
 System.out.println("password -> "+encrytpPassword);
 User userp = userDao.getUser(userid, encrytpPassword);

 if(userp == null)
 {
      return new ModelAndView("home","errorMessage",true);
 }

 HttpSession session = request.getSession();
 session.setAttribute("user", userp);
 session.setAttribute("role", "student");

 List<Product> prod = productDao.getProducts();
      //mv = new ModelAndView("user-dashboard", "prodList", prod);
      mv = new ModelAndView("user-dashboard");
 return mv;
```

```
	}

	@RequestMapping(value = "/register-user-success.deepak", method =
RequestMethod.POST)
	protected ModelAndView registerUser(
		HttpServletRequest request,
		HttpServletResponse response) throws Exception {

		 ModelAndView mv = null;

	 String userid = request.getParameter("userid");
	 String password = request.getParameter("password");
	 String fname = request.getParameter("fname");
	 String lname = request.getParameter("lname");
	 String city = request.getParameter("city");
	 String state= request.getParameter("state");
	 String phoneNumber = request.getParameter("phonenumber");
	 String address = request.getParameter("address");

	 if("husky.neu.edu".equalsIgnoreCase((userid.split("@"))[1]))
	 {
		 System.out.println("It is husky email");

		 //String pass = encPas(password);
		String encrytpPassword = MyEncryption.encPas(password);


		user.setUserid(userid);
		user.setPassword(encrytpPassword);
		user.setFname(fname);
		user.setLname(lname);
		user.setAddress(address);
		user.setCity(city);
		user.setState(state);
		user.setPhoneNumber(phoneNumber);

		userDao.addToDb(user);
		String message = "Welcome to the NU Market Place family "+fname+"
"+lname+" you are successfully registered";
		String subject = "Northeastern Market-Place, your registeration is
Successful,";
		sendEmail(userid, message, subject);


		 mv = new ModelAndView("home");
	 }else
	 {
		 System.out.println("Husky email required");
		 mv = new ModelAndView("register-user", "msg", "Husky email required to
register");
	 }


	 return mv;
```

```
    }

    @RequestMapping(value="/myInfo.deepak" , method=RequestMethod.GET)
    public ModelAndView getMyInfo(HttpServletRequest request)
                                    throws Exception{
            ModelAndView mv = null;

            try{
                    mv= new ModelAndView("user-info");
            }
            catch(Exception e){
                    System.out.println("AdopterController - getMyInfo");
            }
            return mv;
    }

    @RequestMapping(value="/myInfo.deepak" , method=RequestMethod.POST)
    public ModelAndView updateMyInfo(HttpServletRequest request)
                                    throws Exception{
            ModelAndView mv = null;

            String fname = request.getParameter("fname");
            String lname = request.getParameter("lname");
            String city = request.getParameter("city");
            String state = request.getParameter("state");
            String phoneNumber = request.getParameter("phoneNumber");
            User usp = (User)request.getSession().getAttribute("user");

            System.out.println("--------------fname--------------"+ usp.getFname());
            usp.setFname(fname);
            usp.setLname(lname);

            usp.setCity(city);
            usp.setState(state);
            usp.setPhoneNumber(phoneNumber);

            User usp1 = (User)request.getSession().getAttribute("user");
            System.out.println("--------------New fname--------------
"+usp1.getFname());

            userDao.updateUserDetails(usp);

            List<Product> prod = productDao.getProducts();

            mv = new ModelAndView("user-dashboard", "prodList", prod);

            return mv;
    }

    @RequestMapping(value="/logout.deepak", method = RequestMethod.GET)
    public String goToLoginPage(HttpServletRequest req){
            HttpSession session = req.getSession();
            session.invalidate();
            return "home";
```

```
	}

	@RequestMapping(value="/logout.deepak", method = RequestMethod.POST)
	public String goToLogin(HttpServletRequest req){
		HttpSession session = req.getSession();
		session.invalidate();
		return "home";
	}

