

Foods R Us

An E-Commerce Website

Designed and Developed By

Ameen (cse: aljailaa)

Henry (cse:)

Savan (cse:savan414)

*Submitted as the group project for the course
EECS4413 - Building E-Commerce Systems Prof.*

H. Roumani - Fall 2018

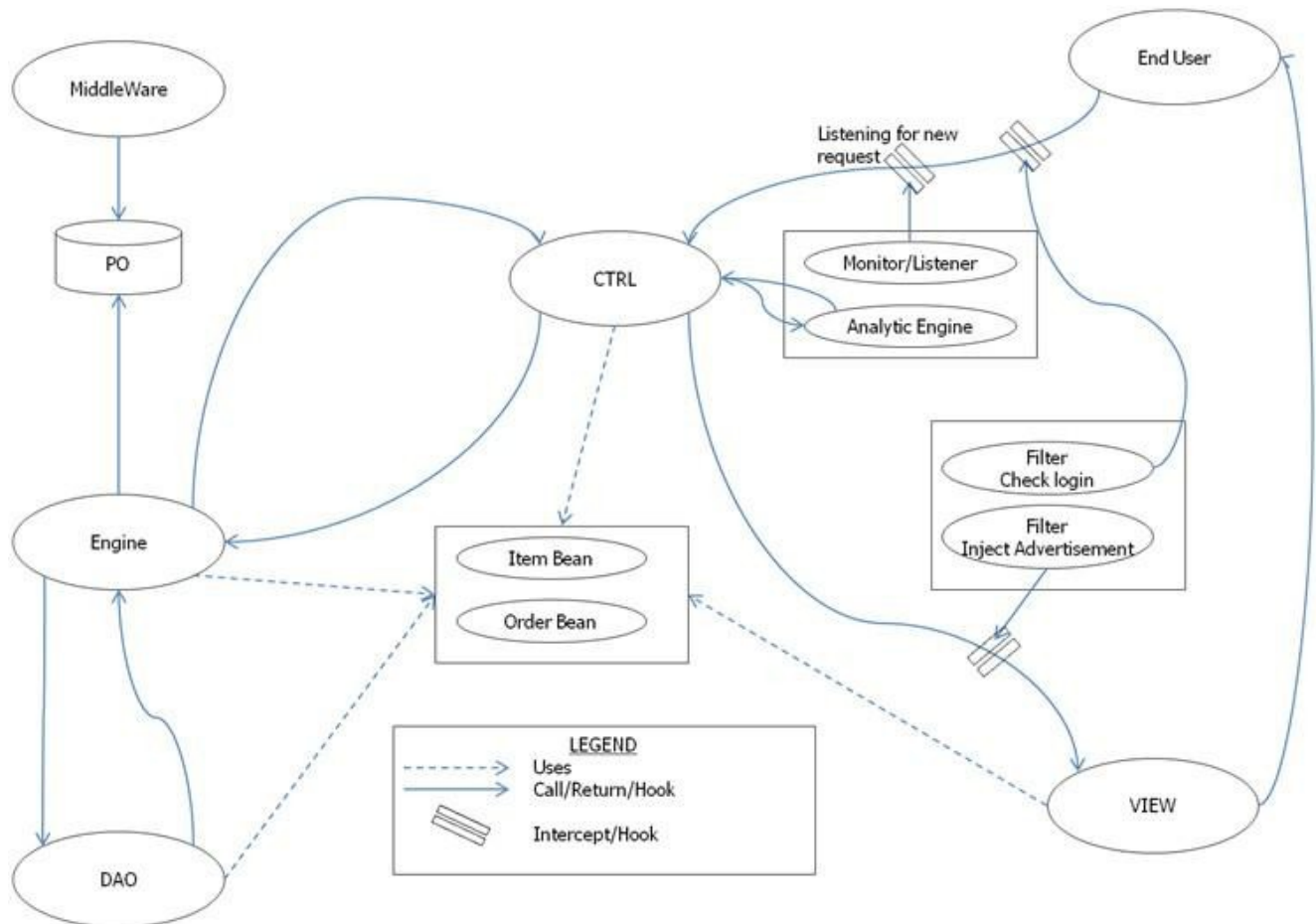
Table of Contents

Foods R Us.....	1
An E-Commerce Website.....	1
Diagram:.....	4
Implementation.....	5
Retrieving the Catalog information.....	5
Allowing End-user to view list of items by Category Option.....	5
Role of the Browse controller.....	5
Role of ItemDAO.....	5
Role of Engine for the Browsing Service.....	5
Implementing the Service of Maintaining Shopping Cart.....	6
Issues to consider when adding item:.....	6
Issues to consider when updating cart/order:.....	6
Implementing the Service of Login.....	7
Implementing the Service of Analytics.....	7
Implementing the filter for Ad Hoc Advertisement.....	7
Implementing the Middleware.....	7
A client checks out.....	8
A client visits the URL of a P/O.....	8
The frontend Implementation.....	8
The Team.....	9
Work division.....	9
Meetings.....	9
Lesson learned.....	9
Output samples:.....	10
Page Upon Entering (Browse Main Page).....	10
Listing of items, with the option to sort available.....	10
Prior to Log in, the Log in link will be provided.....	11

