

Spring 2020 CSYE6220 Project Report

Subject: Enterprise software Design

(Underlining tools utilized: Spring MVC architecture, Java EE, Hibernate, MySQL)

Instructor: Prof. Yusuf Ozbek

***Submitted as partial fulfilment of the semester project by:
Deepak Lokwani (001316769)***

***Teaching Assistants:
Divya Taneja
Suryanarayana Malladi
Yan Lyu
Zidi Xu***

NU Marketplace – An Ecommerce portal

It is an enterprise application to trade items online for the registered members.

Abstraction:

The idea is to propose and develop an online platform for anyone and everyone to buy and sell items. The application has two roles for the users:

- a. Seller
- b. Customer

Different privileges and functionalities come with the different roles.

Following functionalities can be performed with each role:

A. Seller:

- i. Register themselves as the seller with a unique email address
- ii. Login and Logout functionality using password
- iii. Add products
- iv. View their products
- v. Edit product Details
- vi. Remove products
- vii. View their orders
- viii. View profile information
- ix. Update profile information using both username and password
- x. View reviews

B. Customer:

- i. Register themselves as the customers with a unique email address
- ii. Login and Logout functionality using password
- iii. Search products with filtering, sorting and paging feature
- iv. Can add and products to carts and maintain in cart
- v. Edit items in cart
- vi. Remove items from cart
- vii. Checkout the items in cart
- viii. Payment gateway with credit card validations
- ix. Write reviews for the products(one review per product)
- x. View orders

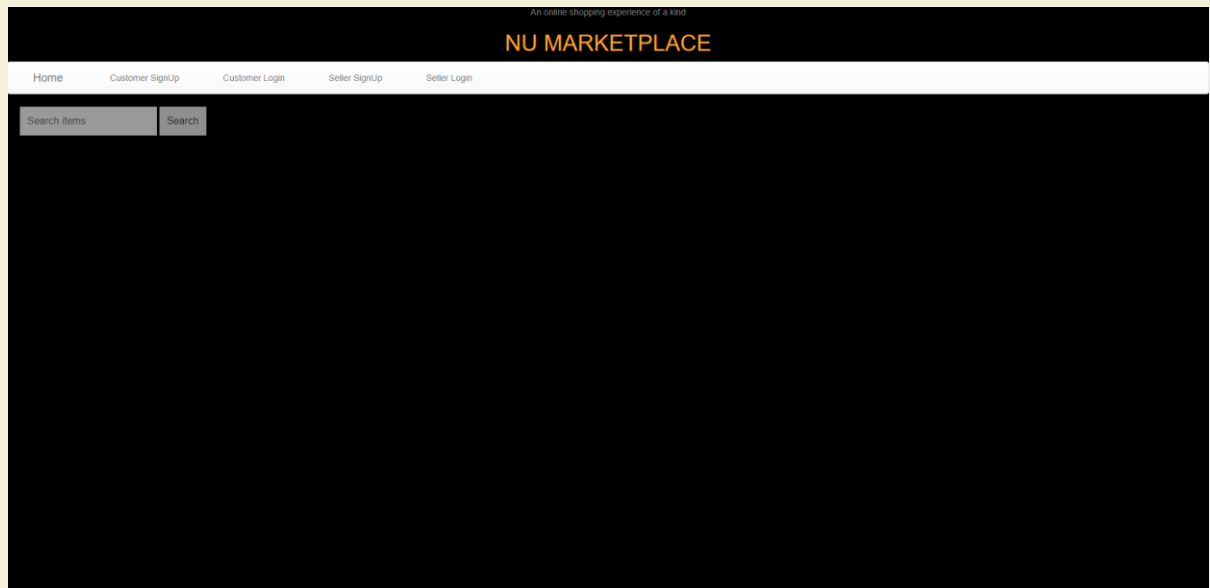
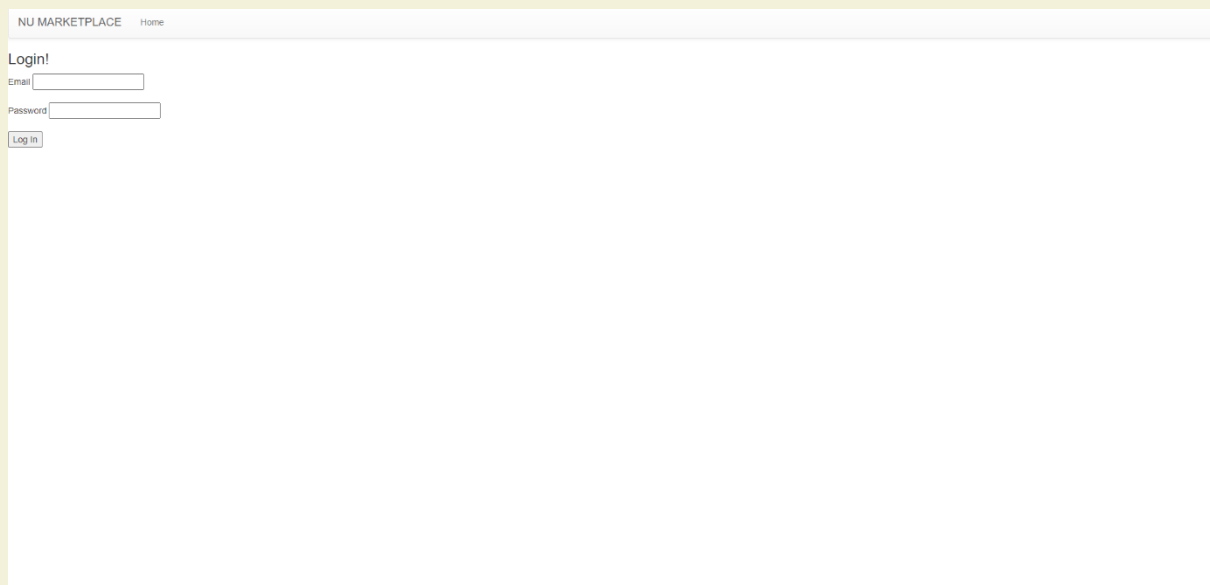
Technology Stack:

It is a Spring MVC application with following technologies used at the respective ends

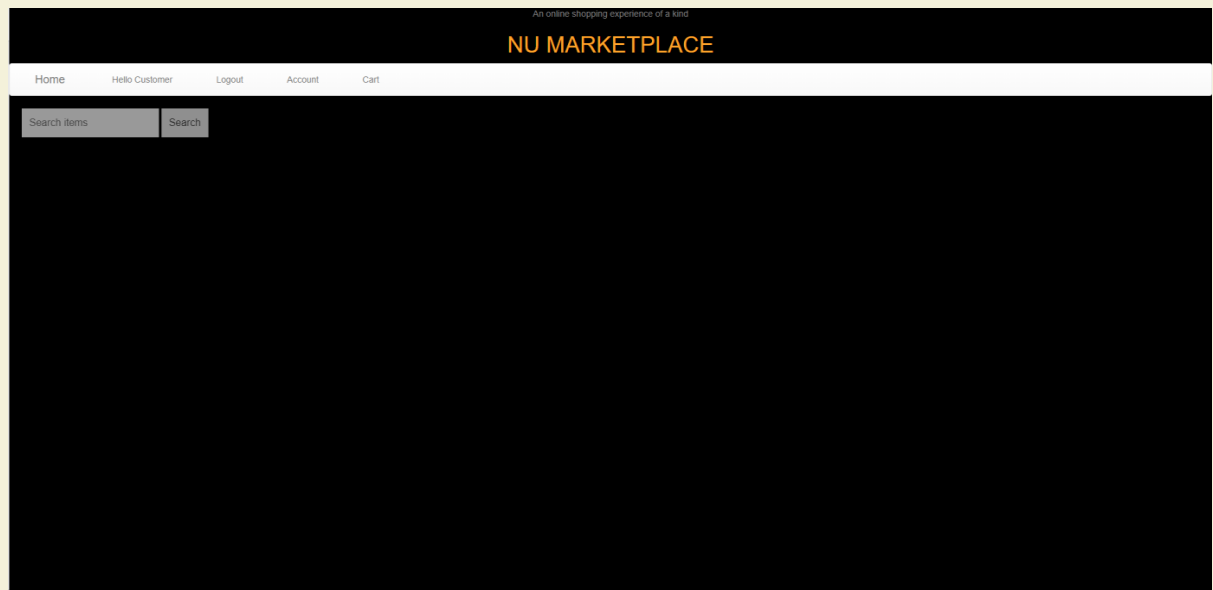
Particulars	Tools/Technologies/Languages
Application	Spring MVC
Server end	Java EE
Client end	HTML, CSS, JSP
Database design	MySQL
Other Frameworks	Hibernate, JQuery, JPA
Tools	Eclipse, MySQL Workbench, Web browser

Following concepts were used as a part of the learning:

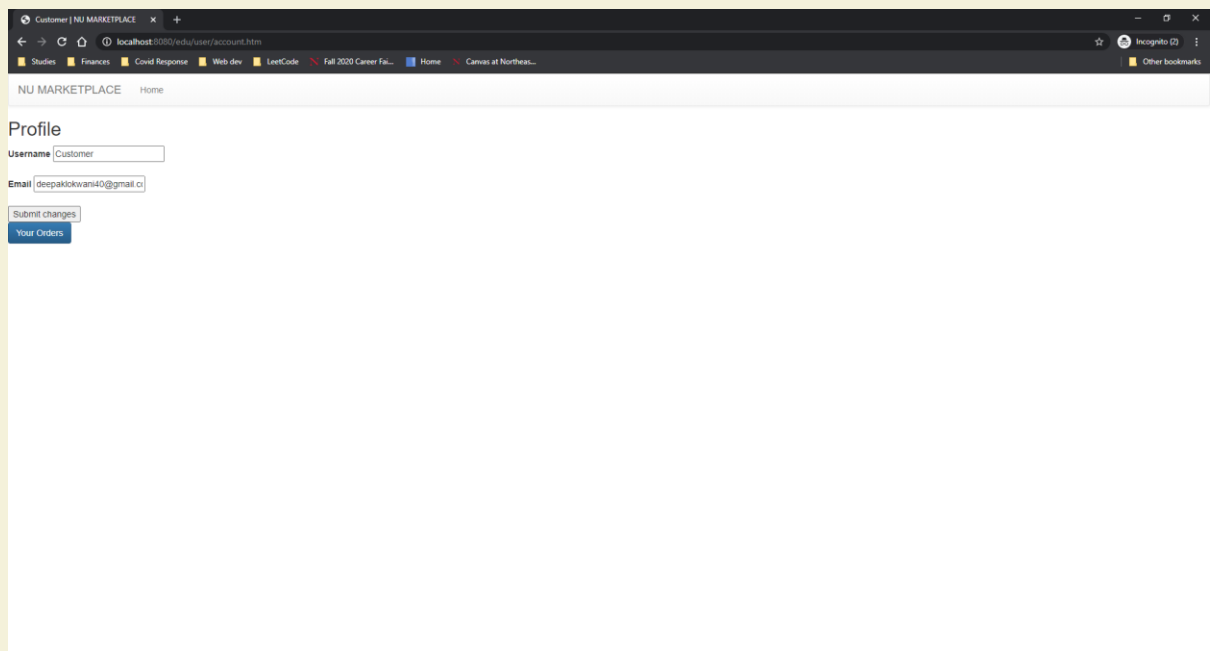
Annotation based Database mapping
Hibernate Criteria
Filters and Validations
BCrypt Authentication for passwords
Logging

Views of the Web app:**1. Landing page****2. Login Page**

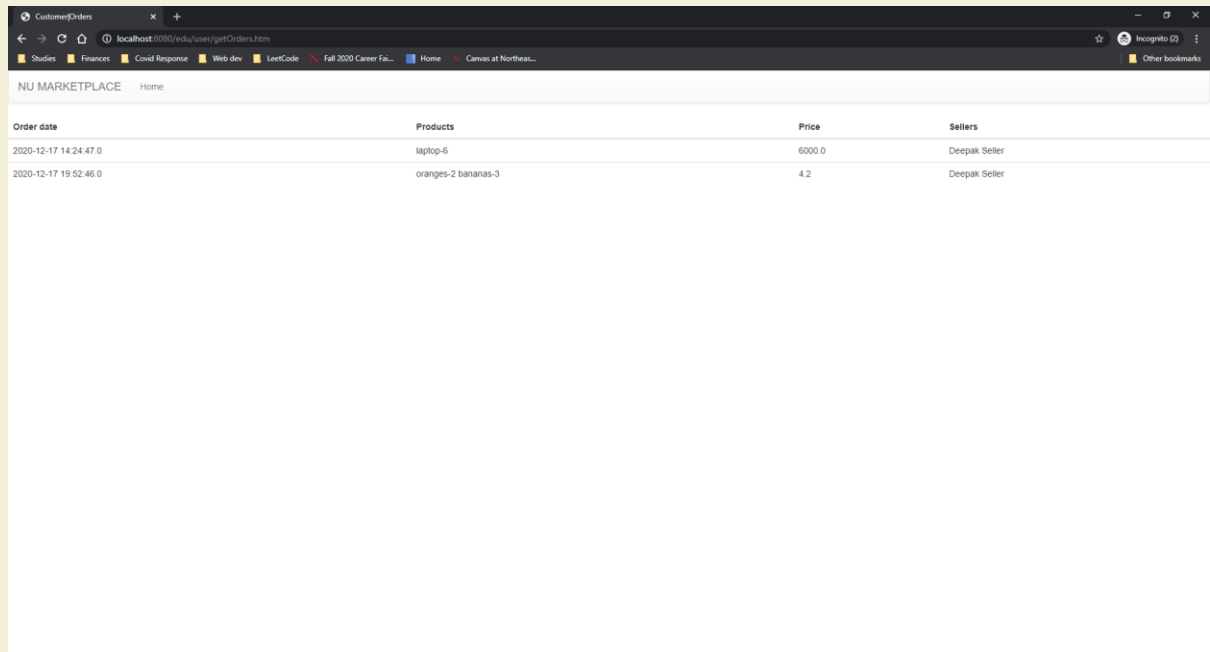
3. Homepage



4. User Account Page

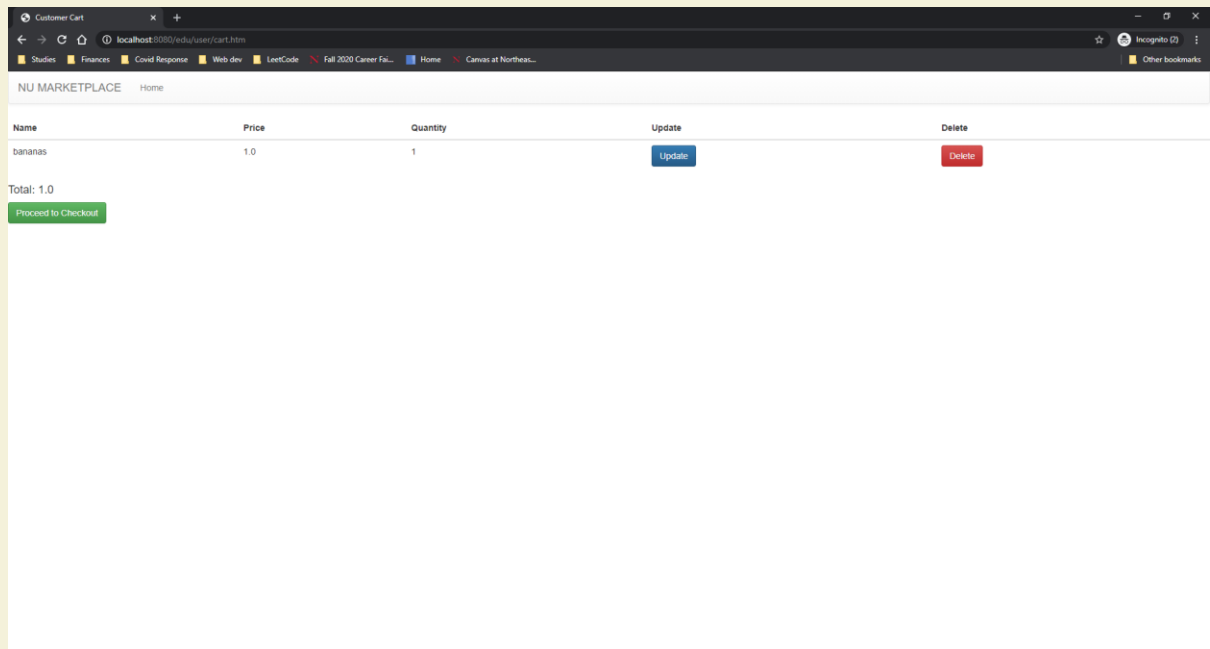


5. Customer Orders



Order date	Products	Price	Sellers
2020-12-17 14:24:47.0	laptop-6	6000.0	Deepak Seller
2020-12-17 19:52:46.0	oranges-2 bananas-3	4.2	Deepak Seller