	@RequestMapping(value="/getProductList.deepak", method=RequestMethod.POST,
produces = MediaType.ALL_VALUE)
	public @ResponseBody String getProductListKeyword(HttpServletRequest
request,HttpServletResponse response)
	{
		String keyword = request.getParameter("val");
		String tmp ="";
		if(!keyword.isEmpty())
		{
			List<Product> productList = productDao.getProductList(keyword);
			if(productList.size()==0)
			{
				tmp = "<table><tr><td><h3>Product is not
available</h3></td></tr></table>";
			}
			else
			{
				tmp="<h3>Products available</h3>";
				tmp += "<table style='border: 1px solid black'>";
				for(int i=0;i<productList.size();i++)
				{

					tmp += "<tr><td><b>Unique
Id</b>:"+productList.get(i).getProductId()+"</td></tr>";
					tmp +=
"<tr><td><b>Name</b>:"+productList.get(i).getProductName()+"</td></tr>";
					tmp +=
"<tr><td><b>Price</b>:"+productList.get(i).getProductPrice()+"</td></tr>";
					tmp += "<tr><td><hr/></td></tr>";
				}

				tmp+="</table>";
			}
		}

		return tmp;
	}

	@RequestMapping(value="/view-all-products.deepak" , method=RequestMethod.GET)
	public ModelAndView viewAllProducts(HttpServletRequest request)
						throws Exception{
		ModelAndView mv = null;

		int pagenumber= Integer.parseInt(request.getParameter("pageid"));
		HttpSession session = request.getSession();
		session.setAttribute("pageid", pagenumber);
```

```
		List<Product> prod = productDao.getLimitedProducts(pagenumber);
		List<Category> catList = categoryDao.getCategory();;
		Map<String, Object> map = new HashMap<String, Object>();
		map.put("products", prod);
		//map.put("catList", catList);
		session.setAttribute("catList", catList);
		mv = new ModelAndView("viewProducts", "map", map);
		return mv;
	}

	@RequestMapping(value="/categorySpecificResult.deepak" ,
method=RequestMethod.GET)
	public ModelAndView categorySpecificResult(HttpServletRequest request)
						throws Exception{
		ModelAndView mv = null;
		String selectedCategory = request.getParameter("selectedCategory");
		System.out.println("selectedCategory "+selectedCategory);
		Map<String, Object> map = new HashMap<String, Object>();
		if(selectedCategory.equalsIgnoreCase("All Category")){
			List<Product> filteredProducts = productDao.getProducts();
			map.put("products", filteredProducts);
		}else if(selectedCategory != null){
			List<Product> filteredProducts =
productDao.getCategoryProducts(selectedCategory);
			map.put("products", filteredProducts);
		}
		mv = new ModelAndView("viewProducts", "map", map);
		return mv;
	}


	@RequestMapping(value="/view-next-products.deepak" , method=RequestMethod.GET)
	public ModelAndView viewNextProducts(HttpServletRequest request)
						throws Exception{
		ModelAndView mv = null;

		HttpSession session = request.getSession();
		int pagenumber = (Integer) session.getAttribute("pageid");
		pagenumber++;

		session.setAttribute("pageid", pagenumber);
		List<Product> prod = productDao.getLimitedProducts(pagenumber);
		if(prod.size() == 0)
		{
			System.out.println("Setting pageid to 1");
			session.setAttribute("pageid", 1);
			prod = productDao.getLimitedProducts(1);
		}
		Map<String, Object> map = new HashMap<String, Object>();
		map.put("products", prod);
		mv = new ModelAndView("viewProducts", "map", map);
		System.out.println(session.getAttribute("pageid"));
		return mv;
```

```
	}


	@RequestMapping(value="/view-previous-products.deepak" ,
method=RequestMethod.GET)
	public ModelAndView viewPreviousProducts(HttpServletRequest request)
					throws Exception{
		ModelAndView mv = null;

		List<Product> prod = null;
		HttpSession session = request.getSession();
		int pagenumber = (Integer) session.getAttribute("pageid");
		if(pagenumber == 1)
		{
			prod = productDao.getLimitedProducts(1);

		}
		else
		{
			pagenumber--;
			session.setAttribute("pageid", pagenumber);
			prod = productDao.getLimitedProducts(pagenumber);
		}
		Map<String, Object> map = new HashMap<String, Object>();
		map.put("products", prod);
		mv = new ModelAndView("viewProducts", "map", map);
		System.out.println(session.getAttribute("pageid"));
		return mv;