<u>The Login Redirected view.....</u>	<u>11</u>
<u>After the Login, the login link will not appear.....</u>	<u>11</u>
<u>The Cart View.....</u>	<u>12</u>
<u>The view after a checkout was successful.....</u>	<u>13</u>
<u>The Appearance of advertised item when End-user add into cart item number 1409S4.....</u>	<u>14</u>
<u>Purchase order view.....</u>	<u>15</u>
<u>The Page for Admin to view Analytics Report.....</u>	<u>15</u>
<u>Source Code.....</u>	<u>16</u>
<u>Tree.....</u>	<u>16</u>

Design

Diagram:



Overview

From the diagram above we can conclude the following:

- o 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.
- o When a user makes a purchase order, an XML file is generated and stored in folder structure denoted above as PO
- o 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
- o The application uses the MVC design pattern.
- o 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:

- o main page' or 'fresh request' to view category list
- o viewing items of a category
- o viewing item with identified item number
- o 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:

- o List of items (ItemBean) base on the category id requested
- o List of all available categories (CatalogBean)
- o 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:

- o List of available Catalogues
- o List of Items to display based on requested catalogue id.
- o 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.

- o First we need to identify if it is a request to delete a row item.
- o 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:

- o Create a toDelete List, and check if there is item(s) to be deleted. We make the Engine accept zero length list.
- o 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](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 jsp 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

Bootstrap was used to make the the code pretty.

The Team

Work division

Work amongst team members was divided as follows:

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

Member's Contributions

- **Henry:**
Worked on the backend for browse, cart, analytics, and advertisement. Contributed in fixing many bugs as they arise
- **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

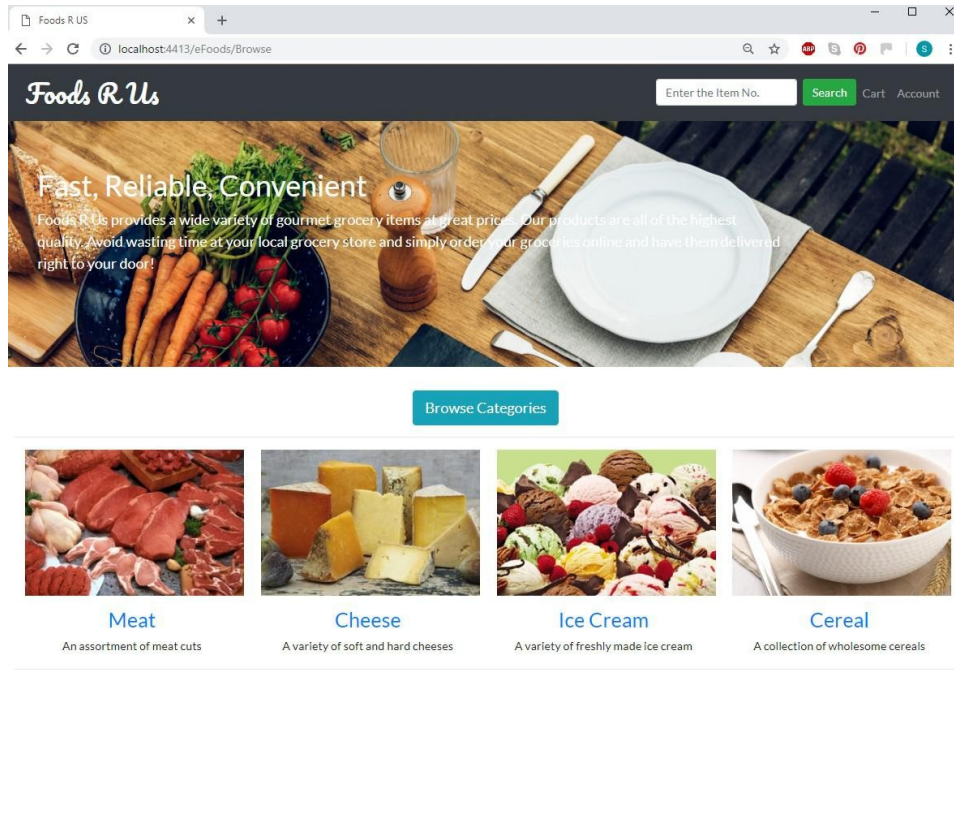
In the second meeting, we sit together briefly to have a plan on making the report. All team members were accessible through Email

Lesson learned

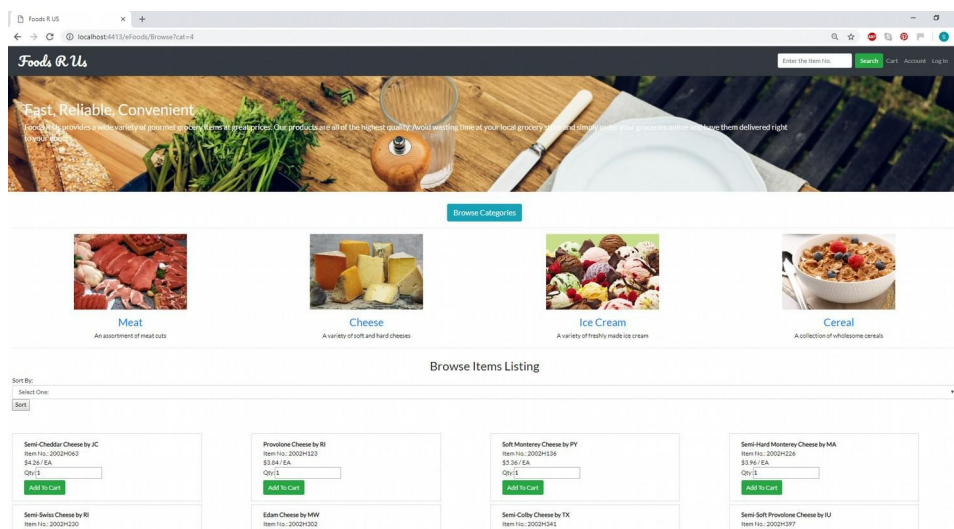
One of the lessons we learned is the importance of using version control when working in a project involving multiple team members. Unfortunately, we were not successful in utilizing this tool and as a result a lot of productive time was wasted merging fixes and keeping track of who did what