6. Customer Cart

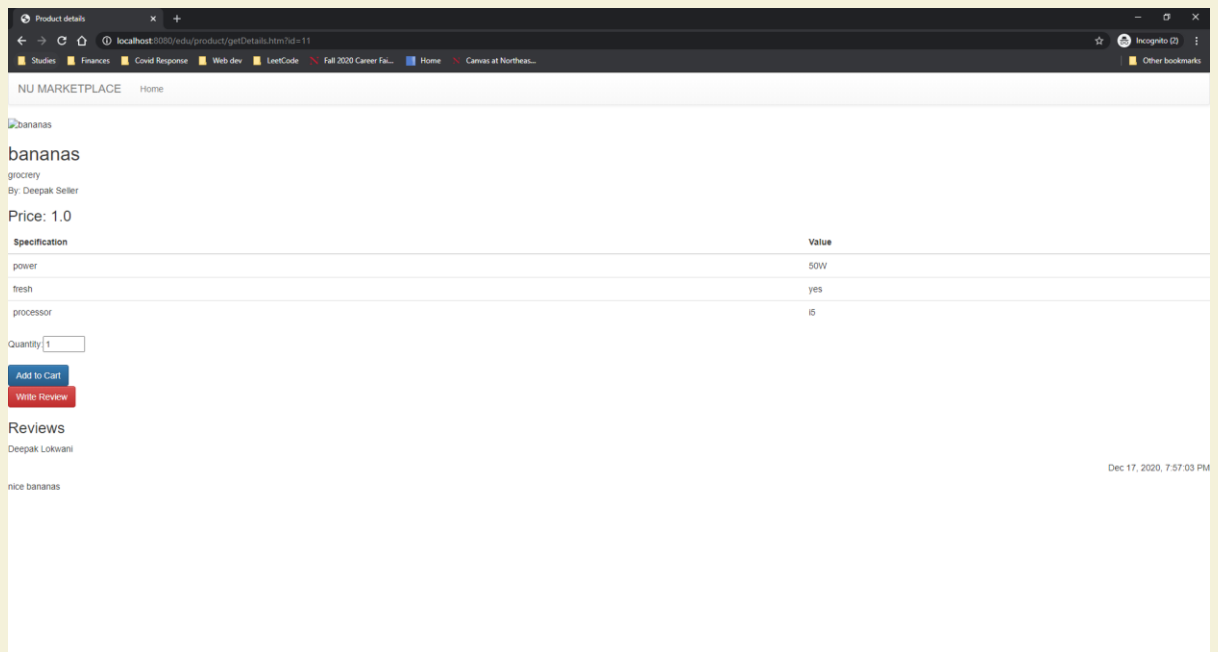


Name	Price	Quantity	Update	Delete
bananas	1.0	1	Update	Delete

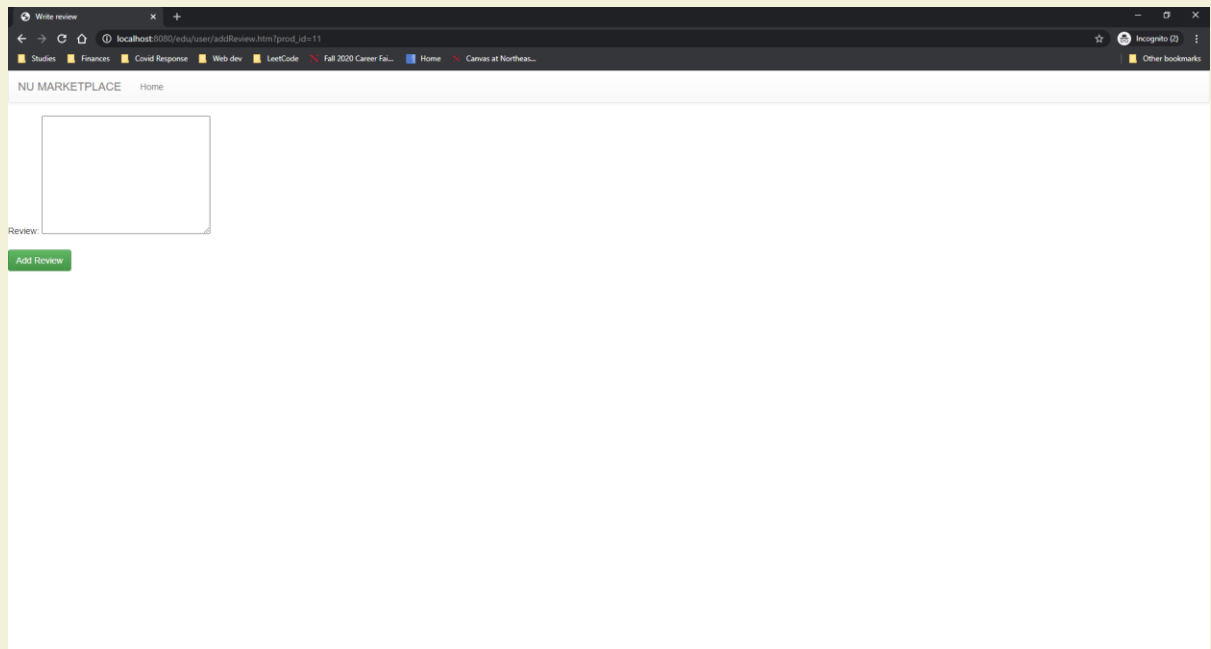
Total: 1.0

[Proceed to Checkout](#)

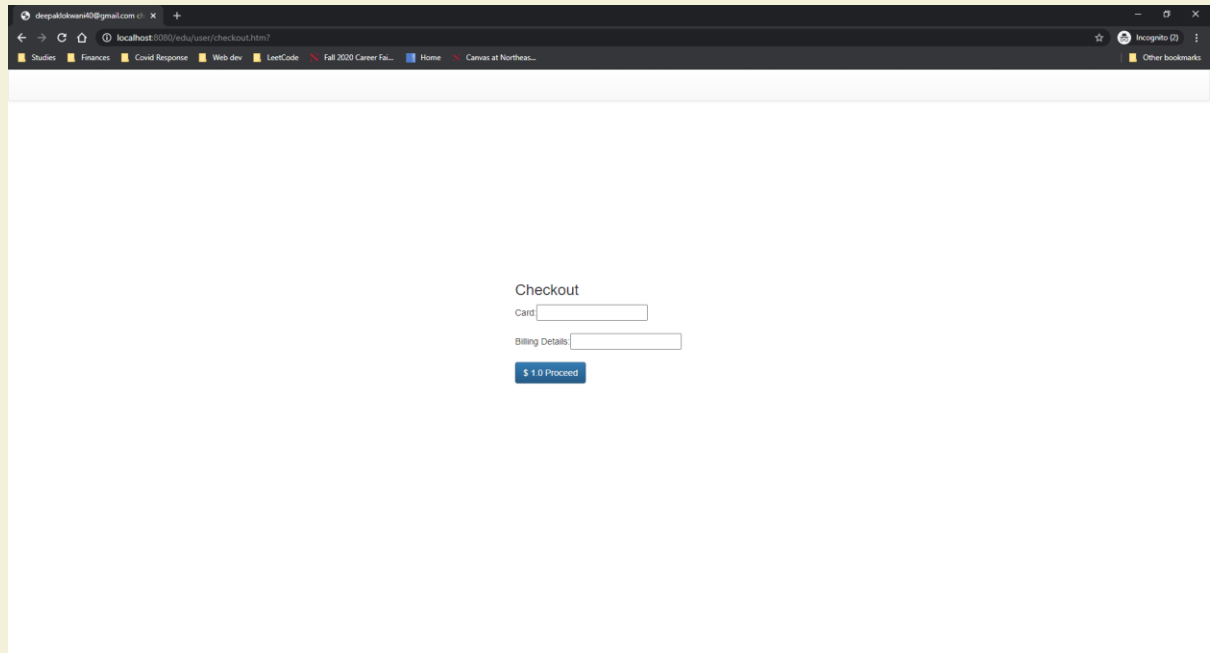
7. Customer cart update view



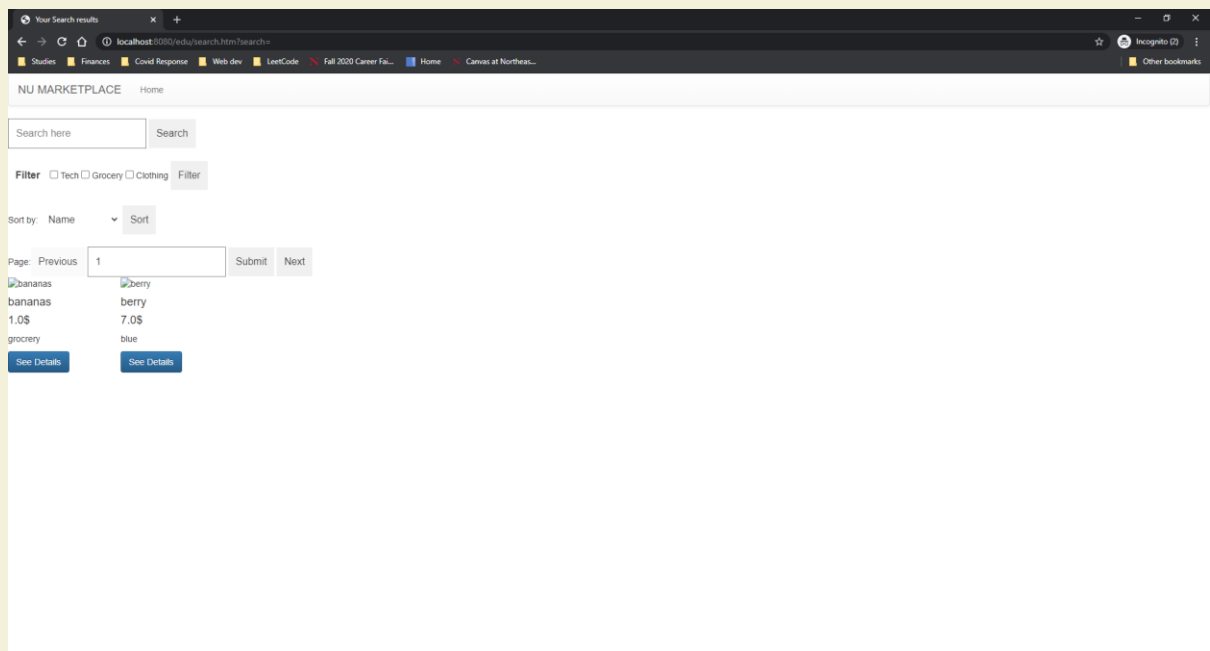
8. Write Review – Customer page



9. Checkout Page



10. Search Item page



11. Seller Account Page

NU Marketplace Home

Profile

Username

Email

Add new Products

Products

Id	Name	Price	Quantity	Update	Delete
5	laptop	1000	0	<input type="button" value="update"/>	<input type="button" value="delete"/>
7	oranges	0.6	6	<input type="button" value="update"/>	<input type="button" value="delete"/>
11	bananas	1	10	<input type="button" value="update"/>	<input type="button" value="delete"/>

12. Seller Orders Page

NU MARKETPLACE Home

Order date	Products	Price	User
2020-12-17 14:24:47.0	laptop-6	6000.0	Customer
2020-12-17 19:52:46.0	bananas-3 oranges-2	4.2	Customer
2020-12-18 17:07:46.0	bananas-1	1.0	Customer