	}

	@RequestMapping(value="/sell-products.deepak", method=RequestMethod.GET)
	public ModelAndView sellProducts(HttpServletRequest request, Model model)
					throws Exception{
		List<Category> categoryList = categoryDao.getCategory();
		System.out.println("categoryList -> "+categoryList.size());
		//Map<String, List<ProductPojo>> map = new HashMap<String,
List<ProductPojo>>();
		//request.getSession().setAttribute("catList", categoryList);
		ModelAndView mv = new ModelAndView();
		mv.addObject("catList", categoryList);
		mv.addObject("product", new Product());
		mv.setViewName("seller-addProducts");
		return mv;
	}

	@RequestMapping(value="/sell-products.deepak", method=RequestMethod.POST)
	public String handleUpload(@ModelAttribute("product") Product product,
HttpServletRequest request) {
		String UPLOAD_PATH = "C:\\Users\\Deepak Chandwani\\Documents\\workspace-
sts-3.9.3.RELEASE\\NEUMarketPlace\\src\\main\\resources\\images\\";
		try {

			System.out.println("name-> "+product.getProductName());
			System.out.println("price-> "+product.getProductPrice());
```

```
                    System.out.println("description-> "+product.getDescription());
                    System.out.println("cat-> "+product.getCategory());


                                                        // We need to transfer to a file
                                CommonsMultipartFile photoInMemory =
product.getPhoto();

                                String fileName =
photoInMemory.getOriginalFilename();
                                // could generate file names as well

                                File localFile = new File(UPLOAD_PATH, fileName);

                                // move the file from memory to the file

                                photoInMemory.transferTo(localFile);
                                product.setImageFilePath(UPLOAD_PATH+fileName);
                                product.setStatus("under review");
                                HttpSession session = request.getSession();
                                User user = (User) session.getAttribute("user");
                            product.setUser(user);
                            //session.getAttribute("role");
                                System.out.println("File is stored at" +
localFile.getPath());



                                //-----------------------getting user location from
zip----------------------
//                              String zipCode = String.valueOf(product.getZip());
//                              LatLngResponse res = getLatLng(zipCode);
//                                if(res.getStatus().equals("OK"))
//                                {
//                                 for(Result result : res.getResults())
//                                 {
//                                  System.out.println("Lattitude of address is :"
+result.getGeometry().getLocation().getLat());
//                                  System.out.println("Longitude of address is :" +
result.getGeometry().getLocation().getLng());
//                                  System.out.println("Location is " +
result.getGeometry().getLocation_type());
//                                 }
//                                }
//                                else
//                                {
//                                 System.out.println(res.getStatus());
//                                }
                                //-----------------------getting user location from
zip----------------------


                                //product.setFilename(fileName);
                                boolean success = productDao.create(product);
                                System.out.println("---Product has been added.---");
```

```
                                if(success)
                                        return "success";


            } catch (IllegalStateException e) {
                    System.out.println("*** IllegalStateException: " +
e.getMessage());
            } catch (IOException e) {
                    // TODO Auto-generated catch block
                    System.out.println("*** IOException: " + e.getMessage());
            } catch (Exception e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
            }

            System.out.println("Inside FileController POST method");
            return "error";
    }



    @RequestMapping(value="/toCart.deepak", method=RequestMethod.GET)
    public ModelAndView addToCart(HttpServletRequest request,
@ModelAttribute("cart") UserCart cart)
                    throws Exception{

            ModelAndView mv = null;
            Product pj =
productDao.getProductFromId(Integer.parseInt(request.getParameter("id")));
            User up = (User)request.getSession().getAttribute("user");
            cart.setUser(up);
            cart.setProduct(pj);
            cart.setQuantity(Integer.parseInt(request.getParameter("qty")));

            boolean status = cartDao.checkInCart(up.getUserid(), pj.getProductId());
            if(status)
            {
                    cartDao.updateQuantity(cart);
            }
            else {

            cartDao.addProducttoCart(cart);
            }
            int pagenumber = (Integer)request.getSession().getAttribute("pageid");
            List<Product> prod = productDao.getLimitedProducts(pagenumber);
            Map<String, Object> map = new HashMap<String, Object>();
            map.put("products", prod);
            map.put("successMessage", "true");

            return new ModelAndView("viewProducts", "map", map);
    }

    @RequestMapping(value="/viewCart.deepak", method=RequestMethod.GET)
    public ModelAndView viewCart(HttpServletRequest request)
                    throws Exception{
```

