Foods R Us

An E-Commerce Website

Designed and Developed By

Ameen (cse: aljailaa)

Heny (cse:)

Savan (cse:savan414)

Submitted as the group project for the course

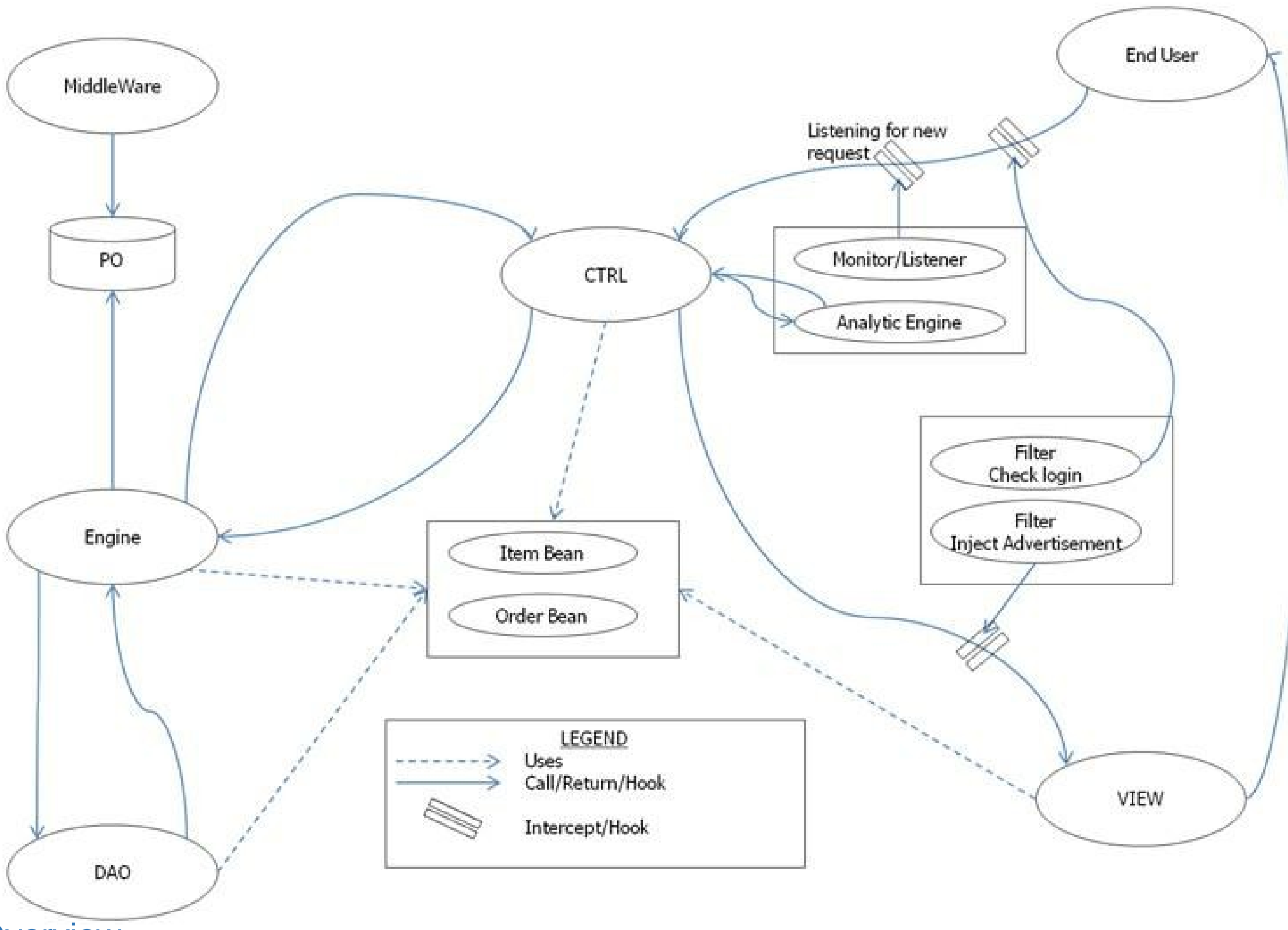
EECS4413 - Building E-Commerce Systems Prof.

H. Roumani - Fall 2018

Table of Contents

Design

Diagram:



Overview

From the diagram above we can conclude the following:

- Engine acts as the modal which is in charge of making sure that users have the data that they requested. To do this, the Engine communicates with the database using the DAO design pattern.
- When a user makes a purchase order, an XML file is generated and stored in folder structure denoted above as PO
- End users can contact the controller. However, there are filters and listeners which keep track of user usage and make sure that users can only access some of the services if they are authorized
- The application uses the MVC design pattern.
- We used JSPX to serve the view to the user. To make the view pretty, we utilized bootstrap

Implementation

Retrieving the Catalog information

As part of the DAO design pattern, we decided to create a bean class to represent a Catalog object, as a mean to pass catalog data to the view. This will facilitate the passing of catalogs information from the engine to the controller and to the view. It will be neat and easier for view to handle a Java bean object as well. The bean for each catalog will also contain information pertaining each catalog.

The common name 'agreed' by the controller and the view on retrieving the list of available catalogues is 'catalogues'.

Allowing End-user to view list of items by Category Option

To request browsing items of a category, we decided to supply a link for the requested category, with the parameter value of the respective category coded on the link on the view itself. The number of the catalogue is obtained from the database; hence it will always be the updated one. We will avoid to error such as unavailable or unknown/unrecognized category.

Role of the Browse controller

The Browse controller will determine if the request is for:

- main page' or 'fresh request' to view category list
- viewing items of a category
- viewing item with identified item number
- It will 'contact' the engine to supply

Role of ItemDAO

For the engine to access the items of a category, we have provided the ItemDAO as an interface to access the database. ItemDAO is a singleton class, whose services will be available to Engine. The main service it provides is returning:

- List of items (ItemBean) base on the category id requested
- List of all available categories (CatalogBean)
- Search and Return an item (ItemBean) base on item number requested.

Note: The entry of the item number has to be exactly the same as that in database, including case sensitivity. It will also pass any error message from the Engine as attribute '**error**'.

Role of Engine for the Browsing Service

The Engine will engage ItemDAO to retrieve:

- List of available Catalogues
- List of Items to display based on requested catalogue id.
- Search Item based on item number (which it will return to controller as a list of item, to make it consistent for display of items).

The engine will also pass the error message should it fail to retrieve information from ItemDAO.

Implementing the Service of Maintaining Shopping Cart

We have decided to use java object OrderBean as a mean to pass information of the End-User's Shopping Cart throughout the session. We have decided to name the attribute 'order', though it does not link well to mean Shopping Cart. However, it meant that it is in the form of java object OrderBean and it is in fact an open order.

We decide not to create another bean for shopping cart since the attributes of a shopping cart will also be the same as that of an order.

The use of order will also make it easy to update the open order/shopping cart, since it has all the relevant attributes as the total price, shipping cost etc.

The updating/calculating of the open order is done by the Engine. The ctrl **Add** will identify if it is a request to add additional item, or updating the cart/order. It will then call on the Engine, pass to the engine the OrderBean of the End-user, to process the adding of the selected item, or updating the cart/order.

Issues to consider when adding item:

We have initially omit the handling of negative entry of quantity. This is caught when we are testing our webapp.

The engine will need to check if the item added, is already in the cart. If it is, it will just increment the quantity in the new added request.

Issues to consider when updating cart/order:

We found updating the cart actually requires more tact than adding item to the order/cart.

- First we need to identify if it is a request to delete a row item.
- Second we need to find out if there is any changes in the quantity.

We decide, whenever there is a request to update the cart, the ctrl Add will:

- Create a toDelete List, and check if there is item(s) to be deleted. We make the Engine accept zero length list.
- Create the item list to get the items already in the cart and the quantity passed through the update request.

The Engine will then just update accordingly. We have created a separate helper method to delete item from cart, and to update quantity. We also create helper method to calculate the updated cost.

Implementing the Service of Login

To implement the required login for service of 'Check Out' or viewing 'My Account', we decided to implement to a filter. The filter will check for "Check Out" and "Account" service requests, and will divert to our Login servlet to direct End-User to login if they are not logged in.

Implementing the Service of Analytics

The service of Analytics is accessible via the address: http://localhost:4413/eFoods/ Analytics

We decide to package the analytics component into one package, consist of the listener to get and accumulate data for the analytics, the engine that will compute the requested analytics information, and to facilitate easier recording of the required analytics, we created a bean to track a user's activity throughout its session. Using this method, we do not need to create multiple attributes to store different kind of data: such as an attribute for list of checkout time, an attribute for list of adding item. We decide to create an entry per user, store it in a bean-like object UserActivity. The Servlet Context will have an attribute to store the information of each user. When computing the analytics data, we can just iterate through each user data to get the number.

Implementing the filter for Ad Hoc Advertisement

Since we are supposed not to make any changes to the servlet, we fixed in a hook on Add servlet, before it gets to the servlet. It will check if it is a request to add the 'hooked' item. If yes, we will poke in the advertise item to the request, which will get printed on the 'advertised' division.

Implementing the Middleware

To facilitate for platform independence, the main location of the PO (which is the main directory of the webapp) will be read as the main's arguments. The subdirectory where the PO is located is assumed to be known to the middleware I.e (WEBCONTENT/PO). The middleware will then iterate through the PO. For each PO xml file, it will extract the items and corresponding quantity listed in the PO, and after processing that, moved the PO into the 'processed PO' folder. This process though, has to been 'priorly agreed' between the webapp and the middleware, as the webapp will need to iterated between two folders when showing users their PO link.

A client checks out

Whenever a client presses the checkout button, a listener ensures that the client is logged in. If the client is not logged in, the listener will redirect the client to the login page. When the client successfully logs in, the client will be redirected to the checkout page again. The client will have to click on the confirm button to finish the purchase order.

At this point, the client's request will be redirected to the checkout controller. The checkout controller retrieves the order bean of the client from the session. Using the order bean, an XML file is created and stored under the unprocessed XML folder created dynamically whenever the server starts if it does not exist

A client visits the URL of a P/O

When the client visits the account or the checkout controller, the client will see all the processed and unprocessed Purchase Orders associated with this client. When a request comes into the “Account” controller with a the parameter “view”, the controller will call a method in the engine. This method will look under processed purchase order folder and the unprocessed purchase order folder. If the file is found with the same name as was provided in the view parameter, this file is converted into the order bean object. This bean is served to the user through a jspx file

The account controller

The account controller displays all the processed and unprocessed PO associated with the user. The implementation of this service is as follows. The code will iterate through the files under “PO” folder and “Unprocessed_PO“ folder. Each file is converted into an order bean. From the order bean, user name is extracted. If the user name matches the user name in the session, the name and the path of this bean will be visible to the user

The frontend Implementation

Bootstap was used to make the the code pretty.

The Team

Work division

Work amongst team members was divided as follows:

- Heny was in charge of browse, cart, analytics and advertisement
- Ameen was in charge of checkout, account controller, and checkout controller
- Savan was in charge of the frontend design

Member's Contributions

- Heny:
  Worked on the backend for browse, cart, analytic s, and advertisement. Contributed in fixing many bugs as they raise

- Ameen:
  Worked on creating the checkout Usecases. Added multiple features to the controller and contributed in fixing a lot of the bugs

- Savan:
  Worked on the front-end design and flow of the view, helped with redesigning of some structural components in the servlets to make it more modular and debugging the errors related with the view. Since none of us were comfortable with using CSS or Bootstrap it was better to have one person to learn the CSS and Bootstrap framework and get comfortable in it while others focus on making the backend work. Initially I thought it wouldn't be that hard to learn it but it quickly became a struggle sifting through and learning about 100's of different components and classes within the bootstrap framework. I have to choose between spending a lot of time watching the different tutorials on how to build a responsive view using CSS and Bootstrap or making an UI which is minimal but still elegant on the eyes. I went with the later one!