13. Add Product – Seller View

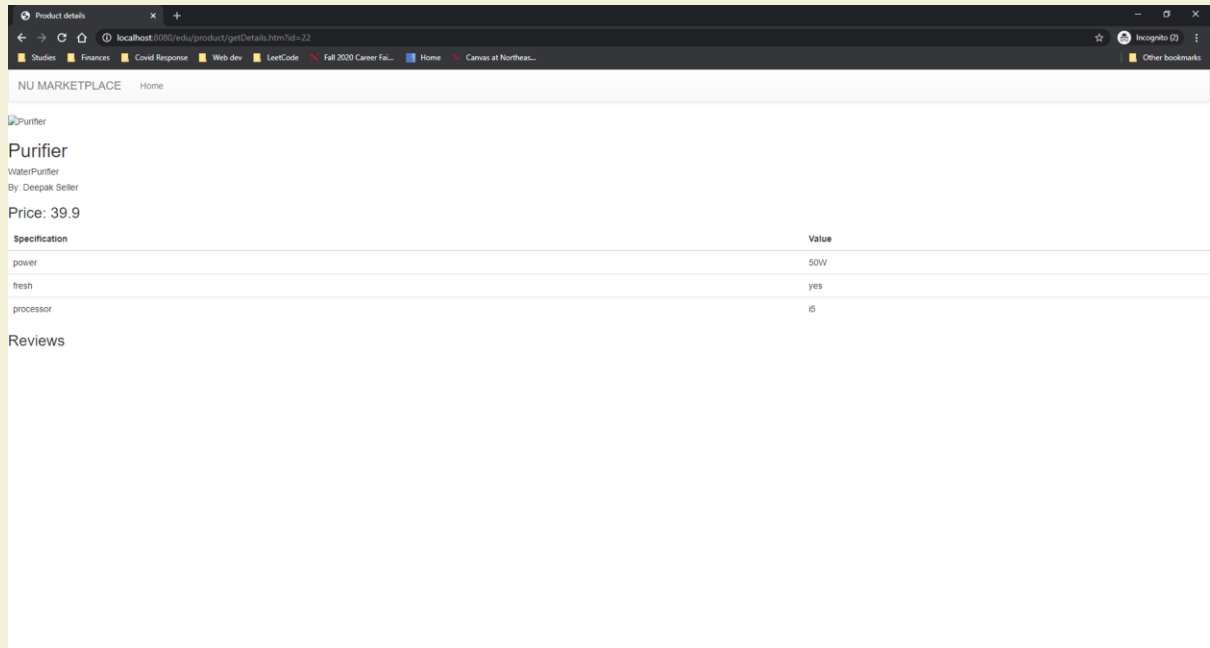
The screenshot shows a web browser window with the URL `localhost:8080/edu/seller/addTechProduct.htm`. The page title is "Add Tech product". The form includes the following fields and elements:

- Name:** A text input field.
- Description:** A text input field.
- Photo:** A file upload button labeled "Choose File | No file chosen".
- Price:** A text input field.
- Quantity:** A text input field.
- Specification Table:** A table with two columns, "Specification" and "Value". It contains three empty rows for data entry.
- Add product:** A button at the bottom of the form.

14. Search bar for Seller

The screenshot shows a web browser window with the URL `localhost:8080/edu/search.htm?search=purifier&filter=tech&sort=name`. The page title is "Your Search results". The search results page includes the following elements:

- Search Bar:** A text input field containing "purifier" and a "Search" button.
- Filter:** A section with checkboxes for "Tech", "Grocery", and "Clothing". The "Tech" checkbox is selected. A "Filter" button is next to it.
- Sort by:** A dropdown menu showing "Name" and a "Sort" button.
- Page:** A section with "Previous", "1", "Submit", and "Next" buttons.
- Search Results:** A list of results showing a "Purifier" with a price of "39.9\$". Below the price, it says "WaterPurifier". A "See Details" button is at the bottom of the result.

15. Product Details for Seller

As seen, it is clear that seller account does not have the buying privileges.

```
<html>
  <head>
    <title> PARTING NOTE </title>
  </head>
  <body>
    <%= "THANK YOU" %>
  </body>
</html>
```

Controller Source Code**1. HomeController**

```
package com.finalproject.controllers;

import java.util.Locale;
import javax.servlet.http.HttpServletRequest;
import org.slf4j.Logger;
import org.springframework.stereotype.Controller;
import org.slf4j.LoggerFactory;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;

/**
 *
 * @author Deepak_Lokwani
 *
 *      NUID: 001316769
 *
 *      Project name: Finalproject Package name: com.finalproject.controllers
 */

@Controller
public class HomeController {

    /**
     * Handles requests for the application home page.
     */
    private static final Logger Logger =
        LoggerFactory.getLogger(HomeController.class);

    /**
     * It selects home view to render by reverting its name.
     */
    @RequestMapping(value = { "/", "/index.htm" }, method = RequestMethod.GET)
    public String home(Locale locale, Model model) {
        return "home";
    }

    @RequestMapping(value = "/error.htm", method = RequestMethod.GET)
    public String error(HttpServletRequest request) {
        request.setAttribute("errorMessage", "something went wrong!!");
        return "error";
    }
}
```

2. ProdController:

```
package com.finalproject.controllers;

import java.io.File;
import java.util.ArrayList;
import java.sql.Timestamp;

import java.io.IOException;
import java.util.Arrays;

import java.util.HashMap;
import java.util.List;
import java.util.Calendar;
import java.util.Map;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.Cookie;
import javax.servlet.http.HttpServletResponse;

import org.springframework.stereotype.Controller;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.ui.ModelMap;

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

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

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

import org.springframework.web.multipart.commons.CommonsMultipartFile;
import org.springframework.web.client.HttpServerErrorException;
import com.finalproject.DAO.ProductDAO;

import com.finalproject.DAO.SellerDAO;
import com.finalproject.DAO.ReviewDAO;
import com.finalproject.DAO.UserDAO;

import com.finalproject.POJO.Person;
import com.finalproject.POJO.GroceryProduct;

import com.finalproject.POJO.Review;
import com.finalproject.POJO.Product;

import com.finalproject.POJO.TechProduct;
import com.finalproject.POJO.Seller;

import com.finalproject.utils.GroceryProductValidator;
import com.finalproject.POJO.User;

import com.finalproject.utils.TechproductValidator;
import com.finalproject.utils.ReviewValidator;
import com.google.gson.Gson;
```

```
import com.google.gson.GsonBuilder;

/**
 *
 * @author Deepak_Lokwani
 *
 *      NUID: 001316769
 *
 *      Project name: Finalproject
 *      Package name: com.finalproject.controllers
 */

@Controller
public class ProdController {

    @Autowired
    ReviewValidator reviewValidator;
    @Autowired
    TechproductValidator techValidator;
    @Autowired
    GroceryProductValidator groceryValidator;

    /**
     * method to add a technical product
     *
     * @param technicalProduct
     * @param modelMap
     * @param req
     * @return
     */
    @RequestMapping(value = "/seller/addTechProduct.htm", method =
RequestMethod.GET)
    public String getAddTechnicalProduct(TechProduct technicalProduct, ModelMap
modelMap, HttpServletRequest req) {
        if (req.getSession().getAttribute("user") == null) {
            req.setAttribute("errorMessage", "Please login before
continuing");
            return "error";
        }
        modelMap.addAttribute(technicalProduct);
        return "addTechProduct";
    }

    /**
     * method to handle the method to add a technical product
     *
     * @param technicalProduct
     * @param productDao
     * @param sellerDao
     * @param bindingResult
     * @param sessionStatus
     * @param req
     * @param res
     * @return
     */
    @RequestMapping(value = "/seller/addTechProduct.htm", method =
RequestMethod.POST)
```

```
public String addTechnicalProduct(@ModelAttribute("techProduct")
TechProduct technicalProduct,
ProductDAO productDao, SellerDAO sellerDao, BindingResult
bindingResult, SessionStatus sessionStatus,
HttpServletRequest req, HttpServletResponse res) {
    CommonsMultipartFile iconPic = technicalProduct.getPhoto();
    if (iconPic.getSize() != 0) {
        String fileName = "img" + technicalProduct.getProduct_id() +
Calendar.getInstance().getTimeInMillis()
+ iconPic.getContentType();
        File photoFile = new File("C:/Users/deepa/Desktop/images",
fileName);

        technicalProduct.setPhotoFile(fileName);
        try {
            iconPic.transferTo(photoFile);
        } catch (IllegalStateException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
            return "error";
        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
            return "error";
        }
    }
    String[] specs = req.getParameterValues("spec.key");
    String[] values = req.getParameterValues("spec.value");

    Map<String, String> specificationMap = new HashMap<String,
String>();
    for (int i = 0; i < specs.length; i++) {
        specificationMap.put(specs[i], values[i]);
    }
    technicalProduct.setSpec(specificationMap);
    technicalProduct.setType("tech");

    Seller seller = sellerDao.getSellerByEmail((String)
req.getSession().getAttribute("user"));
    if (seller == null) {
        req.setAttribute("errorMessage", "there is no seller with that
id");

        return "error";
    }
    technicalProduct.setSeller(seller);
    technicalProduct.setAddedAt(new
Timestamp(Calendar.getInstance().getTimeInMillis()));

    if (productDao.addProduct(technicalProduct))
        return "redirect:
http://localhost:8080/edu/seller/account.htm";
    else {
        req.setAttribute("errorMessage", "Error: Try adding a
different product");
        return "error";
    }
}

/**
 * Method to add a grocery product
```

```
*
* @param req
* @param model
* @param grocProd
* @return
*/
@RequestMapping(value = "/seller/addGroceryProduct.htm", method =
RequestMethod.GET)
public String getAddGrocProd(HttpServletRequest req, ModelMap model,
GroceryProduct grocProd) {
    if (req.getSession().getAttribute("user") == null) {
        req.setAttribute("errorMessage", "Please login, to continue
having any access to the page");
        return "error";
    }
    model.addAttribute(grocProd);
    return "addGroceryProduct";
}

/**
 * Method to add a grocery product by a seller
 *
 * @param request
 * @param grocProd
 * @param results
 * @param productdao
 * @param sellerdao
 * @return
 */
@RequestMapping(value = "/seller/addGroceryProduct.htm", method =
RequestMethod.POST)
public String addGrocProd(HttpServletRequest request,
@ModelAttribute("groceryProduct") GroceryProduct grocProd,
BindingResult results, ProductDAO productdao, SellerDAO
sellerdao) {
    if (request.getSession().getAttribute("user") == null) {
        request.setAttribute("errorMessage", "Please login, to
continue having any access to the page");
        return "error";
    }
    /*
     * handling a picture file
     */
    CommonsMultipartFile iconPic = grocProd.getPhoto();
    if (iconPic.getSize() != 0) {
        String fileName = "img" + grocProd.getProduct_id() +
Calendar.getInstance().getTimeInMillis()
+ iconPic.getContentType();
        File file = new File("C:/Users/deepa/Desktop/images",
fileName);
        grocProd.setPhotoFile(fileName);
        try {
            iconPic.transferTo(file);
        } catch (IllegalStateException e) {
            e.printStackTrace();
            return "error";
        } catch (IOException e) {
            e.printStackTrace();
            return "error";
        }
    }
}
```



```

    }
    }
    grocProd.setType("grocery");
    Seller seller = sellerdao.getSellerByEmail((String)
request.getSession().getAttribute("user"));
    if (seller == null) {
        request.setAttribute("errorMessage", "there is no seller with
that id");
        return "error";
    }
    grocProd.setSeller(seller);
    grocProd.setAddedAt(new
Timestamp(Calendar.getInstance().getTimeInMillis()));
    if (productdao.addProduct(grocProd))
        return "redirect:
http://localhost:8080/edu/seller/account.htm";
    else {
        request.setAttribute("errormesssge", "Error adding product");
        return "error";
    }
}

/**
 * method to handle a search product
 *
 * @param req
 * @param prodDao
 * @return
 */
@RequestMapping(value = "/search.htm", method = RequestMethod.GET)
public String searchProduct(HttpServletRequest req, ProductDAO prodDao) {

    String key = req.getParameter("search");
    String[] filterKeys = req.getParameterValues("filter");
    String sort = req.getParameter("sort");
    sort = sort == null ? "name" : sort;
    int page = req.getParameter("page") == null ? 1 :
Integer.parseInt(req.getParameter("page"));
    List<Product> products = prodDao.getProducts(key, filterKeys, sort,
page);

    req.setAttribute("products", products);
    req.setAttribute("search", key);
    req.setAttribute("page", page);
    req.setAttribute("sort", sort);
    return "searchResults";
}

/**
 * method to handle a delete product
 *
 * @param req
 * @param prodDao
 * @param sellerdao
 * @return
 */
@RequestMapping(value = "/seller/deleteProduct.htm", method =
RequestMethod.GET)
public String deleteProd(HttpServletRequest req, ProductDAO prodDao,
SellerDAO sellerdao) {