```
		ModelAndView mv = null;
		User up = (User)request.getSession().getAttribute("user");
		System.out.println(up.getUserid());
		List<UserCart> listcart = cartDao.getProductListFromId(up.getUserid());
		System.out.println("Size is"+listcart.size());
		return new ModelAndView("viewCart", "listcart", listcart);
	}

	@RequestMapping(value="/deleteFromCart.deepak", method=RequestMethod.GET)
	public ModelAndView deleteProductsFromCart(HttpServletRequest request)
			throws Exception{

		ModelAndView mv = null;
		User pj = (User)request.getSession().getAttribute("user");
		String prodid = request.getParameter("id");
		List<UserCart> listcart =
cartDao.deleteProductsFromCart(pj.getUserid(),Integer.parseInt(prodid));
		return new ModelAndView("viewCart", "listcart", listcart);
	}

	@RequestMapping(value="/addOrder.deepak", method=RequestMethod.POST)
	public ModelAndView addOrder(HttpServletRequest request)
			throws Exception{

		ModelAndView mv = null;
		User up = (User)request.getSession().getAttribute("user");
		List<UserCart> listcart = cartDao.getProductListFromId(up.getUserid());
		System.out.println("Size is"+listcart.size());

		return new ModelAndView("finalReview", "listcart", listcart);
	}

	@RequestMapping(value="/success.deepak", method=RequestMethod.POST)
	public ModelAndView success(HttpServletRequest request)
			throws Exception{

		ModelAndView mv = null;
		User up = (User)request.getSession().getAttribute("user");
		List<UserCart> listcart = cartDao.getProductListFromId(up.getUserid());
		System.out.println("Size is"+listcart.size());
		String streetAddress = request.getParameter("address");
		String city = request.getParameter("city");
		String state = request.getParameter("state");
		String pnumber = request.getParameter("phonenumber");
		String email = request.getParameter("email");
		request.getSession().setAttribute("address", streetAddress);
		request.getSession().setAttribute("city", city);
		request.getSession().setAttribute("state", state);
		request.getSession().setAttribute("pnumber", pnumber);
		request.getSession().setAttribute("email", email);
		request.getSession().setAttribute("listcart", listcart);
		cartDao.updateStatus(up.getUserid());

		if(listcart.size()>0)
```

```
		{
			User user = (User) request.getSession().getAttribute("user");
			String msg = "Dear "+user.getFname()+" "+user.getLname()
				+ "\n Your order has been placed successfully\n"
				+ "Thankyou for shopping with NEU Market Place";
			String subject = "NEU Market Place, Order placed successfully";
			sendEmail(user.getUserid(), msg, subject);

			System.out.println("Successful order Email has been sent");
		}
		return new ModelAndView("orderSuccess", "listcart", listcart);
	}

	@RequestMapping(value="/previousOrders.deepak", method=RequestMethod.GET)
	public ModelAndView booksPurchased(HttpServletRequest request)
			throws Exception{


		ModelAndView mv = null;

		User up= (User) request.getSession().getAttribute("user");
		List<UserCart> listCart = cartDao.getOrderedProducts(up);
		Set<String> cartSet = new HashSet<String>();
		for(UserCart lc : listCart)
		{
			cartSet.add(lc.getProduct().getProductName());
		}
		List<String> newList = new ArrayList<String>(cartSet);
		request.getSession().setAttribute("newList", newList);

		return new ModelAndView("orderedProducts","newList", newList);
	}

	@RequestMapping(value = "/export.deepak", method = RequestMethod.POST)
	public ModelAndView exportData(HttpServletRequest request, XlsReport excel) {
		List<String> newList =
(List<String>)request.getSession().getAttribute("newList");
		return new ModelAndView(excel, "newList", newList);
	}