Meetings

Two meetings was arranged between the team members. In the first meeting, we discussed the requirements and distributed the work amongst in such a way that one member's work would be in conflict with other team members work significantly

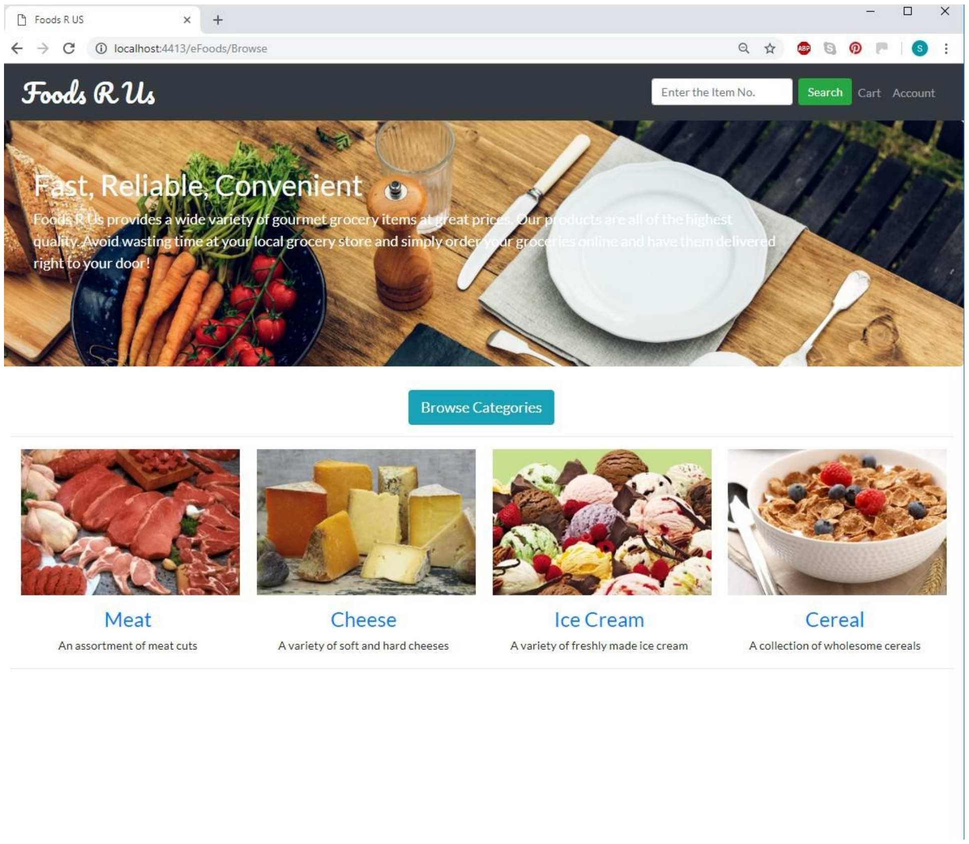
In the second meeting, we sit together briefly to have a plan on making the report. All team members were accessible through Email

Lesson learned

One of the lessons we learned is the importance of using version control when working in a project involving multiple team members. Unfortunately, we were not successful in utilizing this tool and as a result a lot of productive time was wasted merging fixes and keeping track of who did what

Output samples:

Page Upon Entering (Browse Main Page)



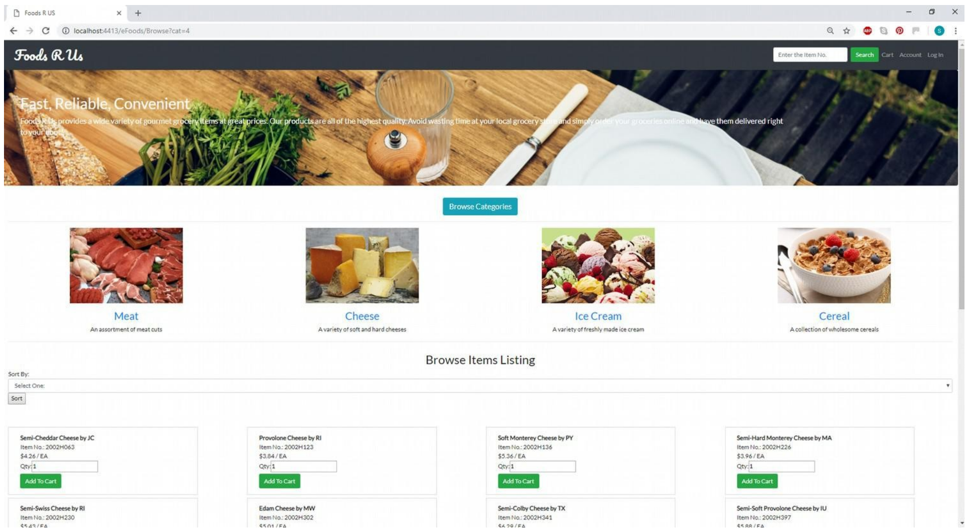
Fast, Reliable, Convenient

Foods R Us provides a wide variety of gourmet grocery items at great prices. Our products are all of the highest quality. Avoid wasting time at your local grocery store and simply order your groceries online and have them delivered right to your door!

Browse Categories

Meat	Cheese	Ice Cream	Cereal
An assortment of meat cuts	A variety of soft and hard cheeses	A variety of freshly made ice cream	A collection of wholesome cereals

Listing of items, with the option to sort available



Fast, Reliable, Convenient

Foods R Us provides a wide variety of gourmet grocery items at great prices. Our products are all of the highest quality. Avoid wasting time at your local grocery store and simply order your groceries online and have them delivered right to your door!

Meat	Cheese	Ice Cream	Cereal
An assortment of meat cuts	A variety of soft and hard cheeses	A variety of freshly made ice cream	A collection of wholesome cereals

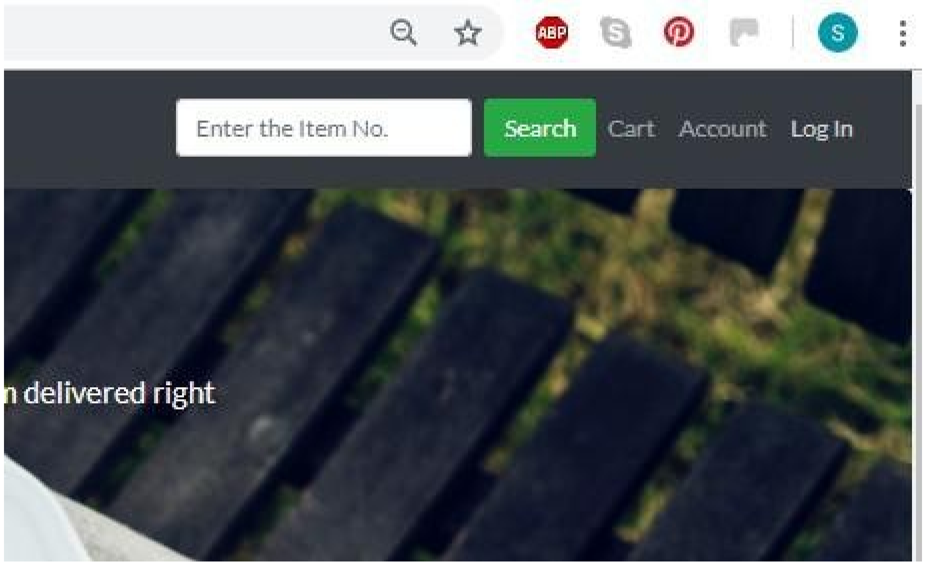
Browse Items Listing

Sort By: Select One:

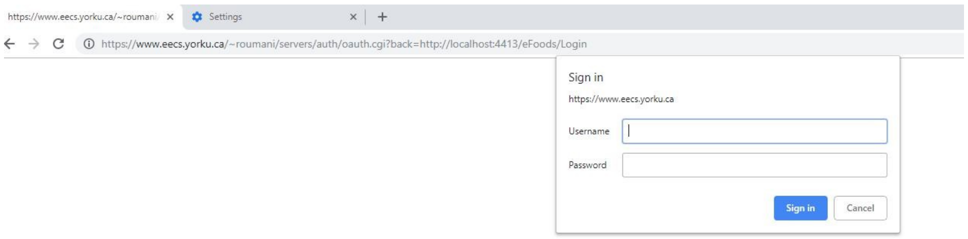
Sort

Semi-Cheddar Cheese by JC	Provolone Cheese by RI	Soft Monterey Cheese by PY	Semi-Hard Monterey Cheese by MA
Item No.: 2002H063	Item No.: 2002H123	Item No.: 2002H136	Item No.: 2002H226
$4.26 / EA	$3.84 / EA	$5.36 / EA	$3.96 / EA
Qty: 1	Qty: 1	Qty: 1	Qty: 1
Add To Cart	Add To Cart	Add To Cart	Add To Cart
Semi-Swiss Cheese by RI	Edam Cheese by MW	Semi-Colby Cheese by TX	Semi-Soft Provolone Cheese by IU
Item No.: 2002H230	Item No.: 2002H302	Item No.: 2002H341	Item No.: 2002H397
$5.43 / EA	$5.01 / EA	$6.29 / EA	$5.88 / EA

Prior to Log in, the Log in link will be provided



The Login Redirected view


https://www.eecs.yorku.ca/~roumani/servers/auth/oauth.cgi?back=http://localhost:4413/eFoods/Login

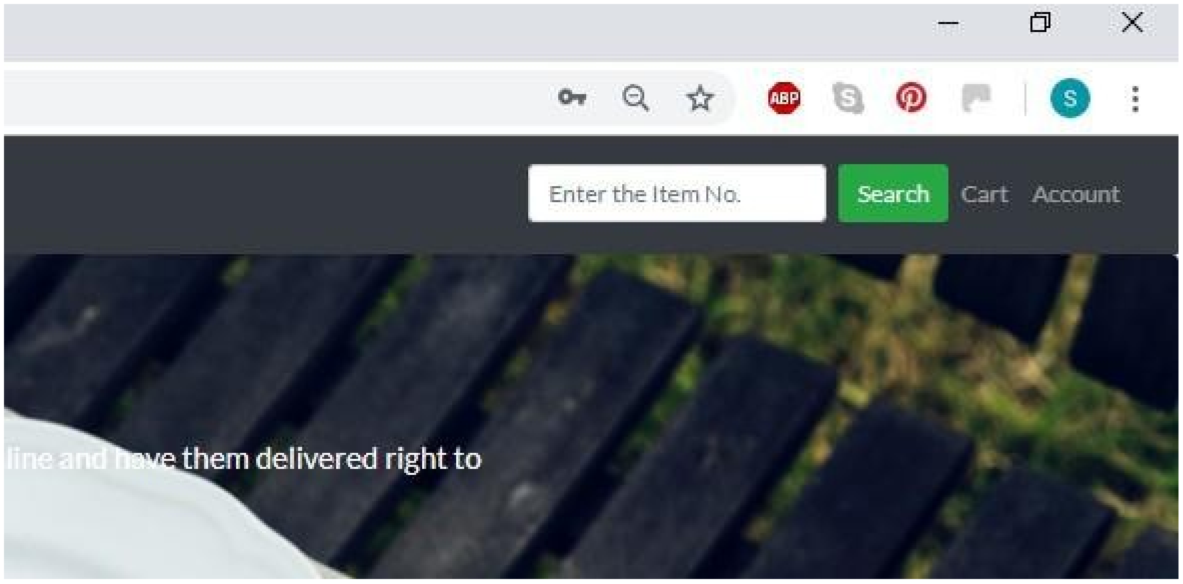
Sign in

https://www.eecs.yorku.ca

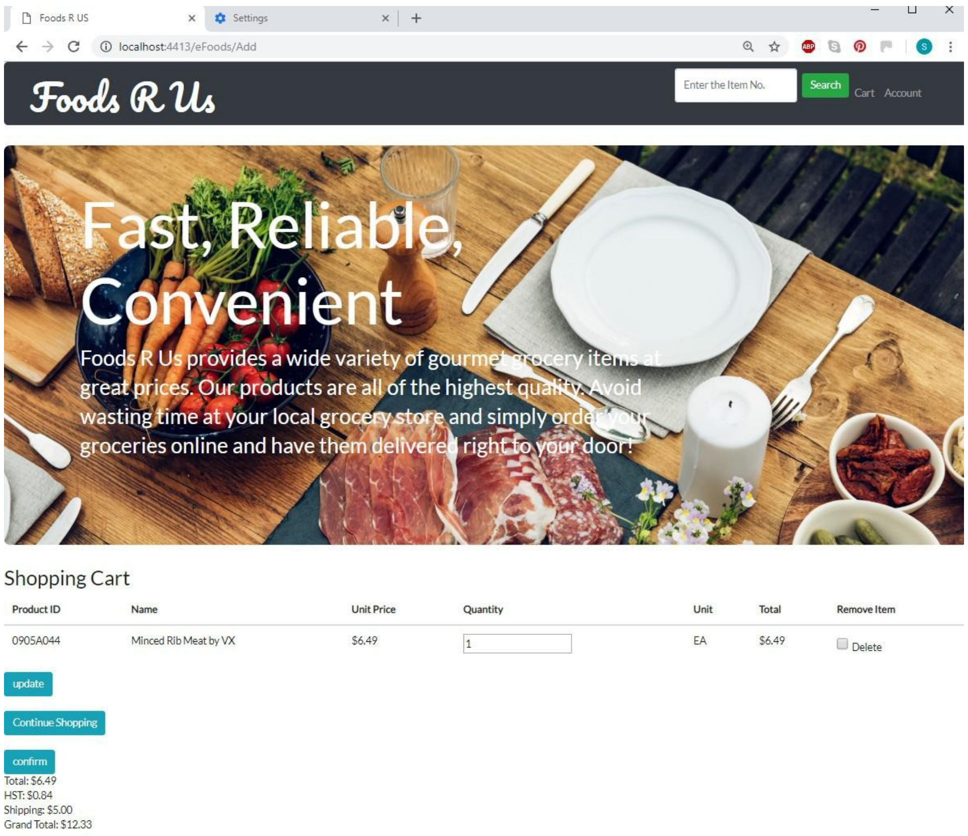
Username

Password

After the Login, the login link will not appear.