```

```
        if (req.getSession().getAttribute("user") == null) {
            req.setAttribute("errorMessage", "Please login, to continue
having any access to the page");
            return "error";
        }
        if (req.getParameter("prod_id") == null) {
            req.setAttribute("errorMessage", "Invalid id for product");
            return "error";
        }

        Product product =
prodDao.getProductById(Integer.parseInt(req.getParameter("prod_id")));
        if (product == null) {
            req.setAttribute("errorMessage", "this product does not seem
to exist");
            return "error";
        }
        if
(!product.getSeller().getEmail().equals(req.getSession().getAttribute("user"))) {
            req.setAttribute("errorMessage", "Please verify your product
id");
            return "error";
        }
        if
(!prodDao.deleteProduct(Integer.parseInt(req.getParameter("prod_id")))) {
            req.setAttribute("errorMessage", "error deleting this product,
please tryagain later");
            return "error";
        }
        return "redirect: http://localhost:8080/edu/seller/account.htm";
    }

    /**
     * method to update a product
     *
     * @param request
     * @param model
     * @param product
     * @param productdao
     * @return
     */
    @RequestMapping(value = "/seller/updateProduct.htm", method =
RequestMethod.GET)
    public String getUpdateProd(HttpServletRequest request, ModelMap model,
Product product, ProductDAO productdao) {
        if (request.getSession().getAttribute("user") == null) {
            request.setAttribute("errorMessage", "Please login, to
continue having any access to the page");
            return "error";
        }
        if (request.getParameter("prod_id") == null) {
            request.setAttribute("errorMessage", "Invalid product id");
            return "error";
        }
        Product productToUpdate =
productdao.getProductById(Integer.parseInt(request.getParameter("prod_id")));
        if (productToUpdate == null) {
            request.setAttribute("errorMessage", "Invalid product id");
            return "error";
        }
    }
```

```
    }

    model.addAttribute("product", productToUpdate);
    return "productUpdate";
}

/**
 * method to update a product by a seller
 *
 * @param request
 * @param product
 * @param results
 * @param productdao
 * @return
 */
@RequestMapping(value = "/seller/updateProduct.htm", method =
RequestMethod.POST)
    public String updateProduct(HttpServletRequest request,
@ModelAttribute("product") Product product,
        BindingResult results, ProductDAO productdao) {

        if (!productdao.updateProduct(product)) {
            request.setAttribute("errorMessage", "Error updating
product");
            return "error";
        }
        return "redirect: /edu/seller/account.htm";
    }

/**
 * method to get a product
 *
 * @param request
 * @param prodDao
 * @return
 */
@RequestMapping(value = "/product/getDetails.htm", method =
RequestMethod.GET)
    public String getProductDetails(HttpServletRequest request, ProductDAO
prodDao) {
        if (request.getParameter("id") == null) {
            request.setAttribute("errorMessage", "Invalid product id");
            return "error";
        }
        Product product =
prodDao.getProductById(Integer.parseInt(request.getParameter("id")));
        if (product == null) {
            request.setAttribute("errorMessage", "Product does not
exist");
            return "error";
        }
        request.setAttribute("product", product);
        return "productDetails";
    }

/**
 * method to add a product to the cart
 *
 * @param request
```

```
* @param productdao
* @param userdao
* @return
*/
@RequestMapping(value = "/user/addToCart.htm", method = RequestMethod.POST)
public String addProdsToCart(HttpServletRequest request, ProductDAO
productdao, UserDAO userdao) {
    Product product =
productdao.getProductById(Integer.parseInt(request.getParameter("id")));
    int quantity = Integer.parseInt(request.getParameter("quantity"));
    if (product == null) {
        request.setAttribute("errorMessage", "This product does not
seem to exist");
        return "error";
    }
    User user = userdao.getUserByEmail((String)
request.getSession().getAttribute("user"));
    if (user == null) {
        request.setAttribute("errorMessage", "Please sign up here
before further continuing");
        return "error";
    }
    if (product.getQuantity() < quantity) {
        request.setAttribute("errorMessage", "Invalid quantity");
        return "error";
    }
    if (!userdao.deleteProductFromCart(user, product)) {
        request.setAttribute("errorMessage", "Item is not in your
cart");
        return "error";
    }
    if (!userdao.addProductToCart(user, product, quantity)) {
        request.setAttribute("errorMessage", "Error Adding product to
cart");
        return "error";
    }
    return "redirect: /edu/user/cart.htm";
}

/**
 * method to delete a product from the cart
 *
 * @param request
 * @param productdao
 * @param userdao
 * @return
 */
@RequestMapping(value = "/user/deleteFromCart.htm", method =
RequestMethod.GET)
public String delProdsFromCart(HttpServletRequest request, ProductDAO
productdao, UserDAO userdao) {
    if (request.getSession().getAttribute("user") == null) {
        request.setAttribute("errorMessage", "Please login before
continuing");
        return "error";
    }
}
```

```
        Product product =
productdao.getProductById(Integer.parseInt(request.getParameter("id")));
        if (product == null) {
            request.setAttribute("errorMessage", "PThis product does not
seem to exist");
            return "error";
        }
        User user = userDao.getUserByEmail((String)
request.getSession().getAttribute("user"));
        if (user == null) {
            request.setAttribute("errorMessage", "Please sign up here
before further continuing");
            return "error";
        }
        if (!userdao.deleteProductFromCart(user, product)) {
            request.setAttribute("errorMessage", "Item is not in your
cart");
            return "error";
        }
        return "redirect: /edu/user/cart.htm";
    }

/**
 * method to add a review to the product
 *
 * @param request
 * @param model
 * @param review
 * @param productdao
 * @return
 */
@RequestMapping(value = "/user/addReview.htm", method = RequestMethod.GET)
public String addReviews(HttpServletRequest request, ModelMap model, Review
review, ProductDAO productdao) {
    if (request.getSession().getAttribute("user") == null) {
        request.setAttribute("errorMessage", "Please login before
continuing");
        return "error";
    }
    if (request.getParameter("prod_id") == null) {
        request.setAttribute("errorMessage", "Invalid product id");
        return "error";
    }
    Product product =
productdao.getProductById(Integer.parseInt(request.getParameter("prod_id")));
    if (product == null) {
        request.setAttribute("errorMessage", "Product does not
exist");
        return "error";
    }
    model.addAttribute(review);
    return "review";
}

/**
 * method to process a review
 *
 * @param req
```

```

    * @param userDao
    * @param productdao
    * @param review
    * @param results
    * @param reviewdao
    * @return
    */
    @RequestMapping(value = "/user/addReview.htm", method = RequestMethod.POST)
    public String processReviews(HttpServletRequest req, UserDao userDao,
        ProductDAO productdao,
        @ModelAttribute("review") Review review, BindingResult
        results, ReviewDAO reviewdao) {
        reviewValidator.validate(review, results);
        if (results.hasErrors()) {
            System.out.print(results.getErrorCount());
            System.out.println(results.getAllErrors());

            System.out.println("Hello!!");
            return "review";
        }

        if (req.getSession().getAttribute("user") == null) {
            req.setAttribute("errorMessage", "Login before continuing");
            return "error";
        }
        Product product =
        productdao.getProductById(Integer.parseInt(req.getParameter("prod_id")));
        User user = userDao.getUserByEmail((String)
        req.getSession().getAttribute("user"));
        if (user == null) {
            req.setAttribute("errorMessage", "Signup before continuing");
            return "error";
        }
        review.setProduct(product);
        review.setSeller(product.getSeller());
        review.setUser(user);
        review.setUsername(user.getName());
        review.setAddedAt(new
        Timestamp(Calendar.getInstance().getTimeInMillis()));
        if (!reviewdao.addReview(review)) {
            req.setAttribute("errorMessage", "Error adding review");
            return "error";
        }
        return "redirect: /edu/index.htm";
    }

    /**
     * method to get a review
     */
    * @param request
    * @param reviewDao
    * @return
    */
    @RequestMapping(value = "/product/getReviews.htm", method =
    RequestMethod.GET)
    @ResponseBody
    public String getReviews(HttpServletRequest request, ReviewDAO reviewDao) {
        List<Review> reviewList =
        reviewDao.getReviewsByProduct(Integer.parseInt(request.getParameter("id")),