	@RequestMapping(value = "/report.deepak", method = RequestMethod.POST)
	public ModelAndView generateReport(HttpServletRequest request, XlsReport
excel) {
		Map<String,Object> model = new HashMap<String, Object>();
		model.put("listcart", request.getSession().getAttribute("listcart"));
		model.put("address", request.getSession().getAttribute("address"));
		model.put("city", request.getSession().getAttribute("city"));
		model.put("state", request.getSession().getAttribute("state"));
		model.put("pnumber", request.getSession().getAttribute("pnumber"));
		model.put("email", request.getSession().getAttribute("email"));
		model.put("user", request.getSession().getAttribute("user"));
		return new ModelAndView(new PdfReport(), model);
	}
```

```
	public void sendEmail(String e_mail, String msg, String subject) {
		try {
			Email email = new SimpleEmail();
			email.setHostName("smtp.googlemail.com");
			email.setSmtpPort(465);
			String no_reply_email = "northeasternmarketplace@gmail.com";
			String pas = "NUMarketPlace2017";
			email.setAuthenticator(new DefaultAuthenticator(no_reply_email,
pas));
			email.setSSLOnConnect(true);
			email.setFrom(no_reply_email); // This user email does not

	// exist
			email.setSubject(subject);
			email.setMsg(msg); // Retrieve email from the DAO and send this
			email.addTo(e_mail);
			email.send();
		} catch (EmailException e) {
			System.out.println("Email cannot be sent "+e.toString());
		}
	}
}
```

Category Controller

```
@Controller
public class CategoryController {

		@Autowired
		@Qualifier("categoryValidator")
		CategoryValidator categoryValidator;

		@Autowired
		@Qualifier("categoryDao")
		CategoryDAO categoryDAO;



		@RequestMapping(value = "/addCategory.deepak", method =
RequestMethod.POST)
		public ModelAndView addCategory(@ModelAttribute("category") Category
category, BindingResult result) throws Exception {

			categoryValidator.validate(category, result);

			if (result.hasErrors()) {
				return new ModelAndView("category-form", "category",
category);
			}

			try {
				Category cat = categoryDAO.get(category.getTitle());

				if(cat == null)
				{
					category = categoryDAO.create(category.getTitle());
				}
				else {
					return new ModelAndView("category-form",
"errorMessage", "The name already exists");
				}

			} catch (CategoryException e) {
				System.out.println(e.getMessage());
			}
			return new ModelAndView("category-form", "success", "The category
is successfully created");

		}

		@RequestMapping(value="/addCategory.deepak", method = RequestMethod.GET)
		public ModelAndView initializeForm() throws Exception {
			return new ModelAndView("category-form", "category", new
Category());
		}

}
```

PDF Report

```
public class PdfReport extends AbstractPdfView {


	@Override

	protected void buildPdfDocument(Map<String, Object> model,
com.lowagie.text.Document document, PdfWriter writer,

		HttpServletRequest request, HttpServletResponse response) throws
Exception {

		document.add(new Paragraph("Order Receipt",
FontFactory.getFont(FontFactory.HELVETICA, 20)));

		document.add(Chunk.NEWLINE);

		User user = (User) model.get("user");

		String address =(String)model.get("address");

		String city =(String)model.get("city");

		String state =(String)model.get("state");

		String phoneNumber =(String)model.get("pnumber");

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

		List<UserCart> listCart = (List<UserCart>)model.get("listcart");


		document.add(new Paragraph("Customer Details",
FontFactory.getFont(FontFactory.HELVETICA, 16)));

		document.add(Chunk.NEWLINE);

		document.add(new Paragraph("Customer Name: " + user.getFname() + " " +
user.getLname(), FontFactory.getFont(FontFactory.HELVETICA, 14)));

		document.add(new Paragraph("Address: "+ address,
FontFactory.getFont(FontFactory.HELVETICA, 14)));

		document.add(new Paragraph("City: "+ city,
FontFactory.getFont(FontFactory.HELVETICA, 14)));

		document.add(new Paragraph("State: "+ state,
FontFactory.getFont(FontFactory.HELVETICA, 14)));

		document.add(new Paragraph("Phone Number: "+ phoneNumber,
FontFactory.getFont(FontFactory.HELVETICA, 14)));

		document.add(new Paragraph("Email: "+ email,
FontFactory.getFont(FontFactory.HELVETICA, 14)));
```