The Cart View



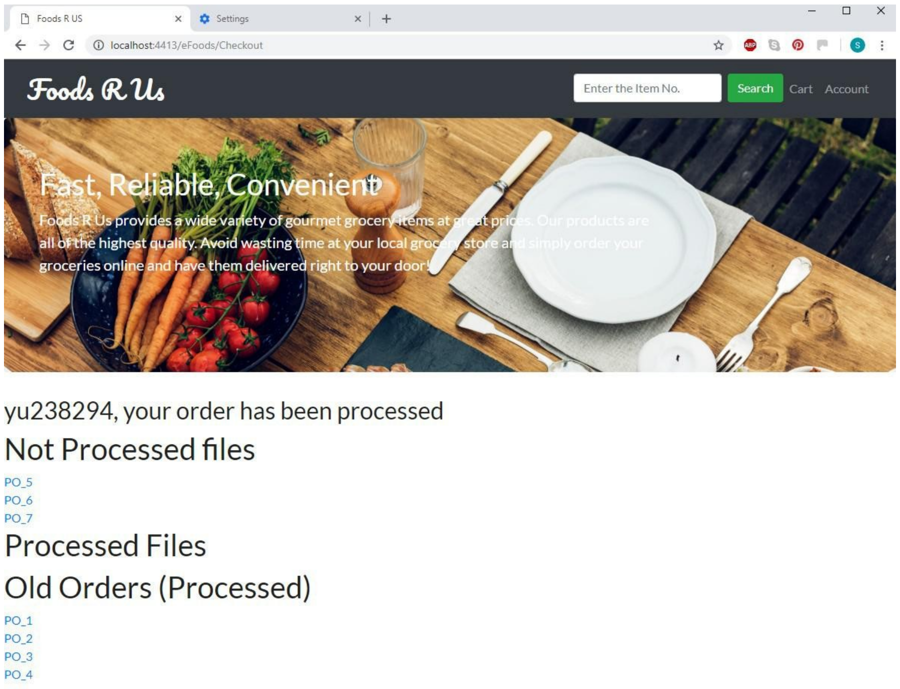
Foods R Us provides a wide variety of gourmet grocery items at great prices. Our products are all of the highest quality. Avoid wasting time at your local grocery store and simply order your groceries online and have them delivered right to your door!

Shopping Cart

Product ID	Name	Unit Price	Quantity	Unit	Total	Remove Item
0905A044	Minced Rib Meat by VX	$6.49	1	EA	$6.49	☐ Delete

Total: $6.49
HST: $0.84
Shipping: $5.00
Grand Total: $12.33

The view after a checkout was successful



Fast, Reliable, Convenient

Foods R Us provides a wide variety of gourmet grocery items at great prices. Our products are all of the highest quality. Avoid wasting time at your local grocery store and simply order your groceries online and have them delivered right to your door!

yu238294, your order has been processed

Not Processed files

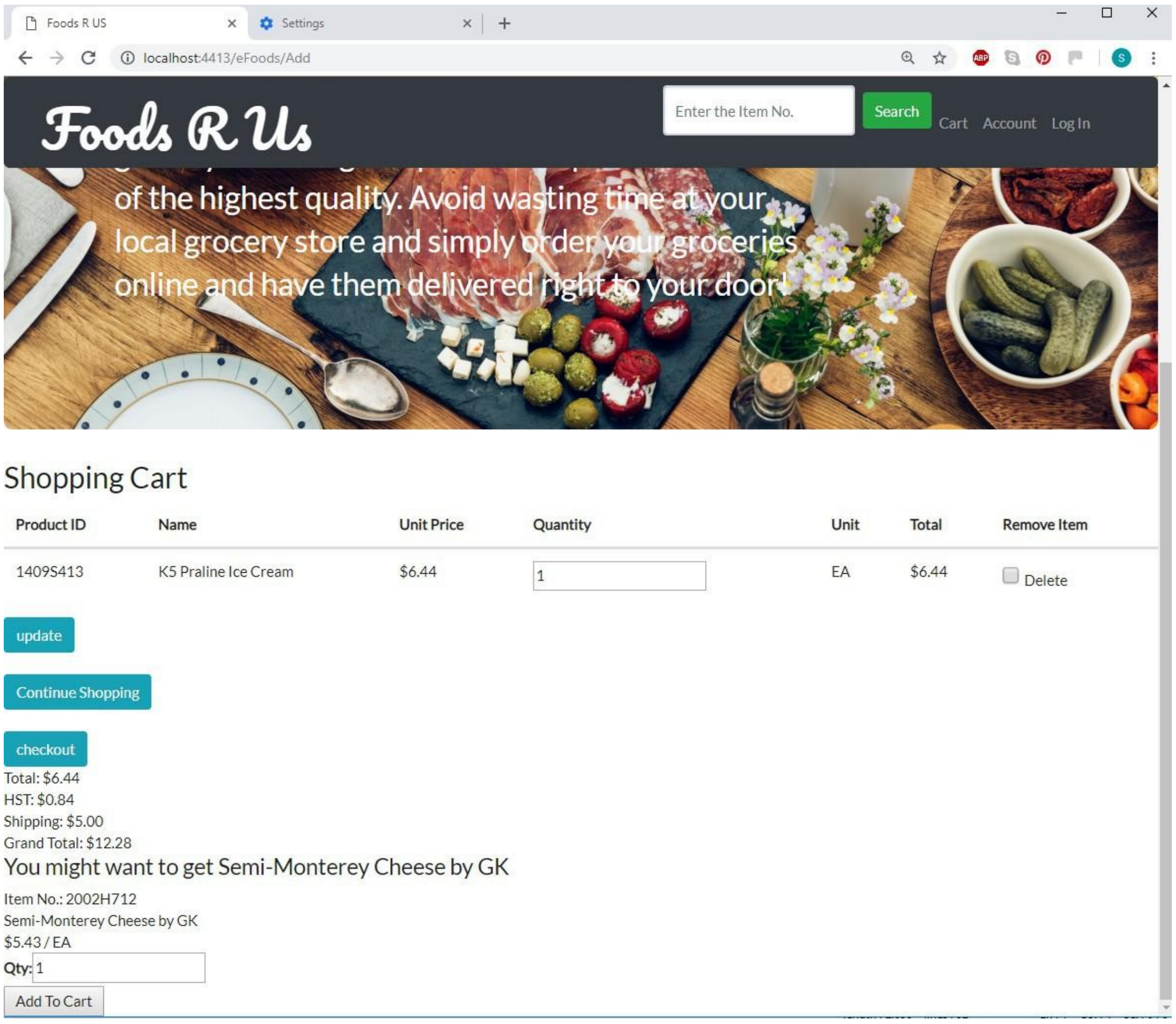
PO_5
PO_6
PO_7

Processed Files

Old Orders (Processed)

PO_1
PO_2
PO_3
PO_4

The Appearance of advertised item when End-user add into cart item number 1409S4


Foods R US
Settings
localhost:4413/eFoods/Add
Foods R Us
Enter the Item No.
Search
Cart
Account
Log In
of the highest quality. Avoid wasting time at your local grocery store and simply order your groceries online and have them delivered right to your door!

Shopping Cart

Product ID	Name	Unit Price	Quantity	Unit	Total	Remove Item
1409S413	K5 Praline Ice Cream	$6.44	1	EA	$6.44	☐ Delete

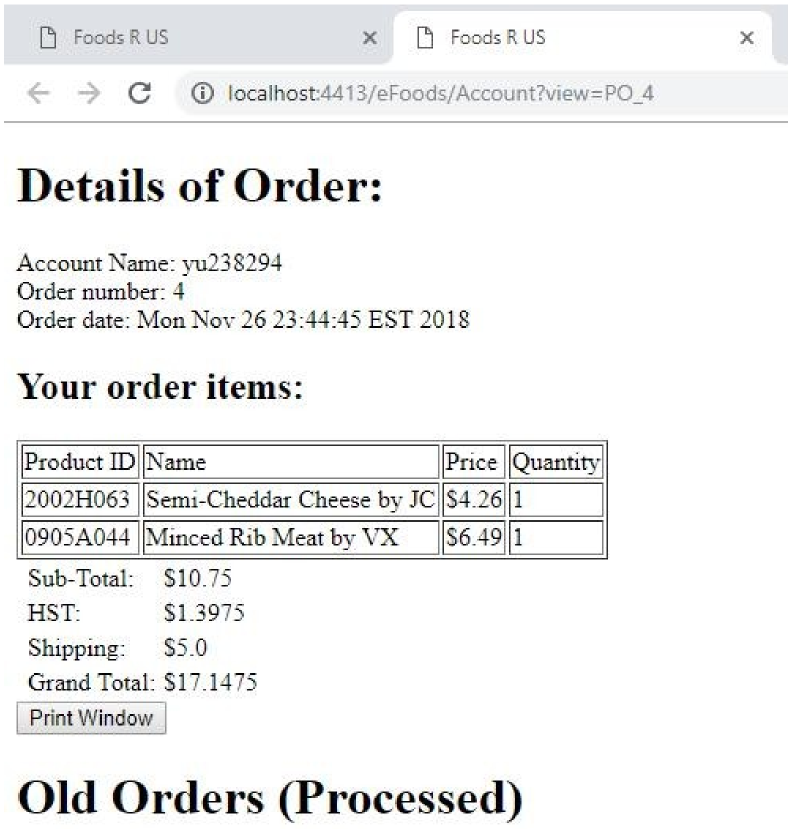
update
Continue Shopping
checkout

Total: $6.44
HST: $0.84
Shipping: $5.00
Grand Total: $12.28

You might want to get Semi-Monterey Cheese by GK

Item No.: 2002H712
Semi-Monterey Cheese by GK
$5.43 / EA
Qty: 1

Purchase order view



Details of Order:

Account Name: yu238294
Order number: 4
Order date: Mon Nov 26 23:44:45 EST 2018

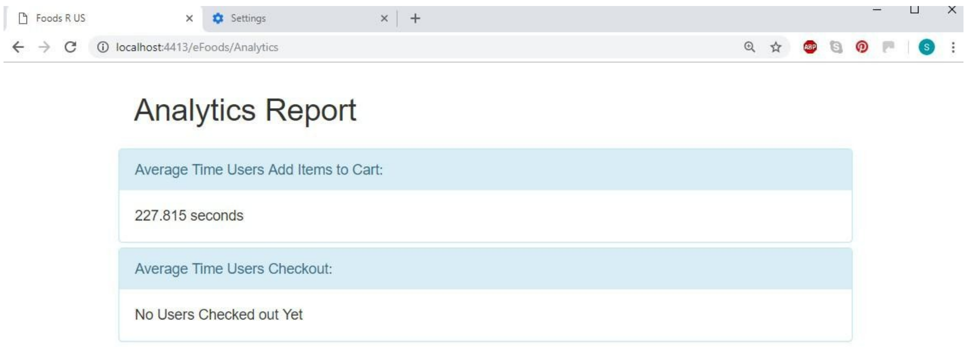
Your order items:

Product ID	Name	Price	Quantity
2002H063	Semi-Cheddar Cheese by JC	$4.26	1
0905A044	Minced Rib Meat by VX	$6.49	1

Sub-Total:	$10.75
HST:	$1.3975
Shipping:	$5.0
Grand Total:	$17.1475

Old Orders (Processed)

The Page for Admin to view Analytics Report



Analytics Report

Average Time Users Add Items to Cart:

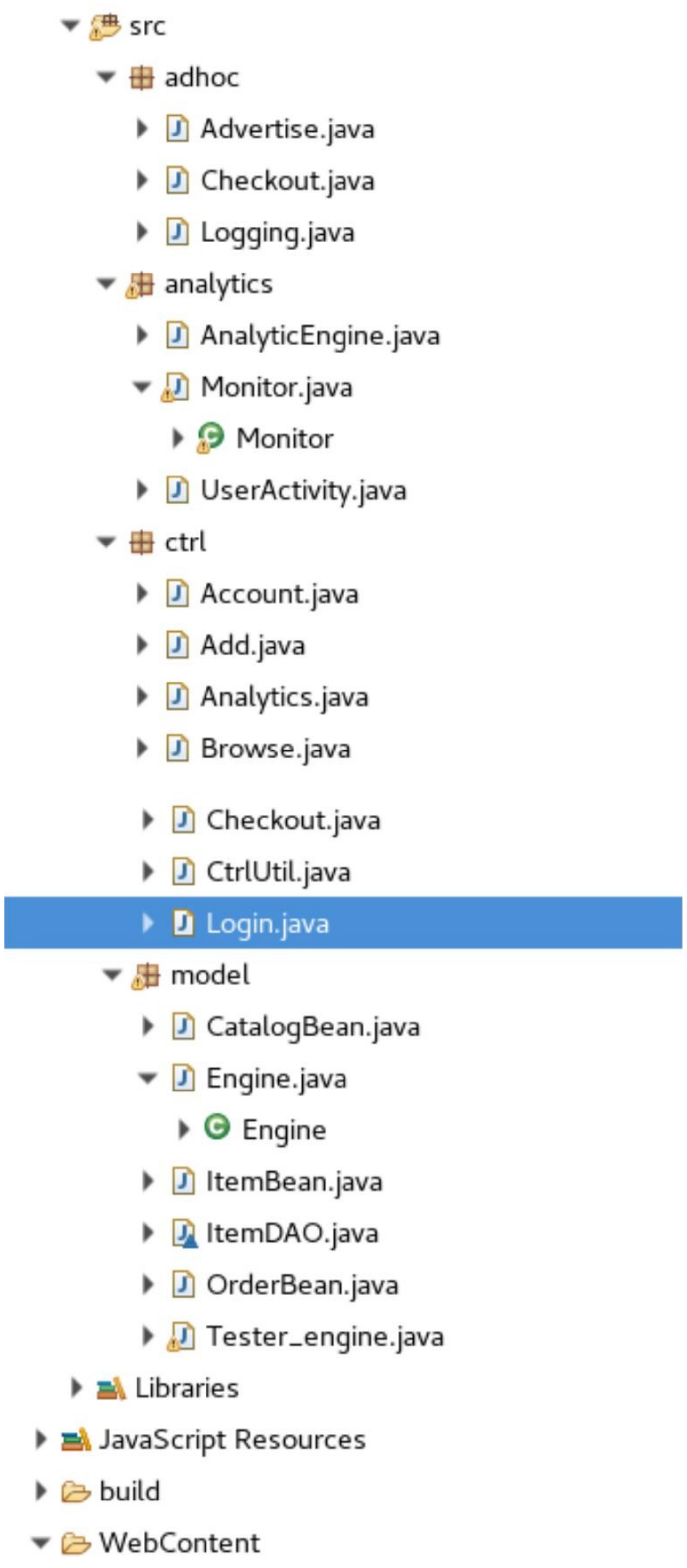
227.815 seconds

Average Time Users Checkout:

No Users Checked out Yet

Source Code

Tree



```
src
  adhoc
    Advertise.java
    Checkout.java
    Logging.java
  analytics
    AnalyticEngine.java
    Monitor.java
      Monitor
    UserActivity.java
  ctrl
    Account.java
    Add.java
    Analytics.java
    Browse.java
    Checkout.java
    CtrlUtil.java
    Login.java
  model
    CatalogBean.java
    Engine.java
      Engine
    ItemBean.java
    ItemDAO.java
    OrderBean.java
    Tester_engine.java
  Libraries
JavaScript Resources
build
WebContent
```

```
package adhoc;

import java.io.IOException;

/**
  * Servlet Filter implementation class Advertise
  */
@WebFilter(dispatcherTypes = {
//                              DispatcherType.REQUEST,
                                DispatcherType.FORWARD
                }
                , urlPatterns = { "/Advertise","/Add.jspx" }, servletNames = { "Add" }
)
public class Advertise implements Filter
{
        String number_hook_item ="1409S413";
        String number_advertise_item = "2002H712";
        public Advertise()
        {
        }

        public void destroy()
        {
        }

public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain) throws
IOException, ServletException
        {


                //check if it is item  1409S413
                HttpServletRequest req = (HttpServletRequest) request;
                String select_item = req.getParameter("select_item");
                if(select_item != null && select_item.equals("Add To Cart") &&
                                req.getParameter("number") != null && req.getParameter
("number").equals(number_hook_item))
                {
                        try
                        {
                                Engine engine = Engine.getInstance();
                                request.setAttribute("advertise", engine.getItem
(number_advertise_item));
                                request.setAttribute("toadvertise", "true");
                        }
                        catch (Exception e)
                        {
                                                System.out.println("Failed to hook advertisement");
                          }
                }
                chain.doFilter(request, response);
        }


        public void init(FilterConfig fConfig) throws ServletException {
        }

    }

```

```
package adhoc;

import java.io.IOException;

/**
 * Servlet Filter implementation class Checkout
 */
@WebFilter({"/Checkout","/Account"})
public class Checkout implements Filter {

	public Checkout()
	{
	}

	public void destroy()
	{
	}

public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain) throws
IOException, ServletException
	{
		HttpServletRequest req = (HttpServletRequest) request;
		if (req.getSession().getAttribute("user")!=null)
		{
			chain.doFilter(request, response);
		}
		else
		{
			((HttpServletResponse)response).sendRedirect("Login?login=true");
		}
	}

	public void init(FilterConfig fConfig) throws ServletException
	{
	}

}
```

```
package adhoc;

import java.io.IOException;

    /**
  * Servlet Filter implementation class Logging
     */
@WebFilter({ "/Logging", "/*" })
public class Logging implements Filter
{

        public Logging()
        {
        }

        public void destroy()
        {
        }

        public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain) throws IOException, ServletException
        {
                CtrlUtil.log_request(System.out, (HttpServletRequest) request, "Incoming");
                chain.doFilter(request, response);
                CtrlUtil.log_request(System.out, (HttpServletRequest) request, "Outgoing");
        }

        public void init(FilterConfig fConfig) throws ServletException
        {
        }
}
```

```java
package analytics;

import java.util.Map;

public class AnalyticEngine
{
	//list of attribute name
	public static final String ORDER = "order";
	public static final String CLIENTS = "clients";

	private static AnalyticEngine instance = null;

	private AnalyticEngine()
	{
	}

	public static AnalyticEngine getInstance()
	{
		if (instance == null)
			instance = new AnalyticEngine();
		return instance;
	}

	public double computeAveAddItemInterval(Map<String, UserActivity> users)
	{
		long totalTime = 0;
		int numberOfIntervals = 0;
		for(UserActivity user :users.values())
		{
			long last = user.startTime;
			int user_c = user.counter;
			if (user_c > 0)
			{
				long[] timeLog = user.addItemTime;
				for(int i = 0; i < user_c; i++)
				{
					totalTime = totalTime + timeLog[i] - last;
					last = timeLog[i];
					numberOfIntervals ++;
				}
			}
		}
		if (numberOfIntervals < 1 ) //in case there is no one has entered yet.
			return -1;
		double ave = (double)totalTime/1000/numberOfIntervals;
		return ave;
	}

	public double computeCheckOutInterval(Map<String, UserActivity> users)
	{
		long totalTime = 0;
		int numberOfUsers = 0;
		for(UserActivity user: users.values())
		{
			long first = user.startTime;
			long checkOut = user.checkoutTime;
			if(checkOut > 0 )
			{
				totalTime = totalTime + checkOut - first;
				numberOfUsers ++;
			}
		}
		if (numberOfUsers < 1 ) //in case there is no one has entered yet.
			return -1;
		return (double)totalTime / numberOfUsers;
	}





}
```

```java
package analytics;

import java.util.ArrayList;

/**
 * Application Lifecycle Listener implementation class Monitor
 *
 */
@WebListener
public class    Monitor implements
        HttpSessionListener, ServletRequestListener
{
        public Monitor()
        {
        }

        public void sessionCreated(HttpSessionEvent se)
        {
                attachOrder (se);
                addUser(se);
        }
        //attach Order Object for the new session(new user)
        private void attachOrder(HttpSessionEvent se)
        {
                //attach order (shopping cart) object to Session context
                HttpSession session = se.getSession();
                List<ItemBean> items = new ArrayList<ItemBean>();
                OrderBean order = new OrderBean(items);
                session.setAttribute(AnalyticEngine.ORDER, order);
        }
        //Add the new session/user to the list of 'userActivity' stored in Servlet contextt
        private void addUser(HttpSessionEvent se)
        {
                //Add client to list of "clients" in ServletContext to monitor all users activity
                ServletContext servlet =se.getSession().getServletContext();
                if (servlet.getAttribute(AnalyticEngine.CLIENTS) == null)
                {
                        servlet.setAttribute(AnalyticEngine.CLIENTS, new HashMap<String, UserActivity>());
                }

                @SuppressWarnings("unchecked")
Map<String, UserActivity> clients = (Map<String, UserActivity>)servlet.getAttribute("clients");
                clients.put(se.getSession().getId(),
                                new UserActivity(se.getSession().getId(),
System.currentTimeMillis()));
        }


        public void requestInitialized(ServletRequestEvent sre)
        {
                //check if it is addItem Request
HttpServletRequest request = (HttpServletRequest) sre.getServletRequest();
                if(request.getParameter("select_item")!= null)
                        logAddItem(sre);
        }

        //to log the adding of item by a user;
        private void logAddItem(ServletRequestEvent sre)
        {
                ServletContext servlet = sre.getServletContext();
String id = ((HttpServletRequest) sre.getServletRequest()).getSession ().getId();

                @SuppressWarnings("unchecked")
                Map<String, UserActivity> users =
                                (Map<String, UserActivity>)servlet.getAttribute
(AnalyticEngine.CLIENTS);
                UserActivity user = users.get(id);
                user.logAddItem(System.currentTimeMillis());
        }

        private void logCheckOut(ServletRequestEvent sre)
        {
                //TODO: has not implemented the invoking of checkout event.
                ServletContext server = sre.getServletContext();
                @SuppressWarnings("unchecked")
                Map<String, UserActivity> users =
                                (Map<String, UserActivity>) server.getAttribute
(AnalyticEngine.CLIENTS);

                UserActivity user = users.get
(((HttpServletRequest)sre.getServletRequest()).getSession().getId());
                user.checkoutTime = System.currentTimeMillis();
        }
}