Output samples:

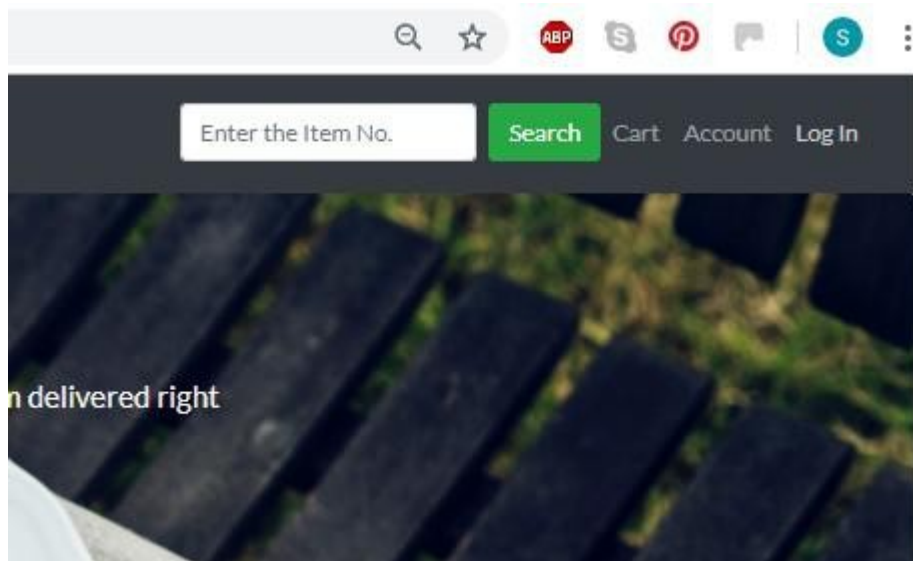
Page Upon Entering (Browse Main Page)



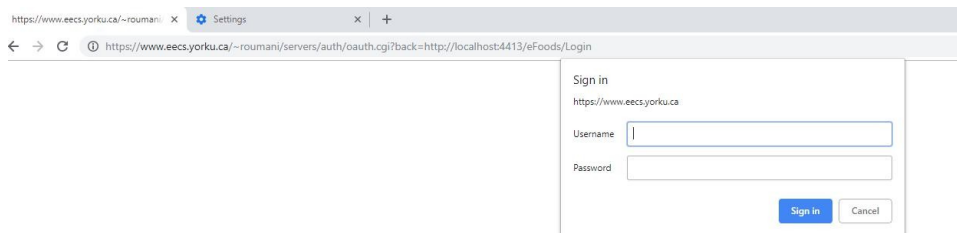
Listing of items, with the option to sort available



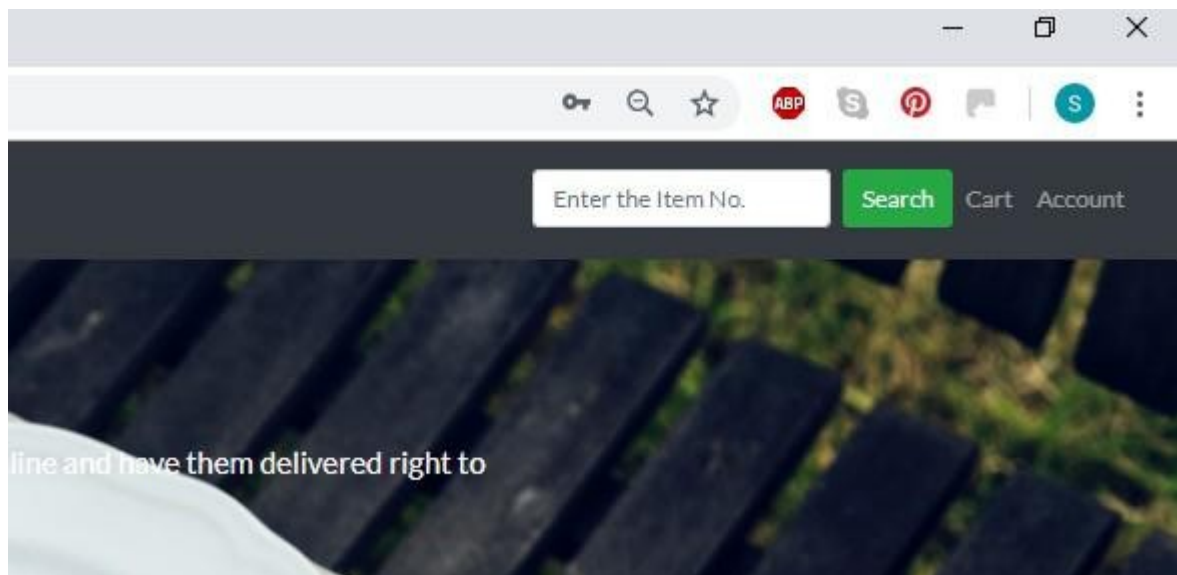
Prior to Log in, the Log in link will be provided



The Login Redirected view



After the Login, the login link will not appear.