```

```
        (Integer.parseInt(request.getParameter("page"))));
        Gson gson = new
GsonBuilder().excludeFieldsWithoutExposeAnnotation().create();
        return gson.toJson(reviewList);
    }
}
```

3. SellerController

```
package com.finalproject.controllers;

import java.io.IOException;
import java.sql.Timestamp;
import java.util.Calendar;
import java.util.List;
import java.io.File;

import javax.servlet.http.Cookie;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.RequestDispatcher;

import org.springframework.web.bind.support.SessionStatus;
import org.springframework.web.multipart.commons.CommonsMultipartFile;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.ResponseBody;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.stereotype.Controller;
import org.springframework.ui.ModelMap;
import org.springframework.validation.BindingResult;

import com.finalproject.utils.UserValidator;
import com.finalproject.utils.SellerLoginValidator;
```

```
import com.finalproject.utils.SellerValidator;
import com.finalproject.utils.Hash;
import com.finalproject.utils.LoginValidator;
import com.finalproject.POJO.Seller;
import com.finalproject.POJO.User;
import com.finalproject.DAO.SellerDAO;
import com.finalproject.POJO.Person;
import com.finalproject.POJO.Product;
import com.finalproject.DAO.UserDAO;
import com.google.gson.GsonBuilder;
import com.google.gson.Gson;

/**
 *
 * @author Deepak_Lokwani
 *
 *      NUID: 001316769
 *
 *      Project name: Finalproject
 *      Package name: com.finalproject.controllers
 */

@Controller
public class SellerController {

    public SellerController() {
    }

    @Autowired
    SellerValidator sellerValidator;

    @Autowired
    SellerLoginValidator sellerLoginValidator;

    /**
     *
     * @param model
     * @param seller
     * @param request
     * @return
     *
     * Method to get Login
     */
    @RequestMapping(value = { "/seller/login.htm" }, method =
RequestMethod.GET)
    private String getLogin(ModelMap model, Seller seller, HttpServletRequest
request) {
        if (request.getSession().getAttribute("user") != null) {
            return "redirect: /edu/index.htm";
        }
        request.setAttribute("seller", "seller");
        model.addAttribute(seller);
        return "login";
    }

    /**
     *

```



```

    * @param seller
    * @param result
    * @param status
    * @param sellerdao
    * @param request
    * @param response
    * @return
    *
    * Post Method to handle login
    */
    @RequestMapping(value = { "/seller/login.htm" }, method =
RequestMethod.POST)
    private String handleLogin(@ModelAttribute("seller") Seller seller,
BindingResult result, SessionStatus status,
SellerDAO sellerdao, HttpServletRequest request,
HttpServletResponse response) {
        sellerLoginValidator.validate(seller, result);
        if (result.hasErrors())
            return "login";
        Person person = sellerdao.getSellerByEmail(seller.getEmail());
        if (person == null) {
            request.setAttribute("errorMessage",
                "User with email " + seller.getEmail() + " is not
found. Please signup!!");
            return "error";
        }
        if (!Hash.checkPasssword(seller.getPassword(),
person.getPassword())) {
            request.setAttribute("errorMessage", "Invalid username or
password");
            return "error";
        }
        createCookie(30 * 24 * 60 * 60, "seller", person.getEmail(), "/",
response);
        createCookie(30 * 24 * 60 * 60, "name", person.getName().split("
")[0], "/", response);
        return "redirect:" + "http://localhost:8080/edu/index.htm";
    }

    /**
     *
     * @param model
     * @param seller
     * @param request
     * @return
     *
     * Method to get signup attrinutes
     */
    @RequestMapping(value = { "/seller/signup.htm" }, method =
RequestMethod.GET)
    private String getSignup(ModelMap model, Seller seller, HttpServletRequest
request) {
        if (request.getSession().getAttribute("user") != null) {
            return "redirect: /edu/index.htm";
        }
        model.addAttribute(seller);
        return "sellerSignup";
    }

```

```
/**
 *
 * @param seller
 * @param result
 * @param status
 * @param sellerdao
 * @param request
 * @param response
 * @return
 *
 * Method to handle signups
 */
@RequestMapping(value = { "/seller/signup.htm" }, method =
RequestMethod.POST)
private String handleSignup(@ModelAttribute("seller") Seller seller,
BindingResult result, SessionStatus status,
SellerDAO sellerdao, HttpServletRequest request,
HttpServletResponse response) {
    sellerValidator.validate(seller, result);
    if (result.hasErrors())
        return "sellerSignup";

    /*
     * Handling photo in signup
     */
    CommonsMultipartFile photo = seller.getPhoto();
    if (photo.getSize() != 0) {
        String fileName = "img" + seller.getId() +
Calendar.getInstance().getTimeInMillis()
+ photo.getContentType();
        File file = new File("C:/Users/deepa/Desktop/images",
fileName);
        seller.setPhotoFile(fileName);
        try {
            photo.transferTo(file);
        } catch (IllegalStateException e) {
            e.printStackTrace();
            return "error";
        } catch (IOException e) {
            e.printStackTrace();
            return "error";
        }
    }
    seller.setCreatedAt(new
Timestamp(Calendar.getInstance().getTimeInMillis()));
    seller.setRole("seller");
    seller.setPassword(Hash.hashPassword(seller.getPassword()));

    if (!sellerdao.addSeller(seller)) {
        request.setAttribute("errormessage", "Error adding User please
try again!!");
        return "error";
    }
    ;

    /*
     * creating a cookie
     */
}
```

```
        createCookie(30 * 24 * 60 * 60, "seller", seller.getEmail(), "/",
response);
        createCookie(30 * 24 * 60 * 60, "name", seller.getName().split("
")[0], "/", response);

        // clean up the session
        status.setComplete();
        return "redirect:" + "http://localhost:8080/edu/index.htm";
    }

    /**
     *
     * @param request
     * @return
     *
     * method to delete user
     */
    @RequestMapping(value = "/seller/delete.htm", method = RequestMethod.GET)
    public String deleteUser(HttpServletRequest request) {
        if (request.getSession().getAttribute("user") == null) {
            request.setAttribute("errorMessage", "Login before
continuing");
            return "error";
        }
        System.out.println((String)
request.getSession().getAttribute("user"));
        request.setAttribute("user", "seller");
        return "removeUser";
    }

    /**
     *
     * @param request
     * @param response
     * @param sellerdao
     * @return
     *
     * method to handle delete user
     */
    @RequestMapping(value = "/seller/delete.htm", method = RequestMethod.POST)
    public String handleDelete(HttpServletRequest request, HttpServletResponse
response, SellerDAO sellerdao) {
        Seller person =
sellerdao.getSellerByEmail(request.getParameter("email"));
        if (person == null) {
            request.setAttribute("errorMessage",
"User with email " +
request.getParameter("email") + " is not found. Please verify!!");
            return "error";
        }
        if (!sellerdao.deleteSeller(person)) {
            request.setAttribute("errorMessage", "Error deleting
seller!!");
            return "error";
        }
        deleteCookie(request, response);
        return "redirect:" + "http://localhost:8080/edu/index.htm";
    }
}
```

```
/**
 * method to get seller account details
 * @param request
 * @param sellerdao
 * @return
 */
@RequestMapping(value = "/seller/account.htm", method = RequestMethod.GET)
public String account(HttpServletRequest request, SellerDAO sellerdao) {
    if (request.getSession().getAttribute("user") == null) {
        request.setAttribute("errorMessage", "Login before
continuing");
        return "error";
    }
    Person person = sellerdao.getSellerByEmail((String)
request.getSession().getAttribute("user"));
    request.setAttribute("user", person);
    return "sellerProfile";
}

/**
 * method to update the seller profile
 * @param userDao
 * @param req
 * @param res
 * @param sellerDao
 * @return
 */
@RequestMapping(value = "/seller/updateProfile.htm", method =
RequestMethod.POST)
public String handleUpdateProfile(UserDAO userDao, HttpServletRequest req,
HttpServletResponse res,
SellerDAO sellerDao) {
    Person person = sellerDao.getSellerByEmail((String)
req.getSession().getAttribute("user"));
    if (userDao.updatePerson(person, req.getParameter("name"),
req.getParameter("email"))) {
        createCookie(30 * 24 * 60 * 60, "seller", person.getEmail(),
"/", res);
        createCookie(30 * 24 * 60 * 60, "name",
person.getName().split(" ")[0], "/", res);
        return "redirect:" + "http://localhost:8080/edu/index.htm";
    } else {
        req.setAttribute("errorMessage", "Error updating your profile.
Please try with different email");
        return "error";
    }
}

/**
 * Method to get the products of a seller
 * @param req
 * @param sellerDao
 * @return
 */
@RequestMapping(value = "/seller/product.htm", method = RequestMethod.GET)
@ResponseBody
public String getProducts(HttpServletRequest req, SellerDAO sellerDao) {
    if (req.getSession().getAttribute("user") == null) {
```

```
        req.setAttribute("errorMessage", "Please login before
continuing");
        return "Login before continuing";
    }
    Seller seller = sellerDao.getSellerByEmail((String)
req.getSession().getAttribute("user"));

    Gson gson = new
GsonBuilder().excludeFieldsWithoutExposeAnnotation().create();
    return gson.toJson(seller.getProducts());
}

/**
 * method to get orders of a seller
 * @param request
 * @param sellerdao
 * @return
 */
@RequestMapping(value = "/seller/getOrders.htm", method =
RequestMethod.GET)
public String getOrders(HttpServletRequest request, SellerDAO sellerdao) {
    if (request.getSession().getAttribute("user") == null) {
        request.setAttribute("errorMessage", "Login before
continuing!!");
        return "error";
    }
    Seller seller = sellerdao.getSellerByEmail((String)
request.getSession().getAttribute("user"));
    if (seller == null) {
        request.setAttribute("errorMessage", "User does not exist");
        return "error";
    }
    request.setAttribute("orders", seller.getOrders());
    return "showOrders";
}

/**
 * method to create a cookie for a session
 * @param time
 * @param name
 * @param msg
 * @param path
 * @param response
 */
public void createCookie(int time, String name, String msg, String path,
HttpServletRequest response) {
    Cookie c = new Cookie(name, msg);
    c.setMaxAge(time);
    c.setPath(path);
    response.addCookie(c);
}

/**
 * method to delete a cookie after the logout
 * @param request
 * @param response
 */
private void deleteCookie(HttpServletRequest request, HttpServletResponse
response) {
```

```
Cookie[] cookies = request.getCookies();
for (Cookie cookie : cookies) {
    cookie.setPath("/");
    cookie.setValue("");
    cookie.setMaxAge(0);
    response.addCookie(cookie);
}

}

}
```