```

```java
package analytics;

public class UserActivity
{
	String sessionId;
	long startTime;
	final int SIZE = 100;
	long[] addItemTime;
	int counter;
	int limitCounter;
	long checkoutTime;

	//Construct a UserActivity object, initializing sessionId and startTime
	UserActivity(String sessionId, long startTime)
	{
		this.sessionId = sessionId;
		this.startTime = startTime;
		addItemTime = new long[SIZE];
		counter = 0;
		limitCounter = SIZE;
	}

	void logAddItem (long time)
	{
		addItemTime[counter++] = time;
		if (counter > limitCounter -1)
		{
			limitCounter = limitCounter + SIZE;
			long[] newArray = new long[limitCounter];
			System.arraycopy(addItemTime, 0, newArray, 0, counter);
			addItemTime = newArray;
		}
	}
}
```

Account.java

```java
package ctrl;

import java.io.File;


@WebServlet("/Account")
public class Account extends HttpServlet {
        private static final long serialVersionUID = 1L;

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {

request.getSession().setAttribute("back", request.getRequestURL ().toString());

                String user = (String) request.getSession().getAttribute("user");
                request.setAttribute("user", user);

                        try {
                            Engine engine = Engine.getInstance();
                            if (request.getParameter("view") != null) {
                                                //Server the PO view
                                        String view = request.getParameter("view") + ".xml";
                                        File file = new File(engine.getXmlFolderPath() + view);
                                        if (file.exists()) {
                                                        OrderBean order =
engine.convertFromXMLFileToObject(file, user);
                                                        request.setAttribute("order", order);
                                        } else {
                                                  file = new File(engine.getXmlPOProcessedFolderPath()
+ view);
                                                  if (file.exists()) {
                                                          OrderBean order =
engine.convertFromXMLFileToObject(file, user);
                                                          request.setAttribute("order", order);
                                                  } else {
                                                          request.setAttribute("error", "File not found");
                                                  }

                                        }

                                }
                                else {
                                        request.setAttribute("fileNames", engine.getXMLLinks(user,
engine.getXmlFolderPath()));
                                        request.setAttribute("processedfileNames",
engine.getXMLLinks(user, engine.getXmlPOProcessedFolderPath()));
                                  }
                        } catch (Exception e) {
```

```
				e.printStackTrace();
			}


this.getServletContext().getRequestDispatcher("/
Account.jspx").forward(request, response);

	}


	protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
		doGet(request, response);
	}

}
```

```java
package ctrl;

import java.io.IOException;

@WebServlet("/Add")
public class Add extends HttpServlet
{
	private static final long serialVersionUID = 1L;

	public Add()
	{
		super();
	}

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
	{
		HttpSession session = request.getSession();
		session.setAttribute("back", request.getRequestURL().toString());
		if (session.getAttribute("user") != null) {
			String user = (String) session.getAttribute("user");
			request.setAttribute("user", user);
		}
		OrderBean order = ((OrderBean) session.getAttribute("order"));
		try
		{
			Engine engine = Engine.getInstance();

			if(request.getParameter("select_item")!=null)//add item
			{
				String number = request.getParameter("number");
				String quantity = request.getParameter("qty");
				engine.orderAddItem(order, number, quantity);
			}
			if(request.getParameter("update")!=null)//update carts
			{
				List<String> toDelete = new ArrayList<String>();
				Map<String, String> itemsQty = new HashMap<String, String>();
				List<String> parameters = Collections.list(request.getParameterNames());
				for(String parameter: parameters)
				{
					if (parameter.startsWith("delete"))
						toDelete.add(request.getParameter(parameter));
					else if(parameter.matches("\\d{4}[A-Z|a-z]\\d{3}"))
						itemsQty.put(parameter, request.getParameter(parameter));
				}
```

```
                    engine.updateOrder(order, itemsQty, toDelete);
                }
            }
            catch (Exception e)
            {
                request.setAttribute("error", e.getMessage());
            }

            String  checkoutName = "checkout";
            if (session.getAttribute("user") != null)
                checkoutName = "confirm";
            request.setAttribute("checkout", checkoutName);

            request.setAttribute("order", order);
this.getServletContext().getRequestDispatcher("/Add.jspx").forward (request, response);

        }

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException
        {
            doGet(request, response);
        }

}
```

```
package ctrl;

import java.io.IOException;

/**
 * Servlet implementation class Analytics
 */
@WebServlet("/Analytics")
public class Analytics extends HttpServlet
{
        private static final long serialVersionUID = 1L;

        public Analytics()
        {
                super();
        }

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException
        {
                AnalyticEngine engine = AnalyticEngine.getInstance();

                @SuppressWarnings("unchecked")
                double aveAddItem = engine.computeAveAddItemInterval((Map<String,
UserActivity>)
request.getServletContext().getAttribute (AnalyticEngine.CLIENTS));

                System.out.println("aveAddItem=" + aveAddItem);
                if(aveAddItem > 0)
                {
                        request.setAttribute("aveAddItem", aveAddItem);
                }
                else
                {
                        request.setAttribute("rpt_AddItem", "No Users Yet");
                }


                @SuppressWarnings("unchecked")
                double aveCheckout = engine.computeCheckOutInterval(((Map<String,
UserActivity>)
request.getServletContext().getAttribute (AnalyticEngine.CLIENTS)));

                System.out.println("aveCheckout=" + aveCheckout);
                if(aveCheckout > 0)
                {
                        request.setAttribute("aveCheckout", aveCheckout);
                }
```

```
		else
		{
			request.setAttribute("rpt_Checkout", "No Users Checked out Yet");
		}


this.getServletContext().getRequestDispatcher("/
Analytics.jspx").forward(request, response);
	}

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException
	{
		doGet(request, response);
	}

}
```

Browse.java

```
package ctrl;

import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import model.Engine;

@WebServlet({ "/Browse"})
public class Browse extends HttpServlet
{
	private static final long serialVersionUID = 1L;

	public Browse()
	{
		super();
	}

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException
	{
		try
		{
			Engine engine = Engine.getInstance();
			request.setAttribute("catalogs", engine.getCatalogs());
			if(request.getParameter("orderBy")!= null)
			{
				String cat = request.getParameter("cat");
				if (cat != null && !cat.isEmpty() && !cat.equals("NONE"))
				{
					request.setAttribute("items",engine.getItems(cat,
request.getParameter("orderBy")));
					request.setAttribute("cat",request.getParameter("cat"));
				}
				else if (request.getParameter("number")!= null)
				{
					request.setAttribute("items",engine.searchItem
(request.getParameter("number")));
					request.setAttribute("number",request.getParameter
("number"));
				}
				request.getParameter("orderBy");
			}
			else if(request.getParameter("cat")!=null)
			{
```

```
request.setAttribute("items",engine.getItems (request.getParameter("cat")));

                                        request.setAttribute("cat",request.getParameter("cat"));
                                }
                                else if(request.getParameter("select_item").equals("search"))
                                {
request.setAttribute("items",engine.searchItem (request.getParameter("number")));

                                        request.setAttribute("number",request.getParameter("number"));
                                }

                        }
                        catch (Exception e)
                        {
                                request.setAttribute("error", e.getMessage());
                        }
                        if (request.getSession().getAttribute("user") != null) {
                                String user = (String) request.getSession().getAttribute("user");
                                request.setAttribute("user", user);
                        }

this.getServletContext().getRequestDispatcher("/Browse.jspx").forward (request, response);

                }

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException
                {
                        doGet(request, response);
                }

}
```

```java
package ctrl;

import java.io.IOException;

@WebServlet({ "/Browse"})
public class Browse extends HttpServlet
{
        private static final long serialVersionUID = 1L;

        public Browse()
        {
                super();
        }

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
        {
                try
                {

                        if (request.getSession().getAttribute("user") != null) {
                                String user = (String) request.getSession().getAttribute("user");
                                request.setAttribute("user", user);
                        }

                        Engine engine = Engine.getInstance();
                        request.setAttribute("catalogs", engine.getCatalogs());
                        if(request.getParameter("orderBy")!= null)
                        {
                                String cat = request.getParameter("cat");
                                if (cat != null && !cat.isEmpty() && !cat.equals("NONE"))
                                {
                                        request.setAttribute("items",engine.getItems(cat, request.getParameter("orderBy")));
                                        request.setAttribute("cat",request.getParameter("cat"));
                                }
                                else if (request.getParameter("number")!= null)
                                {
                                        request.setAttribute("items",engine.searchItem(request.getParameter("number")));
                                        request.setAttribute("number",request.getParameter("number"));
                                }
                        }
                        else if(request.getParameter("cat")!=null)
                        {
                               request.setAttribute("items",engine.getItems
(request.getParameter("cat")));
                        request.setAttribute("cat",request.getParameter("cat"));
                    }
                    else if(request.getParameter("select_item").equals("search"))
                    {
request.setAttribute("items",engine.searchItem (request.getParameter("number")));

                        request.setAttribute("number",request.getParameter("number"));
                    }

                    request.getSession().setAttribute("back", request.getRequestURL
());
                }
                catch (Exception e)
                {
                    request.setAttribute("error", e.getMessage());
                }
this.getServletContext().getRequestDispatcher("/Browse.jspx").forward (request, response);

        }

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException
        {
                doGet(request, response);
        }

}
```