The Cart View

Foods R Us

Enter the Item No.

Search

Cart

Account

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!

Shopping Cart

Product ID	Name	Unit Price	Quantity	Unit	Total	Remove Item
0905A044	Minced Rib Meat by VX	\$6.49	<div>1</div>	EA	\$6.49	<div><div>Delete</div></div>

update

Continue Shopping

confirm

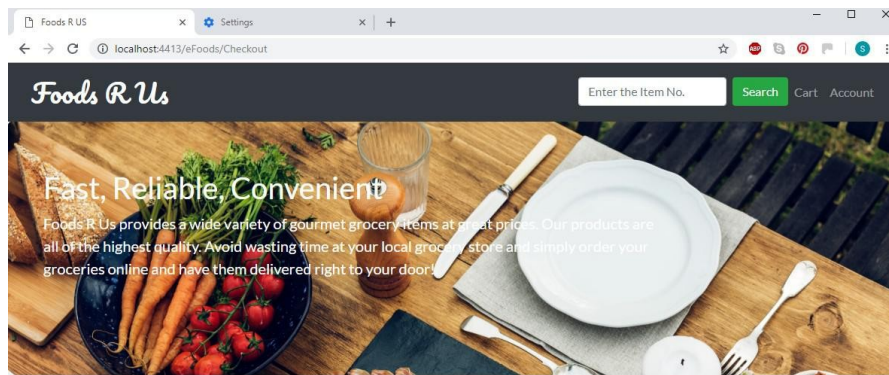
Total: \$6.49

HST: \$0.84

Shipping: \$5.00

Grand Total: \$12.33

The view after a checkout was successful



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

Enter the Item No.

[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	<input type="text" value="1"/>	EA	\$6.44	<input type="checkbox"/> Delete

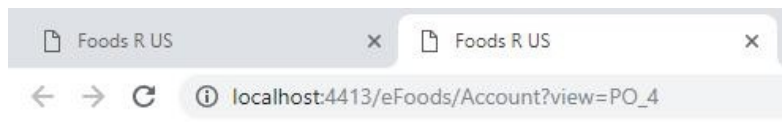
Total: \$6.44
HST: \$0.84
Shipping: \$5.00
Grand Total: \$12.28

You might want to get Semi-Monterey Cheese by GK

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

Qty:

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:

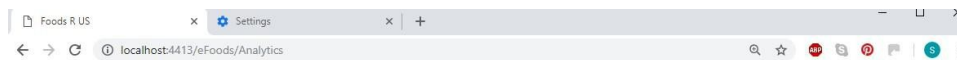
Product ID	Name	Price	Quantity
2002H063	Semi-Cheddar Cheese by JC	\$4.26	1
0905A044	Minced Rib Meat by VX	\$6.49	1

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

[Print Window](#)

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

Advertise.java

```
1 package adhoc;
2
3 import java.io.IOException;
4
5
6
7
8
9
10
11
12
13
14
15 /**
16  * Servlet Filter implementation class Advertise
17  */
18 @WebFilter(dispatcherTypes = {
19     // DispatcherType.REQUEST,
20     DispatcherType.FORWARD
21 })
22     , urlPatterns = { "/Advertise", "/Add.jspx" }, servletNames = { "Add" }
23 )
24 public class Advertise implements Filter
25 {
26     String number_hook_item = "1409S413";
27     String number_advertise_item = "2002H712";
28     public Advertise()
29     {
30     }
31     public void destroy()
32     {
33     }
34
35     public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain) throws
36         IOException, ServletException
37     {
38
39         //check if it is item 1409S413
40         HttpServletRequest req = (HttpServletRequest) request;
41         String select_item = req.getParameter("select_item");
42         if(select_item != null && select_item.equals("Add To Cart") &&
43             req.getParameter("number") != null && req.getParameter
44             ("number").equals(number_hook_item))
45         {
46             try
47             {
48                 Engine engine = Engine.getInstance();
49                 request.setAttribute("advertise", engine.getItem
50                 (number_advertise_item));
51                 request.setAttribute("toadvertise", "true");
52             }
53             catch (Exception e)
54             {
55                 System.out.println("Failed to hook advertisement");
56             }
57         }
58     }
59 }
```

Advertise.java

```
55     }
56     chain.doFilter(request, response);
57 }
58
59
60 public void init(FilterConfig fConfig) throws ServletException {
61 }
62
63 }
64
```

Checkout.java

```
1 package adhoc;
2
3 import java.io.IOException;
13
14 /**
15  * Servlet Filter implementation class Checkout
16  */
17 @WebFilter("/Checkout","/Account")
18 public class Checkout implements Filter {
19
20     public Checkout()
21     {
22     }
23
24     public void destroy()
25     {
26     }
27
28     public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain) throws
        IOException, ServletException
29     {
30         HttpServletRequest req = (HttpServletRequest) request;
31         if (req.getSession().getAttribute("user")!=null)
32         {
33             chain.doFilter(request, response);
34         }
35         else
36         {
37             ((HttpServletResponse)response).sendRedirect("Login?login=true");
38         }
39     }
40
41     public void init(FilterConfig fConfig) throws ServletException
42     {
43     }
44
45 }
```