```
            document.add(Chunk.NEWLINE);

            document.add(new Paragraph("Products Ordered",
FontFactory.getFont(FontFactory.HELVETICA, 16)));

            document.add(Chunk.NEWLINE);
            PdfPTable table = new PdfPTable(5);

            table.addCell("Product ID");
            table.addCell("Name");
            table.addCell("Quantity");
            table.addCell("Price");
            table.addCell("Total");

            for(UserCart lc : listCart)
            {
                  table.addCell(String.valueOf(lc.getProduct().getProductId()));
                  table.addCell(lc.getProduct().getProductName());
                  table.addCell(String.valueOf(lc.getQuantity()));
                  table.addCell(String.valueOf(lc.getProduct().getProductPrice()));

      table.addCell(String.valueOf(lc.getQuantity()*lc.getProduct().getProductPrice(
)));
            }

            document.add(table);

      }



}
```

XLS Report

```
public class XlsReport extends AbstractXlsView {

	@Override
	protected void buildExcelDocument(Map<String, Object> model, Workbook
workbook, HttpServletRequest request,
		HttpServletResponse response) throws Exception {
		// TODO Auto-generated method stub
		response.setHeader("Content-disposition", "attachment;
filename=\"yourPreviousOrders.xls\"");

		@SuppressWarnings("unchecked")
		List<String> list = (List<String>) model.get("newList");

		Sheet sheet = workbook.createSheet("previousOrders");

		Row header = sheet.createRow(0);
		header.createCell(0).setCellValue("Name");

		int rowNum = 1;
		for (String data : list) {
			Row row = sheet.createRow(rowNum++);
			row.createCell(0).setCellValue(data);

		}

	}
}
```

POJO Code

Category

```
@Entity
@Table(name="categoryRecords")
public class Category {

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

        @Column(name="title", unique=true, nullable = false)
  private String title;



  public Category(String title) {
    this.title = title;
  }

  public Category() {
  }

        public long getCategoryId() {
                return categoryId;
        }

        public void setCategoryId(long categoryId) {
                this.categoryId = categoryId;
```

```
        }

        public String getTitle() {
                return title;
        }

        public void setTitle(String title) {
                this.title = title;
        }

        @Override
        public String toString(){
                return title;
        }
}
```

Employee

```java
@Entity
@Table(name="employeeRecords")
public class Employee{

	@Id
	@Column(name="employeeId", nullable = false, unique = true)
	private int employeeId;

	@Column(name = "firstName", nullable = false)
	private String firstName;

	@Column(name="lastName", nullable=false)
	private String lastName;

	@Column(name="role", nullable=false)
	private String role;

	@Column(name = "userName", nullable=false)
	private String userName;

	@Column(name="password", nullable=false)
	private String password;



	public Employee() {

	}

	public int getEmployeeId() {
		return employeeId;
	}
	public void setEmployeeId(int employeeId) {
		this.employeeId = employeeId;
	}
	public String getFirstName() {
		return firstName;
	}
	public void setFirstName(String firstName) {
		this.firstName = firstName;
	}
	public String getLastName() {
		return lastName;
	}
	public void setLastName(String lastName) {
		this.lastName = lastName;
	}
	public String getRole() {
		return role;
	}
	public void setRole(String role) {
		this.role = role;
	}
```

```java
	public String getUserName() {
		return userName;
	}
	public void setUserName(String userName) {
		this.userName = userName;
	}
	public String getPassword() {
		return password;
	}
	public void setPassword(String password) {
		this.password = password;
	}


}
```

Product

```
@Entity
@Table(name="productRecords")
public class Product {

      @Id
      @GeneratedValue(strategy=GenerationType.IDENTITY)
      @Column(name="productId")
      private int productId;

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

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

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

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

      // transient variable will not be persisted
      private transient CommonsMultipartFile photo;

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

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

      @ManyToOne
      private User user;

      @OneToMany(mappedBy="product")
      private Set<UserCart> cart;