Login.java

```java
package ctrl;

import java.io.IOException;

@WebServlet({"/Login"})
public class Login extends HttpServlet {
    private static final long serialVersionUID = 1L;

    public Login()
    {
        super();
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
    {

        if (request.getParameter("user")!=null)
        {
            String user= request.getParameter("user");
            String name = request.getParameter("name");
            String back = (String) request.getSession().getAttribute("back");
            request.getSession().setAttribute("user", user);
            request.getSession().setAttribute("name",name);
            request.setAttribute("user", user);
            request.setAttribute("name", name);
            //this.getServletContext().getRequestDispatcher("/Login.jspx").forward(request, response);
            response.sendRedirect(back);
        }
        else if(request.getParameter("login")!=null)
        {
            String url = "https://www.eecs.yorku.ca/~roumani/servers/auth/oauth.cgi"+
                        "?back="+(request.getRequestURL());
            response.sendRedirect(url);
        }
        else
        {
            request.setAttribute("error", "Wrong Account Number and/or PIN.\\nPlease check your Account and PIN number you have entered");
            this.getServletContext().getRequestDispatcher("/Login.jspx").forward(request, response);
        }

    }

    protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException
        {
                doGet(request, response);
        }

}
```

```java
package model;

import java.io.File;

public class Engine
{
	static final double HST_RATE = 0.13;
	static final double MIN_SHIPPING_WAIVER = 70;
	static final double SHIPPING_COST = 5;
	static Engine engine = null;

	private ItemDAO itemDAO;
	private String xmlPOFolderPath;
	private String xmlPOProcessedFolderPath;
	private int count = 0;

	private Engine() throws Exception
	{
		itemDAO = ItemDAO.getInstance();
		initializeFilePath();
		initializeCount();
	}
	public synchronized static Engine getInstance() throws Exception
	{
		try
		{
			if (engine == null)
				engine = new Engine();
			return engine;
		}
		catch (Exception e)
		{
			throw new Exception ("Issue at instatiating engine", e);
		}
	}
	public List<ItemBean> getItems(String catalog) throws Exception
	{
		try
		{
			return itemDAO.getItems(catalog, null);
		}
		catch (Exception e)
		{
			throw new Exception ("Fail to get items", e);
		}
	}
	public List<ItemBean> getItems(String catalog, String orderBy) throws Exception
	{
		try
		{
			return itemDAO.getItems(catalog, orderBy);
		}
		catch (Exception e)
		{
			throw new Exception ("Fail to get items", e);
		}
	}

	public List<ItemBean> searchItem(String number) throws Exception
	{
		List<ItemBean> result = new ArrayList<ItemBean>();
		result.add(getItem(number));
		//TODO
		return result;
	}
	public ItemBean getItem(String number) throws Exception
	{
		try
		{
			return itemDAO.getItem(number);
		}
		catch (Exception e)
		{
			throw new Exception("Fail to find item of number " + number);
		}
	}

	public List<CatalogBean> getCatalogs() throws Exception
	{
		try
		{
			return itemDAO.getCatalogs();
		}
		catch (Exception e)
		{
			throw new Exception ("Fail to get catalogs",e);
		}
	}

	//return the 'existing item' in the stact if found, else return null;
	private ItemBean isExistingItem(List<ItemBean> items, String number)
	{
		ItemBean item = null;
		for(int i = 0; i < items.size() && item == null; i++)
		{
```

```java
            if (items.get(i).getNumber().equals(number))
                item = items.get(i);
        }
        return item;
    }

    private void updateOrderTotalPrice(OrderBean order)
    {
        List<ItemBean> items = order.getItems();
        double total = 0;
        for(ItemBean item: items)
        {
            total = total + (item.getPrice() * item.getQuantity());
        }
        order.setTotal(total);
        order.setHST(total*HST_RATE);

        //set shipping cost
        if (total == 0 || total > MIN_SHIPPING_WAIVER )
            order.setShipping(0);
        else
            order.setShipping(SHIPPING_COST);

order.setGrandTotal(order.getTotal() + order.getHST() + order.getShipping());
    }

public OrderBean orderAddItem(OrderBean order, String number, String qty) throws Exception
    {
        int quantity;
        try
        {
            quantity = Integer.parseInt(qty);
        }
        catch(Exception e)
        {
            throw new Exception("Invalid quantity entries " + qty);
        }

        try
        {
            ItemBean item = isExistingItem(order.getItems(), number);
            if(item != null)
            {
                item.setQuantity(item.getQuantity() + quantity);
            }
            else
            {
                item = this.getItem(number);
                item.setQuantity(quantity);
                order.getItems().add(item);
            }
            updateOrderTotalPrice(order);
            return order;

        }
        catch (Exception e)
        {
throw new Exception("Fail to add "+ qty +" of item number " + number + " into order");
        }
    }

    private void orderDelItems(OrderBean order, List<String> toDelete)
    {
        List<ItemBean> items = order.getItems();
        for (String e: toDelete)
        {
            ItemBean item = isExistingItem(items, e);
            if(item !=null)
                items.remove(isExistingItem(order.getItems(), e));
        }
    }
private void orderUpdateQty (OrderBean order, Map<String, String> itemsQty) throws
Exception
    {
        List<ItemBean> items = order.getItems();
        Set<String> toUpdate = itemsQty.keySet();
        for(String itemNo: toUpdate)
        {
            ItemBean item = isExistingItem(items, itemNo);
            if (item!=null)
            {
                int quantity;
                try
                {
                    quantity = Integer.parseInt(itemsQty.get(itemNo));
                    if (quantity < 0 )
                        throw new Exception ();
                }
                catch (Exception e)
                {
                    throw new Exception ("Invalid quantity entries!");
                }
                if (quantity == 0)
```

```
                        items.remove(item);
                    item.setQuantity(quantity);
            }
        }
    }

public OrderBean updateOrder(OrderBean order, Map<String, String> itemsQty,
List<String> toDelete) throws Exception
    {
        orderDelItems(order, toDelete);
        orderUpdateQty(order, itemsQty);
        updateOrderTotalPrice(order);
        return order;
    }

public void jaxbObjectToXML(OrderBean order, String id, String path) throws Exception

     {
                //Create JAXB Context
                JAXBContext jaxbContext = JAXBContext.newInstance
(OrderBean.class);

                //Create Marshaller
                Marshaller jaxbMarshaller = jaxbContext.createMarshaller();

                //Required formatting??
                jaxbMarshaller.setProperty(Marshaller.JAXB_FORMATTED_OUTPUT,
Boolean.TRUE);

               //Store XML to File
                File filePath = new File(path + String.format("%s.xml", id));
                if (!filePath.exists()) {
                        filePath.createNewFile();
                }

                //Writes XML file to file-system
                jaxbMarshaller.marshal(order, filePath);
      }


    public OrderBean convertFromXMLFileToObject(File file,String user) {
        JAXBContext jaxbContext;
        try
        {
            jaxbContext = JAXBContext.newInstance(OrderBean.class);
            Unmarshaller jaxbUnmarshaller = jaxbContext.createUnmarshaller();
            OrderBean order = (OrderBean) jaxbUnmarshaller.unmarshal(file);

				if (order.getAccount() != null && order.getAccount().equals(user))
					return order;
		}
		catch (JAXBException e)
		{
			e.printStackTrace();
		}

		return null;

	}
	public ArrayList<String> getXMLLinks(String user, String path) {
		ArrayList<String> names = new ArrayList<>();
		File[] files = new File(path).listFiles();

		for (File file : files) {
			if (!file.isDirectory() && file.getName().contains(".xml")) {

				OrderBean order = convertFromXMLFileToObject(file, user);
				if (order != null)
				{
					String[] f = file.getName().split(".xml");
					names.add(f[0]);
				}
			}
		}
		return names;
	}


	public String getXmlPOProcessedFolderPath() {
		return xmlPOProcessedFolderPath;
	}

	public String getXmlFolderPath() {
		return xmlPOFolderPath;
	}

	public int increment() {
		count++;
		return count;
	}



	private void initializeFilePath() {

		if (xmlPOFolderPath == null) {
			xmlPOFolderPath = System.getProperty("user.dir") + "/appData/
	PO/";
			File filePath = new File(xmlPOFolderPath);
			//Create folder if they don't exist
			if (!filePath.exists()) {
				try{
					if (filePath.getParentFile().exists())
						filePath.getParentFile().mkdirs();

					filePath.mkdirs();
				}

				catch(SecurityException se){
					//handle it
				}
			}
		}

		if (xmlPOProcessedFolderPath == null) {
xmlPOProcessedFolderPath = System.getProperty("user.dir") + "/ appData/PO_processed/";
			File filePath = new File(xmlPOProcessedFolderPath);
			if (!filePath.exists())
					filePath.mkdirs();

		}
	}

	private void initializeCount() {

		File[] files = new File(xmlPOFolderPath).listFiles();

		for (File file : files) {
			if (!file.isDirectory() && file.getName().contains(".xml")) {
					String[] f = file.getName().split(".xml");
					int num = Integer.parseInt(f[0].split("_")[1]);
					if (num > count)
						count = num;
			}
		}

		files = new File(xmlPOProcessedFolderPath).listFiles();
		for (File file : files) {
			if (!file.isDirectory() && file.getName().contains(".xml")) {
					String[] f = file.getName().split(".xml");
					int num = Integer.parseInt(f[0].split("_")[1]);
					if (num > count)
						count = num;
```

```
                }
            }
        }

}
```

```
package model;

import java.sql.Connection;

class ItemDAO
{
private static final String DB_URL = "jdbc:derby://localhost:64413/
EECS;user=student;password=secret";
	private static ItemDAO instance = null;

	private ItemDAO() throws Exception
	{
		Class.forName("org.apache.derby.jdbc.ClientDriver").newInstance();
	}

	static synchronized ItemDAO getInstance() throws Exception
	{
		if(instance == null)
			instance = new ItemDAO();
		return instance;
	}

	private class GeneratePreparedStatement
	{
		private Connection conn;
		PreparedStatement ps;

		GeneratePreparedStatement(String query) throws Exception
		{
			conn = DriverManager.getConnection(DB_URL);
			ps = conn.prepareStatement("SET SCHEMA ROUMANI");
			ps.executeUpdate();
			ps.close();
			ps = conn.prepareStatement(query);
		}

		PreparedStatement getPs()
		{
			return ps;
		}

		void closeConnection() throws Exception
		{
			if (!ps.isClosed()) this.ps.close();
			if(!conn.isClosed()) this.conn.close();
		}