Logging.java

```
1 package adhoc;
2
3 import java.io.IOException;
14
15 /**
16  * Servlet Filter implementation class Logging
17  */
18 @WebFilter({ "/Logging", "/" })
19 public class Logging implements Filter
20 {
21
22     public Logging()
23     {
24     }
25
26     public void destroy()
27     {
28     }
29
30     public void doFilter(ServletRequest request, ServletResponse response,
31         FilterChain chain) throws IOException, ServletException
32     {
33         CtrlUtil.log_request(System.out, (HttpServletRequest) request,
34             "Incoming");
35         chain.doFilter(request, response);
36         CtrlUtil.log_request(System.out, (HttpServletRequest) request,
37             "Outgoing");
38     }
39
40     public void init(FilterConfig fConfig) throws ServletException
41     {
42     }
43 }
```

AnalyticEngine.java

```
1 package analytics;
2
3 import java.util.Map;
4
5 public class AnalyticEngine
6 {
7     //list of attribute name
8     public static final String ORDER = "order";
9     public static final String CLIENTS = "clients";
10
11     private static AnalyticEngine instance = null;
12
13     private AnalyticEngine()
14     {
15     }
16
17     public static AnalyticEngine getInstance()
18     {
19         if (instance == null)
20             instance = new AnalyticEngine();
21         return instance;
22     }
23
24     public double computeAveAddItemInterval(Map<String, UserActivity> users)
25     {
26         long totalTime = 0;
27         int numberOfIntervals = 0;
28         for(UserActivity user :users.values())
29         {
30             long last = user.startTime;
31             int user_c = user.counter;
32             if (user_c > 0)
33             {
34                 long[] timeLog = user.addItemTime;
35                 for(int i = 0; i < user_c; i++)
36                 {
37                     totalTime = totalTime + timeLog[i] - last;
38                     last = timeLog[i];
39                     numberOfIntervals ++;
40                 }
41             }
42         }
43         if (numberOfIntervals < 1 ) //in case there is no one has entered yet.
44             return -1;
45         double ave = (double)totalTime/1000/numberOfIntervals;
46         return ave;
47     }
48 }
```

```
49 public double computeCheckOutInterval(Map<String, UserActivity> users)
50 {
51     long totalTime = 0;
52     int numberOfUsers = 0;
53     for(UserActivity user: users.values())
54     {
55         long first = user.startTime;
56         long checkOut = user.checkoutTime;
57         if(checkOut > 0 )
58         {
59             totalTime = totalTime + checkOut - first;
60             numberOfUsers ++;
61         }
62     }
63     if (numberOfUsers < 1 ) //in case there is no one has entered yet.
64         return -1;
65     return (double)totalTime / numberOfUsers;
66 }
67
68
69
70
71 }
```

Monitor.java

```
1 package analytics;
2
3 import java.util.ArrayList;
19
20 /**
21  * Application Lifecycle Listener implementation class Monitor
22  *
23  */
24 @WebListener
25 public class Monitor implements
26     HttpSessionListener, ServletRequestListener
27 {
28     public Monitor()
29     {
30     }
31
32     public void sessionCreated(HttpSessionEvent se)
33     {
34         attachOrder (se);
35         addUser(se);
36     }
37     //attach Order Object for the new session(new user)
38     private void attachOrder(HttpSessionEvent se)
39     {
40         //attach order (shopping cart) object to Session context
41         HttpSession session = se.getSession();
42         List<ItemBean> items = new ArrayList<ItemBean>();
43         OrderBean order = new OrderBean(items);
44         session.setAttribute(AnalyticEngine.ORDER, order);
45     }
46     //Add the new session/user to the list of 'userActivity' stored in
47     //Servlet context
48     private void addUser(HttpSessionEvent se)
49     {
50         //Add client to list of "clients" in ServletContext to monitor all
51         //users activity
52         ServletContext servlet = se.getSession().getServletContext();
53         if (servlet.getAttribute(AnalyticEngine.CLIENTS) == null)
54         {
55             servlet.setAttribute(AnalyticEngine.CLIENTS, new HashMap<String,
56                 UserActivity>());
57         }
58         @SuppressWarnings("unchecked")
59         Map<String, UserActivity> clients = (Map<String,
60             UserActivity>)servlet.getAttribute("clients");
61         clients.put(se.getSession().getId(),
62             new UserActivity(se.getSession().getId(),
```



```

        System.currentTimeMillis());
60     }
61
62
63     public void requestInitialized(ServletRequestEvent sre)
64     {
65         //check if it is addItem Request
66         HttpServletRequest request = (HttpServletRequest) sre.getServletRequest();
67         if(request.getParameter("select_item")!= null)
68             logAddItem(sre);
69     }
70
71     //to log the adding of item by a user;
72     private void logAddItem(ServletRequestEvent sre)
73     {
74         ServletContext servlet = sre.getServletContext();
75         String id = ((HttpServletRequest) sre.getServletRequest()).getSession().getId();
76
77         @SuppressWarnings("unchecked")
78         Map<String, UserActivity> users =
79             (Map<String, UserActivity>)servlet.getAttribute
80             (AnalyticEngine.CLIENTS);
81         UserActivity user = users.get(id);
82         user.logAddItem(System.currentTimeMillis());
83     }
84
85     private void logCheckOut(ServletRequestEvent sre)
86     {
87         //TODO: has not implemented the invoking of checkout event.
88         ServletContext server = sre.getServletContext();
89         @SuppressWarnings("unchecked")
90         Map<String, UserActivity> users =
91             (Map<String, UserActivity>) server.getAttribute
92             (AnalyticEngine.CLIENTS);
93         UserActivity user = users.get
94             (((HttpServletRequest)sre.getServletRequest()).getSession().getId());
95         user.checkoutTime = System.currentTimeMillis();
96     }
97 }
98
99
100