4. UserController:

```
package com.finalproject.controllers;

import org.springframework.web.multipart.commons.CommonsMultipartFile;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.ui.ModelMap;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.support.SessionStatus;
import org.springframework.stereotype.Controller;

import com.finalproject.utils.LoginValidator;
import com.finalproject.utils.OrderValidator;
import com.finalproject.utils.Mail;
import com.finalproject.utils.RandomTokenGenerator;
import com.finalproject.utils.UserValidator;
import com.finalproject.POJO.User;
import com.finalproject.POJO.Seller;
import com.finalproject.POJO.Product;
import com.finalproject.POJO.Person;
import com.finalproject.DAO.OrderDAO;
import com.finalproject.POJO.Order;
import com.finalproject.DAO.UserDAO;
import com.finalproject.DAO.SellerDAO;
import com.finalproject.DAO.ProductDAO;
import com.finalproject.utils.Hash;

import java.util.HashSet;

import java.util.Map;
import java.util.Set;
import java.io.File;
import java.io.IOException;
import java.sql.Timestamp;

import java.util.Calendar;

import javax.servlet.http.Cookie;
import javax.servlet.http.HttpServletRequest;
```

```
import javax.servlet.http.HttpServletResponse;

@Controller
public class UserController {

    @Autowired
    UserValidator validator;
    @Autowired
    LoginValidator loginValidator;
    @Autowired
    OrderValidator orderValidator;

    /**
     * Method to get login
     *
     * @param model
     * @param user
     * @param request
     * @return
     */
    @RequestMapping(value = { "/user/login.htm" }, method = RequestMethod.GET)
    private String getLogin(ModelMap model, User user, HttpServletRequest
request) {

        if (request.getSession().getAttribute("user") != null) {
            return "redirect: /edu/index.htm";
        }
        model.addAttribute(user);

        return "login";
    }

    /**
     * Method to handle login of a user
     *
     * @param user
     * @param bindingResult
     * @param status
     * @param userDao
     * @param req
     * @param res
     * @return
     */
    @RequestMapping(value = { "/user/login.htm" }, method = RequestMethod.POST)
    private String handleLogin(@ModelAttribute("user") User user, BindingResult
bindingResult, SessionStatus status,
        UserDao userDao, HttpServletRequest req, HttpServletResponse
res) {

        loginValidator.validate(user, bindingResult);
        if (bindingResult.hasErrors())
            return "login";
        Person person = userDao.getUserByEmail(user.getEmail());
        if (person == null) {
            req.setAttribute("errorMessage",
                "User with following email " + user.getEmail() +
                " is not found. Please signup and try again!!");
            return "error";
        }
    }
}
```

```

        if (!Hash.checkPasssword(user.getPassword(), person.getPassword()))
    {
        req.setAttribute("errorMessage", "Invalid username or password!
Try again!");
        return "error";
    }
    person.getPassword();
    createCookie(30 * 24 * 60 * 60, "user", person.getEmail(), "/",
res);
    createCookie(30 * 24 * 60 * 60, "name", person.getName().split("
")[0], "/", res);
    status.setComplete();
    return "redirect:" + "http://localhost:8080/edu/index.htm";
}

/**
 * method to get the signup attributes of a user
 *
 * @param model
 * @param user
 * @param req
 * @return
 */
@RequestMapping(value = { "/user/signup.htm" }, method = RequestMethod.GET)
private String getSignup(ModelMap model, User user, HttpServletRequest req)
{
    if (req.getSession().getAttribute("user") != null) {
        return "redirect: /edu/index.htm";
    }
    model.addAttribute(user);
    return "signup";
}

/**
 * method to handle the sign up of a user
 *
 * @param user
 * @param bindingResult
 * @param status
 * @param userDao
 * @param req
 * @param res
 * @return
 */
@RequestMapping(value = { "/user/signup.htm" }, method =
RequestMethod.POST)
private String handleSignup(@ModelAttribute("user") User user,
BindingResult bindingResult, SessionStatus status,
UserDAO userDao, HttpServletRequest req, HttpServletResponse
res) {

    validator.validate(user, bindingResult);
    if (bindingResult.hasErrors())
        return "signup";

    /**
     * handle the photo upload
     */

```



```

        CommonsMultipartFile iconPhoto = user.getPhoto();
        if (iconPhoto.getSize() != 0) {
            String picfileName = "img" + user.getId() +
Calendar.getInstance().getTimeInMillis()
                + iconPhoto.getContentType();
            File file = new File("C:/Users/deepa/Desktop/images",
picfileName);

            user.setPhotoFile(picfileName);
            try {
                iconPhoto.transferTo(file);
            } catch (IllegalStateException e) {
                e.printStackTrace();
                return "error";
            } catch (IOException e) {
                e.printStackTrace();
                return "error";
            }
        }
        user.setCreatedAt(new
Timestamp(Calendar.getInstance().getTimeInMillis()));
        user.setRole("user");
        user.setPassword(Hash.hashPassword(user.getPassword()));

        /**
         * putting model in my request scope
         */
        if (!userdao.addUser(user)) {
            req.setAttribute("errorMessage", "Please try again with
different email!! Error adding User ");
            return "error";
        }
        ;

        /**
         * creating a session based on cookies
         */
        createCookie(30 * 24 * 60 * 60, "user", user.getEmail(), "/", res);
        createCookie(30 * 24 * 60 * 60, "name", user.getName().split("
")[0], "/", res);

        /**
         * cleaning up a session
         */
        status.setComplete();
        return "redirect:" + "http://localhost:8080/edu/index.htm";
    }

    /**
     * method to delete a user
     *
     * @param request
     * @return
     */
    @RequestMapping(value = "/user/delete.htm", method = RequestMethod.GET)
    public String deleteUser(HttpServletRequest request) {
        if (request.getSession().getAttribute("user") == null) {
            request.setAttribute("errorMessage", "Login before
continuing");
            return "error";
        }
    }