```

```
    }
    List<ItemBean> getItems(String catalog, String orderBy)throws Exception
    {
        GeneratePreparedStatement psGenerator = null;
        ResultSet rs = null;
        try
        {
            //building sql
            String sql = "SELECT * FROM ITEM";
            sql = sql + " WHERE CATID=" + Integer.parseInt(catalog);
            sql = orderBy == null? sql : sql + " ORDER BY " + orderBy;
            System.out.println("sql=" + sql);
            psGenerator = new GeneratePreparedStatement(sql);
            PreparedStatement ps = psGenerator.getPs();
            rs = ps.executeQuery();
            List<ItemBean> items = new ArrayList<ItemBean>();
            while(rs.next())
            {
                //ItemBean(String name, String number, double price, int catalog, String unit)
                String name = rs.getString("NAME");
                String number = rs.getString("NUMBER");
                double price = rs.getDouble("PRICE");
                int catalogID = rs.getInt("CATID");
                String unit = rs.getString("UNIT");

                items.add(new ItemBean(name, number, price, catalogID, unit));
            }
            if (items.size() < 1)
                throw new Exception("No Items Found");
            return items;
        }
        catch (Exception e)
        {
            throw e;
        }
        finally
        {
            if(rs != null && !rs.isClosed()) rs.close();
            if(psGenerator != null ) psGenerator.closeConnection();
        }
    }


    ItemBean getItem(String number) throws Exception
    {
```

```
		GeneratePreparedStatement psGenerator = null;
		ResultSet rs = null;
		try
		{
			String sql = "SELECT * FROM ITEM WHERE NUMBER='" + number + "'";

//			ps.setString(1, number);
//			System.out.println(ps.toString());
//			rs =ps.executeQuery();

			psGenerator = new GeneratePreparedStatement(sql);
			PreparedStatement ps = psGenerator.getPs();
			rs = ps.executeQuery();
			if (rs.getRow() < 0)
				throw new Exception ("No item with number " + number);
			ItemBean result = null;
			if(rs.next())
			{
				String name = rs.getString("NAME");
				String itemNumber = rs.getString("NUMBER");
				double price = rs.getDouble("PRICE");
				int catalogID = rs.getInt("CATID");
				String unit = rs.getString("UNIT");

				result = new ItemBean(name, itemNumber, price, catalogID,
	unit);
			}
			return result;
		}
		catch (Exception e)
		{
			throw e;
		}
		finally
		{
			if(rs != null && !rs.isClosed()) rs.close();
			if(psGenerator != null ) psGenerator.closeConnection();
		}
	}

	List<CatalogBean> getCatalogs() throws Exception
	{
		GeneratePreparedStatement psGenerator = null;
		ResultSet rs = null;
		try
		{

			String sql = "SELECT * FROM CATEGORY"; Page
```

```

            psGenerator = new GeneratePreparedStatement(sql);
            PreparedStatement ps = psGenerator.getPs();
            rs =ps.executeQuery();
            List<CatalogBean> result = new ArrayList<CatalogBean>();
            while(rs.next())
            {
                String name = rs.getString("NAME");
                String description = rs.getString("DESCRIPTION");
                int id = rs.getInt("ID");
                result.add(new CatalogBean(name, description, id));
            }
            return result;
        }
        catch (Exception e)
        {
            throw e;
        }
        finally
        {
            if(rs != null && !rs.isClosed()) rs.close();
            if(psGenerator != null ) psGenerator.closeConnection();
        }
    }

}
```

```
package model;

public class ItemBean
{
	private String name;
	private String number;
	private double price;
	private int quantity;
	private double extended;
	private int catalog;
	private String unit;

	public ItemBean() {

	}


public ItemBean(String name, String number, double price, int catalog, String unit, int quantity, 
double extended)
	{
		super();
		this.name = name;
		this.number = number;
		this.price = price;
		this.quantity = quantity;
		this.extended = extended;
		this.catalog = catalog;
		this.unit = unit;
	}

public ItemBean(String name, String number, double price, int catalog, String unit)

	{
		this(name, number, price, catalog, unit, 0, 0.0);
	}

	public String getUnit()
	{
		return unit;
	}

	public void setUnit(String unit)
	{
		this.unit = unit;
	}

	public String getName()
	{
		return name;
	}

	public void setName(String name)
	{
		this.name = name;
	}

	public String getNumber()
	{
		return number;
	}

	public void setNumber(String number)
	{
		this.number = number.toUpperCase();
	}

	public double getPrice()
	{
		return price;
	}

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

	public int getQuantity()
	{
		return quantity;
	}

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

	public double getExtended()
	{
		return extended;
	}

	public void setExtended(double extended)
	{
		this.extended = extended;
	}

	public int getCatalog()
	{
		return catalog;
	}

	public void setCatalog(int catalog)
	{
		this.catalog = catalog;
	}

	@Override
	public String toString()
	{
return "ItemBean [name=" + name + ", number=" + number + ", price=" + price + ", quantity=" + quantity
				+ ", extended=" + extended + ", catalog=" + catalog + ",
unit=" + unit + "]";
	}


}
```

```
package model;

public class CatalogBean
{
	String name;
	String description;
	int id;
	public CatalogBean(String name, String description, int id)
	{
		super();
		this.name = name;
		this.description = description;
		this.id = id;
	}
	public String getName()
	{
		return name;
	}
	public void setName(String name)
	{
		this.name = name;
	}
	public String getDescription()
	{
		return description;
	}
	public void setDescription(String description)
	{
		this.description = description;
	}
	public int getId()
	{
		return id;
	}
	public void setId(int id)
	{
		this.id = id;
	}
	@Override
	public String toString()
	{
return "CatalogBean [name=" + name + ", description=" + description + ", id=" + id + "]";

	}

}
```

```
package model;

import java.util.ArrayList;


@XmlRootElement(name = "product")
public class OrderBean
{
//  CustomerBean customer
	private List<ItemBean> items;
	private double total;
	private double shipping;
	private double HST;
	private double grandTotal;
	private String id;
	private int orderNumber;
	private Date submitted;
	private String account;

	public int getOrderNumber() {
		return orderNumber;
	}
	public void setOrderNumber(int orderNumber) {
		this.orderNumber = orderNumber;
	}
	public String getAccount()
	{
		return account;
	}
	public void setAccount(String account)
	{
		this.account = account;
	}
	public OrderBean(List<ItemBean> items)
	{
		super();
		this.items = items;
	}

	public OrderBean()
	{
		super();
		this.items = null;
	}
	public List<ItemBean> getItems()
	{
		return items;
	}
	public void setItems(List<ItemBean> items)
	{
		this.items = items;
	}
	public double getTotal()
	{
		return total;
	}
	public void setTotal(double total)
	{
		this.total = total;
	}
	public double getShipping()
	{
		return shipping;
	}
	public void setShipping(double shipping)
	{
		this.shipping = shipping;
	}
	public double getHST()
	{
		return HST;
	}
	public void setHST(double hST)
	{
		HST = hST;
	}
	public double getGrandTotal()
	{
		return grandTotal;
	}
	public void setGrandTotal(double grandTotal)
	{
		this.grandTotal = grandTotal;
	}
	public String getId()
	{
		return id;
	}
	public void setId(String id)
	{
		this.id = id;
	}
	public Date getSubmitted()
	{
		return submitted;
	}
	public void setSubmitted(Date submitted)
	{
		this.submitted = submitted;
	}
	@Override
	public String toString()
	{
		return "OrderBean [items=" + items + ", total=" + total + ", shipping=" + shipping + ", HST=" + HST
				+ ", grandTotal=" + grandTotal + ", id=" + id + ", submitted=" + submitted + "]";
	}

	public boolean isEmpty() {
		if (items == null)
			return true;

		return items.size() == 0;
	}
	public void clear() {

		items = new ArrayList<>();
		total = 0;
		shipping = 0;
		HST = 0;
		grandTotal = 0;
		submitted = null;

	}



}
```

```
<?xml version="1.0" encoding="UTF-8" ?>
<jsp:root xmlns:jsp="http://java.sun.com/JSP/Page" version="2.0"
                xmlns:c="http://java.sun.com/jsp/jstl/core"
                xmlns:fmt="http://java.sun.com/jsp/jstl/fmt">
<jsp:directive.page contentType="text/html; charset=UTF-8"
        pageEncoding="UTF-8" session="false"/>
<jsp:output doctype-root-element="html"
        doctype-public="-//W3C//DTD XHTML 1.0 Transitional//EN"
        doctype-system="http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd"
        omit-xml-declaration="true" />
<html xmlns="http://www.w3.org/1999/xhtml" lang="en">
<head >
        <title>Foods R US</title>
</head>
<body>
<c:if test="${not empty order}">

                <div>
                                <h1>Details of Order:</h1>

                                Account Name: ${order.account}<br/>
                                Order number: ${order.orderNumber}<br/>
                                Order date: ${order.submitted}<br/>

                </div>
                <div>
                        <h2>Your order items:</h2>
                        <table border="1">
                            <tbody>
                                <tr>
                                    <td>Product ID</td>
                                    <td>Name</td>
                                    <td>Price</td>
                                    <td>Quantity</td>
                                </tr>

                                <c:forEach var="e" items="${order.items}">
                                    <tr>
                                        <td>${e.number}</td>
                                        <td>${e.name}</td>
                                        <td>$${e.price}</td>
                                        <td>${e.quantity}</td>
                                    </tr>
                                </c:forEach>

                            </tbody>
                        </table>
                </div>

                <div>
                        <table>
                            <tbody>
                                <tr>
                                    <td></td>
                                    <td>Sub-Total:</td>
                                    <td>$${order.total}</td>
                                </tr>
                                <tr>
                                    <td></td>
                                    <td>HST:</td>
                                    <td>$${order.HST}</td>
                                </tr>
                                <tr>
                                    <td></td>
                                    <td>Shipping:</td>
                                    <td>$${order.shipping}</td>
                                </tr>
                                <tr>
                                    <td></td>
                                    <td>Grand Total:</td>
                                    <td>$${order.grandTotal}</td>
                                </tr>
                            </tbody>
                        </table>
                 </div>


            <div>

                    <button onclick="javascript:window.print()">Print Window</button>

            </div>

</c:if>

<c:if test="${empty order}">
        <jsp:include page="nav.jspx"/>
        <h1>New Orders (Not Processed)</h1>
</c:if>

<c:if test="${not empty fileNames}">
                <c:forEach var="e" items="${fileNames}">
                        <a target="_blank" href="Account?view=${e}">${e}</a><br/>
                </c:forEach>
</c:if>

```

```
<h1>Old Orders (Processed)</h1>
<c:if test="${not empty processedfileNames}">
                <c:forEach var="e" items="${processedfileNames}">
                        <a target="_blank" href="Account?view=${e}">${e}</a><br/>
                </c:forEach>
</c:if>

<c:if test="${not empty error}">
        <h1 style="color:red;">${error}</h1>
</c:if>

</body>
</html>
</jsp:root>
```

Add.jspx

```
<?xml version="1.0" encoding="UTF-8" ?>
<jsp:root xmlns:jsp="http://java.sun.com/JSP/Page" version="2.0"
          xmlns:c="http://java.sun.com/jsp/jstl/core"
          xmlns:fmt="http://java.sun.com/jsp/jstl/fmt">
<jsp:directive.page contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8" session="false"/>
<jsp:output doctype-root-element="html"
    doctype-public="-//W3C//DTD XHTML 1.0 Transitional//EN"
    doctype-system="http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd"
    omit-xml-declaration="true" />
<html xmlns="http://www.w3.org/1999/xhtml" lang="en">
<head >
    <meta charset="utf-8" />
    <meta content="width=device-width, initial-scale=1" name="viewport" />
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css" />
<link href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/ bootstrap.min.css" rel="stylesheet" />
    <link href="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js" rel="stylesheet" />

    <title>Foods R US</title>
</head>
<body>

    <jsp:include page="nav.jspx"/>
    <h2>Shopping Cart</h2>

    <form action="Add" method="POST">
        <table class="table">
            <thead><tr>
                <th>Product ID</th>
                <th>Name</th>
                <th>Unit Price</th>
                <th>Quantity</th>
                <th>Unit</th>
                <th>Total</th>
                <th>Remove Item</th>
            </tr></thead>

            <c:forEach var="e" items="${order.items }">
                <tr>
                    <td>${e.number }</td>
                    <td>${e.name }</td>
                    <td><fmt:formatNumber type="currency" value="${e.price }"/></td>
                    <td><input type="number" step="1" name="${e.number}" value="${e.quantity}"/></td>
```

```
                        <td>${e.unit }</td>
                        <td><fmt:formatNumber type="currency" value="${e.quantity * e.price }"/></td>
                        <td><input type="checkbox" name="delete_${e.number}" value="${e.number}"/> Delete</td>
                    </tr>
                </c:forEach>
            </table>
<div><input type="submit" class="btn btn-info" name="update" value="update" /></div>
            <br/>
<div><input type="button" class="btn btn-info" value="Continue Shopping" onclick="location='Browse'"/></div>
            <br/>
        </form>

        <form action="Checkout" method="POST">
            <input type="hidden" id="force" name="force" value="true"/>
<input type="submit" class="btn btn-info" name="checkout" value="${checkout}" />
        </form>

        <div id="pricing">
            <div>Total: <fmt:formatNumber type="currency" value="${order.total }"/></div>
            <div>HST: <fmt:formatNumber type="currency" value="${order.HST }"/></div>
            <div>Shipping: <fmt:formatNumber type="currency" value="${order.shipping }"/></div>
<div>Grand Total: <fmt:formatNumber type="currency" value="${order.grandTotal }"/></div>
        </div>


        <div id="advertisement"><c:if test="${not empty advertise }">
            <h4>You might want to get ${advertise.name }</h4>
            <div>Item No.: ${advertise.number }</div>
            <div>${advertise.name }</div>
<div><fmt:formatNumber type="currency" value="${advertise.price }"/> / ${advertise.unit } </div>
            <form action="Add" method="POST">
                <div><label for="qty">Qty: </label>
                    <input type="number" name="qty" value="1" step="1"/></div>
                    <input type="hidden" name="number" value="${advertise.number }"/>
                <div><input type="submit" name="select_item" value="Add To Cart"/></div>
            </form>
        </c:if></div>

        <c:if test="${not empty error}">
            <div class="alert alert-danger">${error }</div>
        </c:if>



    </body>

</html>
</jsp:root>
```

Analytics.jspx

```
<?xml version="1.0" encoding="UTF-8" ?>
<jsp:root xmlns:jsp="http://java.sun.com/JSP/Page" version="2.0"
                        xmlns:c="http://java.sun.com/jsp/jstl/core"
                        xmlns:fmt="http://java.sun.com/jsp/jstl/fmt">
<jsp:directive.page contentType="text/html; charset=UTF-8"
        pageEncoding="UTF-8" session="false"/>
<jsp:output doctype-root-element="html"
        doctype-public="-//W3C//DTD XHTML 1.0 Transitional//EN"
        doctype-system="http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd"
        omit-xml-declaration="true" />
<html xmlns="http://www.w3.org/1999/xhtml" lang="en">
<head >
        <meta content="width=device-width, initial-scale=1" name="viewport" />
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css" />
<link href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/ bootstrap.min.css" rel="stylesheet" />
<link href="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/ jquery.min.js" rel="stylesheet" />
        <title>Foods R US</title>
</head>
<body><div class="container">

        <div class="panel-heading"><h2>Analytics Report</h2></div>

          <div class="panel-group">
                <div class="panel panel-info">
                        <div class="panel-heading">Average Time Users Add Items to Cart: </div>
                        <div class="panel-body">
                                <c:choose>
                                        <c:when test="${ not empty rpt_AddItem}">${rpt_AddItem }</c:when>
                                        <c:otherwise>
                                                <fmt:formatNumber maxFractionDigits="3" value="${ aveAddItem}" /> seconds
                                        </c:otherwise>
                                </c:choose>
                        </div>
                </div>

                <div class="panel panel-info">
                        <div class="panel-heading">Average Time Users Checkout: </div>
                        <div class="panel-body">
                                <c:choose>
                                        <c:when test="${ not empty rpt_Checkout}">${rpt_Checkout }</c:when>
                                        <c:otherwise>
```

```
                                                <fmt:formatNumber maxFractionDigits="3"
            value="${ aveCheckout}" /> seconds
                                        </c:otherwise>
                                </c:choose>
                        </div>
                </div>
        </div>

</div></body>
</html>
</jsp:root>
```