```

UserActivity.java

```
1 package analytics;
2
3 public class UserActivity
4 {
5     String sessionId;
6     long startTime;
7     final int SIZE = 100;
8     long[] addItemTime;
9     int counter;
10    int limitCounter;
11    long checkoutTime;
12
13    //Construct a UserActivity object, initializing sessionId and startTime
14    UserActivity(String sessionId, long startTime)
15    {
16        this.sessionId = sessionId;
17        this.startTime = startTime;
18        addItemTime = new long[SIZE];
19        counter = 0;
20        limitCounter = SIZE;
21    }
22
23    void logAddItem (long time)
24    {
25        addItemTime[counter++] = time;
26        if (counter > limitCounter -1)
27        {
28            limitCounter = limitCounter + SIZE;
29            long[] newArray = new long[limitCounter];
30            System.arraycopy(addItemTime, 0, newArray, 0, counter);
31            addItemTime = newArray;
32        }
33    }
34 }
```

Account.java

```

1 package ctrl;
2
3 import java.io.File;
13
14
15 @WebServlet("/Account")
16 public class Account extends HttpServlet {
17     private static final long serialVersionUID = 1L;
18
19     protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
        ServletException, IOException {
20
21         request.getSession().setAttribute("back", request.getRequestURL ().toString());
22
23         String user = (String) request.getSession().getAttribute("user");
24         request.setAttribute("user", user);
25
26         try {
27             Engine engine = Engine.getInstance();
28             if (request.getParameter("view") != null) {
29                 //Server the PO view
30                 String view = request.getParameter("view") + ".xml";
31                 File file = new File(engine.getXmlFolderPath() + view);
32                 if (file.exists()) {
33                     OrderBean order =
34                         engine.convertFromXMLFileToObject(file, user);
35                     request.setAttribute("order", order);
36                 } else {
37                     file = new File(engine.getXmlIPOProcessedFolderPath()
38                         + view);
39                     if (file.exists()) {
40                         OrderBean order =
41                             engine.convertFromXMLFileToObject(file, user);
42                         request.setAttribute("order", order);
43                     } else {
44                         request.setAttribute("error", "File not found");
45                     }
46                 }
47             }
48             else {
49                 request.setAttribute("fileNames", engine.getXMLLinks(user,
50                     engine.getXmlFolderPath()));
51                 request.setAttribute("processedfileNames",
52                     engine.getXMLLinks(user, engine.getXmlIPOProcessedFolderPath()));
53             }
54         } catch (Exception e) { Page 1

```

Account.java

```
51         e.printStackTrace();
52     }
53
54
55     this.getServletContext().getRequestDispatcher("/
Account.jspx").forward(request, response);
56
57     }
58
59
60     protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
61         doGet(request, response);
62     }
63
64 }
```

Add.java

```
1 package ctrl;
2
3 import java.io.IOException;
19
20 @WebServlet("/Add")
21 public class Add extends HttpServlet
22 {
23     private static final long serialVersionUID = 1L;
24
25     public Add()
26     {
27         super();
28     }
29
30     protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
        ServletException, IOException
31     {
32         HttpSession session = request.getSession();
33         session.setAttribute("back", request.getRequestURL().toString());
34         if (session.getAttribute("user") != null) {
35             String user = (String) session.getAttribute("user");
36             request.setAttribute("user", user);
37         }
38         OrderBean order = ((OrderBean) session.getAttribute("order"));
39         try
40         {
41             Engine engine = Engine.getInstance();
42
43             if(request.getParameter("select_item")!=null)//add item
44             {
45                 String number = request.getParameter("number");
46                 String quantity = request.getParameter("qty");
47                 engine.orderAddItem(order, number, quantity);
48             }
49             if(request.getParameter("update")!=null)//update carts
50             {
51                 List<String> toDelete = new ArrayList<String>();
52                 Map<String, String> itemsQty = new HashMap<String, String>();
53                 List<String> parameters = Collections.list
                    (request.getParameterNames());
54                 for(String parameter: parameters)
55                 {
56                     if (parameter.startsWith("delete"))
57                         toDelete.add(request.getParameter(parameter));
58                     else if(parameter.matches("\\d{4}[A-Z|a-z]\\d{3}")
59                         itemsQty.put(parameter, request.getParameter
                    (parameter));
60                 }
```

Add.java

```
61         engine.updateOrder(order, itemsQty, toDelete);
62     }
63 }
64 catch (Exception e)
65 {
66     request.setAttribute("error", e.getMessage());
67 }
68
69 String checkoutName = "checkout";
70 if (session.getAttribute("user") != null)
71     checkoutName = "confirm";
72 request.setAttribute("checkout", checkoutName);
73
74 request.setAttribute("order", order);
75 this.getServletContext().getRequestDispatcher("/Add.jsp").forward (request, response);
76 }
77
78 protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
    ServletException, IOException
79 {
80     doGet(request, response);
81 }
82
83 }
```