      @Column(name="latitude")
      private double latitude;

      @Column(name="longitude")
      private double longitude;

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





      public double getLatitude() {
            return latitude;
```

```
}
public void setLatitude(double latitude) {
	this.latitude = latitude;
}
public double getLongitude() {
	return longitude;
}
public void setLongitude(double longitude) {
	this.longitude = longitude;
}
public int getZip() {
	return zip;
}
public void setZip(int zip) {
	this.zip = zip;
}
public Set<UserCart> getCart() {
	return cart;
}
public void setCart(Set<UserCart> cart) {
	this.cart = cart;
}
public User getUser(){
	return user;
}
public void setUser(User user) {
	this.user = user;
}
public String getStatus() {
	return status;
}
public void setStatus(String status) {
	this.status = status;
}

public int getProductId() {
	return productId;
}
public void setProductId(int productId) {
	this.productId = productId;
}
public String getProductName() {
	return productName;
}
public void setProductName(String productName) {
	this.productName = productName;
}
public int getProductPrice() {
	return productPrice;
}
public void setProductPrice(int productPrice) {
	this.productPrice = productPrice;
}
public CommonsMultipartFile getPhoto() {
	return photo;
```

```
	}
	public void setPhoto(CommonsMultipartFile photo) {
		this.photo = photo;
	}
	public String getImageFilePath() {
		return imageFilePath;
	}
	public void setImageFilePath(String imageFilePath) {
		this.imageFilePath = imageFilePath;
	}
	public String getDescription() {
		return description;
	}
	public void setDescription(String description) {
		this.description = description;
	}
	public String getCategory() {
		return category;
	}
	public void setCategory(String category) {
		this.category = category;
	}


}
```

User

```
package com.deepak.pojo;

import java.util.Set;

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


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

	@Id
	@Column(name="userid")
	String userid;

	@Column(name="password", unique=true, nullable=false)
	String password;

	@Column(name="fname", unique=false, nullable=false)
	String fname;
```

```
@Column(name="lname", unique=false, nullable=false)
String lname;

@Column(name="city", unique=false, nullable=false)
String city;

@Column(name="state", unique=false, nullable=false)
String state;

@Column(name="phoneNumber", unique=false, nullable=false)
String phoneNumber;

@Column(name="address", unique=false, nullable=false)
String address;

public String getCity() {
	return city;
}

public void setCity(String city) {
	this.city = city;
}

public String getState() {
	return state;
}

public void setState(String state) {
```

```
        this.state = state;
}

public String getPhoneNumber() {
        return phoneNumber;
}

public void setPhoneNumber(String phoneNumber) {
        this.phoneNumber = phoneNumber;
}

public String getAddress() {
        return address;
}

public void setAddress(String address) {
        this.address = address;
}

@OneToMany(mappedBy="user", cascade=CascadeType.ALL)
private Set<UserCart> cart;

public Set<UserCart> getCart() {
        return cart;
}

public void setCart(Set<UserCart> cart) {
        this.cart = cart;
}
```

```
public String getUserid() {
        return userid;
}

public void setUserid(String userid) {
        this.userid = userid;
}

public String getPassword() {
        return password;
}

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

public String getFname() {
        return fname;
}

public void setFname(String fname) {
        this.fname = fname;
}

public String getLname() {
        return lname;
}
```

```
	public void setLname(String lname) {
		this.lname = lname;
	}

}
```

User Cart

```
package com.deepak.pojo;


import java.io.Serializable;


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


@Entity
@Table(name="usercart")
public class UserCart implements Serializable {



	@Id
	@GeneratedValue(strategy=GenerationType.AUTO)
	private int cartid;


	@Column(name="status")
	private int status=0;



	@ManyToOne
	@JoinColumn(name="product_id")
```

```
private Product product;



@ManyToOne
@JoinColumn(name="user_id")
private User user;

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



public int getQuantity() {
	return quantity;
}

public void setQuantity(int quantity) {
	this.quantity = quantity;
}

public User getUser() {
	return user;
}

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

public Product getProduct() {
	return product;
```

```
        }

        public void setProduct(Product product) {
                this.product = product;
        }

}
```