```

```
    }
    request.setAttribute("user", "user");
    return "removeUser";
}

/**
 * method to delete a user
 *
 * @param request
 * @param response
 * @param userDao
 * @return
 */
@RequestMapping(value = "/user/delete.htm", method = RequestMethod.POST)
public String handleDelete(HttpServletRequest request,
    HttpServletResponse response,
    UserDao userDao) {
    Person person =
userDao.getUserByEmail(request.getParameter("email"));
    if (person == null) {
        request.setAttribute("errorMessage",
            "User with email " +
request.getParameter("email") + " is not found. Please verify!!");
        return "error";
    }
    userDao.deletePerson(person);
    deleteCookie(request, response);
    return "redirect:" + "http://localhost:8080/edu/index.htm";
}

/**
 * method to get an account of a user
 *
 * @param request
 * @param userDao
 * @return
 */
@RequestMapping(value = "/user/account.htm", method = RequestMethod.GET)
public String account(HttpServletRequest request, UserDao userDao) {
    if (request.getSession().getAttribute("user") == null) {
        request.setAttribute("errorMessage", "Please login before
continuing");
        return "error";
    }
    Person person = userDao.getUserByEmail((String)
request.getSession().getAttribute("user"));
    if (person == null) {
        request.setAttribute("errorMessage", "Please login before
continuing");
        return "error";
    }
    request.setAttribute("user", person);
    return "userprofile";
}

/**
 * method to update a profile for the user
 *
 * @param userDao
```

```
* @param request
* @param response
* @return
*/
@RequestMapping(value = "/user/updateProfile.htm", method =
RequestMethod.POST)
public String handleUpdateProfile(UserDAO userdao, HttpServletRequest
request, HttpServletResponse response) {
    Person person = userdao.getUserByEmail((String)
request.getSession().getAttribute("user"));
    if ((request.getParameter("name").length() == 0 ||
request.getParameter("email").length() == 0)) {
        request.setAttribute("errorMessage", "Error updating your
profile. Please insert valid inputs");
        return "error";
    }
    if (userdao.updatePerson(person, request.getParameter("name"),
request.getParameter("email"))) {
        createCookie(30 * 24 * 60 * 60, "user", person.getEmail(),
"/", response);
        createCookie(30 * 24 * 60 * 60, "name",
person.getName().split(" ")[0], "/", response);
        return "redirect:" + "http://localhost:8080/edu/index.htm";
    } else {
        request.setAttribute("errorMessage", "Error updating your
profile. Please try with different email");
        return "error";
    }
}

/**
 * method to handle a logout of a user
 *
 * @param request
 * @param response
 * @return
 */
@RequestMapping(value = "/logout.htm", method = RequestMethod.GET)
public String handleLogout(HttpServletRequest request, HttpServletResponse
response) {
    deleteCookie(request, response);
    return "redirect:" + "http://localhost:8080/edu/index.htm";
}

/**
 * method to get cart of a current user
 *
 * @param request
 * @param userdao
 * @return
 */
@RequestMapping(value = "/user/cart.htm", method = RequestMethod.GET)
public String getCart(HttpServletRequest request, UserDAO userdao) {
    if (request.getSession().getAttribute("user") == null) {
        request.setAttribute("errorMessage", "Please login before
continuing");
        return "error";
    }
}
```

```
    }
    User user = userDao.getUserByEmail((String)
request.getSession().getAttribute("user"));
    request.setAttribute("cart", user.getCart());
    request.setAttribute("total", calculateCartTotal(user.getCart()));
    return "cart";
}

/**
 * method to calculate the total amount
 *
 * @param cart
 * @return
 */
public float calculateCartTotal(Map<Product, Integer> cart) {
    float sum = 0;
    for (Product prod : cart.keySet()) {
        sum += (prod.getPrice() * cart.get(prod));
    }
    return sum;
}

/**
 * method to handle a checkout of an order for an user
 *
 * @param request
 * @param userDao
 * @param model
 * @param order
 * @return
 */
@RequestMapping(value = "/user/checkout.htm", method = RequestMethod.GET)
public String getCheckOut(HttpServletRequest request, UserDao userDao,
ModelMap model, Order order) {
    if (request.getSession().getAttribute("user") == null) {
        request.setAttribute("errorMessage", "Please login before
continuing");
        return "error";
    }
    User user = userDao.getUserByEmail((String)
request.getSession().getAttribute("user"));
    if (user.getCart().size() == 0) {
        request.setAttribute("errorMessage",
"You have nothing in your cart. Please add to
cart and proceed to checkout!!");
        return "error";
    }
    model.addAttribute(order);
    request.setAttribute("total", calculateCartTotal(user.getCart()));
    return "checkout";
}

/**
 * method to process an order checkout based on the order detail object
 *
 * @param order
 * @param result
 * @param status
 * @param request
```

```

    * @param userdao
    * @param productdao
    * @param sellerdao
    * @param orderdao
    * @return
    */
    @RequestMapping(value = "/user/checkout.htm", method = RequestMethod.POST)
    public String processCheckout(@ModelAttribute("order") Order order,
    BindingResult result, SessionStatus status,
    HttpServletRequest request, UserDao userdao, ProductDAO
    productdao, SellerDAO sellerdao,
    OrderDAO orderdao) {
        orderValidator.validate(order, result);
        if (result.hasErrors()) {
            return "checkout";
        }
        User user = userdao.getUserByEmail((String)
    request.getSession().getAttribute("user"));

        /**
         * checking if all the quantity are available
         */
        System.out.println(user.getCart().size() + "first!!!!!!");
        for (Product product : user.getCart().keySet()) {
            Product prod =
    productdao.getProductById(product.getProduct_id());
            if (prod.getQuantity() < user.getCart().get(product)) {
                request.setAttribute("errormessage",
                    product.getName() + "is not available in
    given quantity.Please update your cart");
                return "error";
            }
        }

        /**
         * creating an order object
         */
        order.setOrderedDate(new
    Timestamp(Calendar.getInstance().getTimeInMillis()));
        order.setUser(user);
        Set<Seller> sellerList = new HashSet<Seller>();
        for (Product product : user.getCart().keySet()) {
            Seller seller = product.getSeller();
            productdao.updateProductQuantity(product,
    user.getCart().get(product));
            sellerList.add(seller);
        }
        order.setSellers(sellerList);
        order.setProducts(user.getCart());
        System.out.println(user.getCart().size() + "second!!!!!!");
        order.setTotalPrice(calculateCartTotal(user.getCart()));
        if (!orderdao.addOrder(order)) {
            request.setAttribute("errormessage", "Error placing order");
            return "error";
        }

        // Clear cart on checkout
        userdao.clearCart(user);
    }

```

```
        /**
         * order added to the user, seller and the database persisted
         */

        return "redirect: /edu/index.htm";
    }

    /**
     * method to get user orders
     *
     * @param request
     * @param userDao
     * @return
     */
    @RequestMapping(value = "/user/getOrders.htm", method = RequestMethod.GET)
    public String getUserOrders(HttpServletRequest request, UserDao userDao) {
        if (request.getSession().getAttribute("user") == null) {
            request.setAttribute("errorMessage", "Please login before
continuing");
            return "error";
        }
        User user = userDao.getUserByEmail((String)
request.getSession().getAttribute("user"));
        if (user == null) {
            request.setAttribute("errorMessage",
                "User" + (String)
request.getSession().getAttribute("user") + "does not exist");
            return "error";
        }
        request.setAttribute("orders", user.getOrders());
        return "showOrders";
    }

    /**
     * method to create a session using a cookie
     *
     * @param time
     * @param name
     * @param msg
     * @param path
     * @param response
     */
    public void createCookie(int time, String name, String msg, String path,
        HttpServletResponse response) {
        Cookie c = new Cookie(name, msg);
        c.setMaxAge(time);
        c.setPath(path);
        response.addCookie(c);
    }

    /**
     * method to delete a cookie when logged out
     *
     * @param request
     * @param response
     */
    private void deleteCookie(HttpServletRequest request, HttpServletResponse
response) {
        Cookie[] cookies = request.getCookies();
```

```
        for (Cookie cookie : cookies) {
            cookie.setPath("/");
            cookie.setValue("");
            cookie.setMaxAge(0);
            response.addCookie(cookie);
        }
    }

    /**
     * method to email the user to reset the password when they forget the
password
     *
     * @param request
     * @return
     */
    @RequestMapping(value = "/resetPassword.htm", method = RequestMethod.GET)
    private String getEmailToResetPage(HttpServletRequest request) {
        if (request.getSession().getAttribute("user") != null) {
            request.setAttribute("errorMessage", "Please logout to reset
password");
            return "error";
        }
        return "emailToReset";
    }

    /**
     * method to invoke an smtp server
     *
     * @param request
     * @param response
     * @param userdao
     * @return
     */
    @RequestMapping(value = "/resetPassword.htm", method = RequestMethod.POST)
    private String sendMailForReset(HttpServletRequest request,
    HttpServletResponse response, UserDao userdao) {
        if (request.getSession().getAttribute("user") != null) {
            request.setAttribute("errorMessage", "Please logout to reset
password");
            return "error";
        }
        String email = request.getParameter("email");
        Person person = userdao.getPersonByEmail(email);
        if (person == null) {
            request.setAttribute("errorMessage", "User does not exist");
            return "error";
        } else {
            Mail mail = new Mail(email);
            String subject = "Reset Password";
            String token = RandomTokenGenerator.gernerate();
            person.setResetToken(token);
            person.setResetExpiresAt(new
Timestamp(Calendar.getInstance().getTimeInMillis() + (10 * 60 * 1000)));
            userdao.mergePerson(person);
            mail.sendMail(subject, token);
        }
        return "redirect: /edu/index.htm";
    }
}
```

```
/**
 * method to get a reset password call request
 *
 * @param request
 * @param response
 * @param userdao
 * @return
 */
@RequestMapping(value = "/updatePassword.htm", method = RequestMethod.GET)
private String getResetPassword(HttpServletRequest request,
    HttpServletResponse response, UserDao userdao) {
    if (request.getSession().getAttribute("user") != null) {
        request.setAttribute("errorMessage", "Please logout to reset
password");
        return "error";
    }
    if (request.getParameter("token") == null) {
        request.setAttribute("errorMessage", "No token found");
        return "error";
    }
    String token = (String) request.getParameter("token");
    Person person = userdao.getPersonByResetToken(token);
    if (person == null) {
        request.setAttribute("errorMessage", "Invalid token");
        return "error";
    }
    Timestamp currentTs = new
Timestamp(Calendar.getInstance().getTimeInMillis());
    if (currentTs.compareTo(person.getResetExpiresAt()) <= 0) {
        request.setAttribute("email", person.getEmail());
        return "resetPassword";
    } else {
        request.setAttribute("errorMessage", "Token expired");
        return "error";
    }
}

/**
 * method to update password of a user
 *
 * @param request
 * @param response
 * @param userdao
 * @return
 */
@RequestMapping(value = "/updatePassword.htm", method = RequestMethod.POST)
private String updatePassword(HttpServletRequest request,
    HttpServletResponse response, UserDao userdao) {
    String password = request.getParameter("pword");
    String confirmPassword = request.getParameter("cpword");
    if (!password.equals(confirmPassword)) {
        request.setAttribute("errorMessage", "passwords does not
match");
        return "error";
    }
    password = Hash.hashPassword(password);
    Person person =
userdao.getPersonByEmail(request.getParameter("email"));
}
```



```
        if (person == null) {
            request.setAttribute("errorMessage", "Invalid email");
            return "error";
        }
        person.setPassword(password);
        person.setResetExpiresAt(null);
        person.setResetToken(null);
        if (userdao.mergePerson(person))
            return "redirect:" + "http://localhost:8080/edu/index.htm";
        else {
            request.setAttribute("errorMessage", "Error updating email,
please try again");
            return "error";
        }
    }
}
```