Analytics.java

```
1 package ctrl;
2
3 import java.io.IOException;
14
15 /**
16  * Servlet implementation class Analytics
17  */
18 @WebServlet("/Analytics")
19 public class Analytics extends HttpServlet
20 {
21     private static final long serialVersionUID = 1L;
22
23     public Analytics()
24     {
25         super();
26     }
27
28     protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
        ServletException, IOException
29     {
30         AnalyticEngine engine = AnalyticEngine.getInstance();
31
32         @SuppressWarnings("unchecked")
33         double aveAddItem = engine.computeAveAddItemInterval((Map<String,
        UserActivity>)
34 request.getServletContext().getAttribute (AnalyticEngine.CLIENTS));
35
36         System.out.println("aveAddItem=" + aveAddItem);
37         if(aveAddItem > 0)
38         {
39             request.setAttribute("aveAddItem", aveAddItem);
40         }
41         else
42         {
43             request.setAttribute("rpt_AddItem", "No Users Yet");
44         }
45
46         @SuppressWarnings("unchecked")
47         double aveCheckout = engine.computeCheckOutInterval(((Map<String,
        UserActivity>)
48 request.getServletContext().getAttribute (AnalyticEngine.CLIENTS)));
49
50         System.out.println("aveCheckout=" + aveCheckout);
51         if(aveCheckout > 0)
52         {
53             request.setAttribute("aveCheckout", aveCheckout);
54         }
```

```
54         else
55         {
56             request.setAttribute("rpt_Checkout", "No Users Checked out Yet");
57         }
58
59
60 this.getServletContext().getRequestDispatcher("/
Analytics.jspx").forward(request, response);
61     }
62
63 protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException
64     {
65         doGet(request, response);
66     }
67
68 }
```


Browse.java

```
1 package ctrl;
2
3 import java.io.IOException;
4 import javax.servlet.ServletException;
5 import javax.servlet.annotation.WebServlet;
6 import javax.servlet.http.HttpServlet;
7 import javax.servlet.http.HttpServletRequest;
8 import javax.servlet.http.HttpServletResponse;
9
10 import model.Engine;
11
12 @WebServlet({ "/Browse" })
13 public class Browse extends HttpServlet
14 {
15     private static final long serialVersionUID = 1L;
16
17     public Browse()
18     {
19         super();
20     }
21
22     protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
        ServletException, IOException
23     {
24         try
25         {
26             Engine engine = Engine.getInstance();
27             request.setAttribute("catalogs", engine.getCatalogs());
28             if(request.getParameter("orderBy")!= null)
29             {
30                 String cat = request.getParameter("cat");
31                 if (cat != null && !cat.isEmpty() && !cat.equals("NONE"))
32                 {
33                     request.setAttribute("items",engine.getItems(cat,
34 request.getParameter("orderBy")));
35                     request.setAttribute("cat",request.getParameter("cat"));
36                 }
37                 else if (request.getParameter("number")!= null)
38                 {
39                     request.setAttribute("items",engine.searchItem
40 (request.getParameter("number")));
41                     request.setAttribute("number",request.getParameter
42 ("number"));
43                 }
44                 request.getParameter("orderBy");
45             }
46             else if(request.getParameter("cat")!=null)
47             {
48             }
```

Browse.java

```
45 request.setAttribute("items",engine.getItems (request.getParameter("cat")));
46         request.setAttribute("cat",request.getParameter("cat"));
47     }
48     else if(request.getParameter("select_item").equals("search"))
49     {
50 request.setAttribute("items",engine.searchItem (request.getParameter("number")));
51         request.setAttribute("number",request.getParameter("number"));
52     }
53
54 }
55 catch (Exception e)
56 {
57     request.setAttribute("error", e.getMessage());
58 }
59 if (request.getSession().getAttribute("user") != null) {
60     String user = (String) request.getSession().getAttribute("user");
61     request.setAttribute("user", user);
62 }
63
64 this.getServletContext().getRequestDispatcher("/Browse.jsp").forward (request, response);
65 }
66
67 protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException
68 {
69     doGet(request, response);
70 }
71
72 }
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
```

Browse.java

89
90
91
92
93
94
95
96
97

```

1 package ctrl;
2
3 import java.io.IOException;
11
12 @WebServlet({ "/Browse" })
13 public class Browse extends HttpServlet
14 {
15     private static final long serialVersionUID = 1L;
16
17     public Browse()
18     {
19         super();
20     }
21
22     protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
        ServletException, IOException
23     {
24         try
25         {
26
27             if (request.getSession().getAttribute("user") != null) {
28                 String user = (String) request.getSession().getAttribute
29                 ("user");
30                 request.setAttribute("user", user);
31             }
32
33             Engine engine = Engine.getInstance();
34             request.setAttribute("catalogs", engine.getCatalogs());
35             if(request.getParameter("orderBy")!= null)
36             {
37                 String cat = request.getParameter("cat");
38                 if (cat != null && !cat.isEmpty() && !cat.equals("NONE"))
39                 {
40                     request.setAttribute("items",engine.getItems(cat,
41                     request.getParameter("orderBy")));
42                     request.setAttribute("cat",request.getParameter("cat"));
43                 }
44                 else if (request.getParameter("number")!= null)
45                 {
46                     request.setAttribute("items",engine.searchItem
47                     (request.getParameter("number")));
48                     request.setAttribute("number",request.getParameter
49                     ("number"));
50                 }
51             }
52             else if(request.getParameter("cat")!=null)
53             {
54                 request.setAttribute("items",engine.getItems