Browse.jspx

```
<?xml version="1.0" encoding="UTF-8" ?>
<jsp:root xmlns:jsp="http://java.sun.com/JSP/Page" version="2.0"
        xmlns:c="http://java.sun.com/jsp/jstl/core"
        xmlns:fmt="http://java.sun.com/jsp/jstl/fmt">
        <jsp:directive.page contentType="text/html; charset=UTF-8"
                pageEncoding="UTF-8" session="false" />
        <jsp:output doctype-root-element="html"
                doctype-public="-//W3C//DTD XHTML 1.0 Transitional//EN"
doctype-system="http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd"
                omit-xml-declaration="true" />
        <html xmlns="http://www.w3.org/1999/xhtml" lang="en">
<head>
<meta charset="utf-8"></meta>
<meta name="viewport" content="width=device-width, initial-scale=1"></meta>
<link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css"></link>
<link rel="stylesheet"
        href="https://use.fontawesome.com/releases/v5.5.0/css/all.css"
integrity="sha384-B4dIYHKNBt8Bc12p+WXckhzcICo0wtJAoU8YZTY5qE0Id1GSseTk6S +L3BlXeVIU"
        crossorigin="anonymous"></link>
<link rel="stylesheet" href="res/Style.css"></link>
<title>Foods R US</title>
</head>
<body>

        <jsp:include page="nav.jspx"/>

        <div class="container-fluid padding">
                <div class="row welcome text-center">
                        <div class="col-12">
                                <a href="Browse"><button type="button" class="btn btn-info btn-lg">Browse Categories</button></a>
                                <br></br>
                                <hr class="my-3"></hr>
                        </div>

                </div>

                <div class="container-fluid padding">
                        <div class="row text-center padding">
                                                <c:forEach var="e" items="${catalogs }">
                                                        <div class="col-xs-12 col-sm-6 col-md-3">
                                                        <p class="category-image"
                                                                style="background-image: url('${pageContext.request.contextPath}/res/images/${e.name}.jpg');">
```

```
                                        </p>
                                        <h3>
                                                <a href='Browse?cat=${e.id} '>${e.name }</a>
                                        </h3>
                                        <p>${e.description }</p>
                                </div>
                        </c:forEach>
                </div>
        </div>
        <hr class="my-1"></hr>
</div>

<c:if test="${not empty items }">
        <div class="container-fluid padding" id="items">
                <br></br>
                <h2 class="text-center">Browse Items Listing</h2>
                <form action="Browse" method="POST">
                        <input type="hidden" name="cat" value="${cat }" />
                        <input type="hidden" name="number" value="${number }" />
                        Sort By: <select name="orderBy" class="form-control">
                                <option selected="any" value="NONE">Select One:</option>
                                <option value="PRICE">Price</option>
                                <option value="NAME">Name</option>
                        </select>
                        <input type="submit" name="sort" value="Sort" />
                </form>
                <br></br>
                <hr></hr>
                <div class="row">
                        <c:forEach var="e" items="${items }">
                                <div class="col-sm-4 col-md-3">
                                        <div class="pannel panel-primary item col-md-10 col-
sm-6 col-xs-12">
                                                <div class="col-md-10 col-sm-10 col-xs-10">
                                                        <div class="panel-heading item-name">
${e.name }</div>
                                                        <div class="panel-body item_no">Item No.:
${e.number }</div>
                                                        <div>
                                                                <fmt:formatNumber type="currency"
value="${e.price }" />
                                                                / ${e.unit }
                                                        </div>
                                                        <form action="Add" method="POST">
                                                                <div>
                                                                        <label for="qty">Qty: </label> <input
type="number" name="qty"
                                                                                value="1" step="1" />
                                                                </div>
                                                                <input type="hidden" name="number"
value="${e.number }" />
                                                                <div>
                                                                        <button type="submit"
name="select_item" value="Add To Cart" class="btn btn-success btn-md">Add To
Cart</button>
                                                                </div>
                                                        </form>
                                                </div>
                                        </div>
                                </div>
                        </c:forEach>

                </div>
        </div>
</c:if>

<div id="error">
        <strong>${error }</strong>
</div>



</body>
</html>
</jsp:root>
```

```
<?xml version="1.0" encoding="UTF-8" ?>
<jsp:root xmlns:jsp="http://java.sun.com/JSP/Page" version="2.0"
                    xmlns:c="http://java.sun.com/jsp/jstl/core"
                    xmlns:fmt="http://java.sun.com/jsp/jstl/fmt">
<jsp:directive.page contentType="text/html; charset=UTF-8"
        pageEncoding="UTF-8" session="false"/>
<jsp:output doctype-root-element="html"
        doctype-public="-//W3C//DTD XHTML 1.0 Transitional//EN"
        doctype-system="http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd"
        omit-xml-declaration="true" />
<html xmlns="http://www.w3.org/1999/xhtml" lang="en">
<head >
        <title>Foods R US</title>
</head>
<body>
<jsp:include page="nav.jspx"/>
<c:if test="${not empty order}">
    <h2>${order.account}, your order has been processed</h2>
  </c:if>

  <h1>Not Processed files</h1>
<c:if test="${not empty fileNames}">
                <c:forEach var="e" items="${fileNames}">
                        <a target="_blank" href="Account?view=${e}">${e}</a><br/>
                </c:forEach>
</c:if>

<h1>Processed Files</h1>
<c:if test="${not empty processedfileNames}">
                <h1>Old Orders (Processed)</h1>
                <c:forEach var="e" items="${processedfileNames}">
                        <a target="_blank" href="Account?view=${e}">${e}</a><br/>
                </c:forEach>
</c:if>
</body>
</html>
</jsp:root>
```

```
<?xml version="1.0" encoding="UTF-8" ?>
<jsp:root xmlns:jsp="http://java.sun.com/JSP/Page" version="2.0"
                    xmlns:c="http://java.sun.com/jsp/jstl/core"
                    xmlns:fmt="http://java.sun.com/jsp/jstl/fmt">
<jsp:directive.page contentType="text/html; charset=UTF-8"
        pageEncoding="UTF-8" session="false"/>
<jsp:output doctype-root-element="html"
        doctype-public="-//W3C//DTD XHTML 1.0 Transitional//EN"
        doctype-system="http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd"
        omit-xml-declaration="true" />
<html xmlns="http://www.w3.org/1999/xhtml" lang="en">
<head >
        <title>Foods R US</title>
</head>
<body>
        <jsp:include page="nav.jspx"/>

        <div>${error }</div>
        <c:if test="${not empty user }">
                <div>${name }, You have logged in</div>
                <div><a href="Browse">Continue Shopping</a></div>
                <div><a href="Add">View Cart and to Checkout</a></div>
        </c:if>
</body>
</html>
</jsp:root>
```

nav.jspx

```
<?xml version="1.0" encoding="UTF-8" ?>
<jsp:root xmlns:jsp="http://java.sun.com/JSP/Page" version="2.0"
    xmlns:c="http://java.sun.com/jsp/jstl/core"
    xmlns:fmt="http://java.sun.com/jsp/jstl/fmt">
    <jsp:directive.page contentType="text/html; charset=UTF-8"
        pageEncoding="UTF-8" session="false" />
    <jsp:output doctype-root-element="html"
        doctype-public="-//W3C//DTD XHTML 1.0 Transitional//EN"
doctype-system="http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd"
        omit-xml-declaration="true" />
    <html xmlns="http://www.w3.org/1999/xhtml" lang="en">
<head>
<meta charset="utf-8"></meta>
<meta name="viewport" content="width=device-width, initial-scale=1"></meta>
<link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css"></link>
<link rel="stylesheet"
    href="https://use.fontawesome.com/releases/v5.5.0/css/all.css"
integrity="sha384-B4dIYHKNBt8Bc12p+WXckhzcICo0wtJAoU8YZTY5qE0Id1GSseTk6S +L3BlXeVIU"
    crossorigin="anonymous"></link>
<link rel="stylesheet" href="res/Style.css"></link>
<title>Foods R US</title>
</head>
<body>
    <nav class="navbar navbar-expand-sm bg-dark navbar-dark sticky-top">
        <div class="container-fluid">
            <div class="navbar-header">
                <a class="navbar-brand" href="Browse">Foods R Us</a>
            </div>

            <button class="navbar-toggler" type="button" data-toggle="collapse"
                data-target="#navbarResponsive">
                <span class="navbar-toggler-icon"></span>
            </button>
            <div class="collapse navbar-collapse" id="navbarResponsive">

                <ul class="nav navbar-nav ml-auto">
                    <li>
                        <form class="form-inline" action="Browse" method="post">
                            <input class="form-control mr-sm-2" type="string"
                                placeholder="Enter the Item No." name="number" />
                            <button class="btn btn-success" type="submit"
```

```
name="select_item"
                                        value="search">Search</button>
                                </form>
                        </li>

                        <li><a class="nav-link" href="Add">Cart</a></li>
                        <li><a class="nav-link" href="Account">Account</a></li>
                        <c:if test="${empty user}">
                                <li><a class="nav-link" href="Login?login=true">Log
In</a></li>
                        </c:if>
                </ul>
            </div>
        </div>
    </nav>

    <div class="container-fluid">
        <div class="row jumbotron">
            <div class="col-xs-12 col-sm-12 col-md-9 col-lg-9 col-xl-10">
                <h1>Fast, Reliable, Convenient</h1>
                <p class="lead">Foods R Us provides a wide variety of gourmet
                        grocery items at great prices. Our products are all of the
highest
                        quality. Avoid wasting time at your local grocery store
and simply
                        order your groceries online and have them delivered right
to your
                        door!
                </p>
                <br></br>
            <br></br>
            </div>
        </div>
    </div>

</body>
    </html>
</jsp:root>
```

```
@CHARSET "UTF-8";
@import url('https://fonts.googleapis.com/css?family=Lato|Pacifico|Righteous| Roboto+Condensed|
Ubuntu|Muli');


html,
body {
	color: #212121;
	background-color: white;
	margin: 0px 0px;
	padding: 0px;
	font-family: 'Lato', 'sans-serif';
}


.category-image{

	height:200px;
	width:300px;
	margin-left:auto;
	margin-right:auto;"
}

.form-row{
	padding: 10px 0;
	display: flex;
}

.form-row label{
	padding-right: 10px;
}

.form-row input{
	flex: 1px;
}

.btn-round {
	margin-left: 5%;
	width: 90%;
	padding-left: 5%;
	padding-right: 5%;
	background: red;
	color: white;
	border: 1px solid white;
	font-weight: bolder;
	font-size: 20px;
	border-radius: 24px;
}
```

```

.btn-round:hover {
	border: 1px solid red;
}

.navbar.navbar-default {
	margin-bottom: 0px !important;
	background-color: black;
	font-weight: 700;
}

.navbar-brand{
	font-family: 'Pacifico';
	font-size: 35px;

}

.jumbotron{

	background-image: url('images/bg.jpg');
	background-size: cover;
	color:white;
}


.logo
{
	height:100px;
}

.category-banner {
	height: 20%;
	width: 100%;
	padding: 10% 0% 10% 5%;
	color: white;
	background: #3F51B5;
	background-repeat: no-repeat;
	background-size: 100%;
	background-position: center;
}

.category-banner h1 {
	font-size: 48px;
	font-weight: bolder;
	text-transform: uppercase;
}

.category {
```

```css
	padding: 16px;
}

.category-name {
	font-weight: bold;
	margin-bottom: 8px;
}

.category-description {
	font-weight: light;
}

.filter {
	background: white;
	border: 1px solid #607D8B;
	border-top: none;
	padding: 16px;
	margin-bottom: 16px;
}

.item {
	margin-top: 4px;
	margin-bottom: 4px;
	border: 1px solid #ddd;
	background: white;
	padding: 16px;
	padding-right: 0px;
	transition: background ease-in-out 250ms;
}

.item:hover {
	background: #eee;
}

.item img {
	margin-top: 12px;
	width: 32px;
	height: 32px;
	border-radius: 50%;
}

.item .item-name {
	font-weight: bolder;
	width: 118%;
}

.item .item-number {
	font-size: 11px;
```

```
    font-weight: lighter;
    color: gray;
}

.cart-btn {
    margin-left: 105% !important;
}

.cart {
    margin-top: 8px;
    color: gray;
    font-size: 24px;
}

#cart-count {
    transition: all ease-in-out 500ms;
}

#cart-count.cart-count-glow {
    background: #3F51B5;
    padding: 8px;
    font-size: 18px;
}

.cart-item-name {
    font-weight: bolder;
    font-size: 20px;
}

.cart-item-number {
    font-size: 14px;
    font-weight: lighter;
    color: gray;
    margin-bottom: 8px;
}

.cart-item-price {
    font-size: 18px;
}

.cart-item-amount {
    width: 32px;
}
```