```

Browse.java

```
(request.getParameter("cat")));
51         request.setAttribute("cat",request.getParameter("cat"));
52     }
53     else if(request.getParameter("select_item").equals("search"))
54     {
55 request.setAttribute("items",engine.searchItem (request.getParameter("number")));

56         request.setAttribute("number",request.getParameter("number"));
57     }
58
59     request.getSession().setAttribute("back", request.getRequestURL
60 ());
61     }
62     catch (Exception e)
63     {
64         request.setAttribute("error", e.getMessage());
65 this.getServletContext().getRequestDispatcher("/Browse.jsp").forward (request, response);

66     }
67
68 protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException
69     {
70         doGet(request, response);
71     }
72
73 }
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
```

Browse.java

94
95
96
97
98

Login.java

```
1 package ctrl;
2
3 import java.io.IOException;
4
5
6
7
8
9
10 @WebServlet("/Login")
11 public class Login extends HttpServlet {
12     private static final long serialVersionUID = 1L;
13
14     public Login()
15     {
16         super();
17     }
18
19     protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException
20     {
21
22         if (request.getParameter("user")!=null)
23         {
24             String user= request.getParameter("user");
25             String name = request.getParameter("name");
26             String back = (String) request.getSession().getAttribute("back");
27             request.getSession().setAttribute("user", user);
28             request.getSession().setAttribute("name",name);
29             request.setAttribute("user", user);
30             request.setAttribute("name", name);
31             //this.getServletContext().getRequestDispatcher("/
Login.jspx").forward(request, response);
32             response.sendRedirect(back);
33         }
34         else if(request.getParameter("login")!=null)
35         {
36             String url = "https://www.eecs.yorku.ca/~roumani/servers/auth/
oauth.cgi"+
37                 "?back="+request.getRequestURL();
38             response.sendRedirect(url);
39         }
40         else
41         {
42             request.setAttribute("error", "Wrong Account Number and/or PIN.\n
Please check your Account and PIN number you have entered");
43             this.getServletContext().getRequestDispatcher("/
Login.jspx").forward(request, response);
44         }
45     }
46 }
47
48 protected void doPost(HttpServletRequest request, HttpServletResponse
```

Login.java

```
    response) throws ServletException, IOException
49    {
50        doGet(request, response);
51    }
52
53 }
```



```

1 package model;
2
3 import java.io.File;
13
14 public class Engine
15 {
16     static final double HST_RATE = 0.13;
17     static final double MIN_SHIPPING_WAIVER = 70;
18     static final double SHIPPING_COST = 5;
19     static Engine engine = null;
20
21     private ItemDAO itemDAO;
22     private String xmlPOFolderPath;
23     private String xmlPOProcessedFolderPath;
24     private int count = 0;
25
26     private Engine() throws Exception
27     {
28         itemDAO = ItemDAO.getInstance();
29         initializeFilePath();
30         initializeCount();
31     }
32     public synchronized static Engine getInstance() throws Exception
33     {
34         try
35         {
36             if (engine == null)
37                 engine = new Engine();
38             return engine;
39         }
40         catch (Exception e)
41         {
42             throw new Exception("Issue at instatiating engine", e);
43         }
44     }
45     public List<ItemBean> getItems(String catalog) throws Exception
46     {
47         try
48         {
49             return itemDAO.getItems(catalog, null);
50         }
51         catch (Exception e)
52         {
53             throw new Exception("Fail to get items", e);
54         }
55     }
56     public List<ItemBean> getItems(String catalog, String orderBy) throws
Exception

```

```

57     {
58         try
59         {
60             return itemDAO.getItems(catalog, orderBy);
61         }
62         catch (Exception e)
63         {
64             throw new Exception ("Fail to get items", e);
65         }
66     }
67
68     public List<ItemBean> searchItem(String number) throws Exception
69     {
70         List<ItemBean> result = new ArrayList<ItemBean>();
71         result.add(getItem(number));
72         //TODO
73         return result;
74     }
75     public ItemBean getItem(String number) throws Exception
76     {
77         try
78         {
79             return itemDAO.getItem(number);
80         }
81         catch (Exception e)
82         {
83             throw new Exception("Fail to find item of number " + number);
84         }
85     }
86
87     public List<CatalogBean> getCatalogs() throws Exception
88     {
89         try
90         {
91             return itemDAO.getCatalogs();
92         }
93         catch (Exception e)
94         {
95             throw new Exception ("Fail to get catalogs",e);
96         }
97     }
98
99     //return the 'existing item' in the stack if found, else return null;
100    private ItemBean isExistingItem(List<ItemBean> items, String number)
101    {
102        ItemBean item = null;
103        for(int i = 0; i < items.size() && item == null; i++)
104        {

```

Engine.java

```

105         if (items.get(i).getNumber().equals(number))
106             item = items.get(i);
107     }
108     return item;
109 }
110
111 private void updateOrderTotalPrice(OrderBean order)
112 {
113     List<ItemBean> items = order.getItems();
114     double total = 0;
115     for (ItemBean item: items)
116     {
117         total = total + (item.getPrice() * item.getQuantity());
118     }
119     order.setTotal(total);
120     order.setHST(total*HST_RATE);
121
122     //set shipping cost
123     if (total == 0 || total > MIN_SHIPPING_WAIVER )
124         order.setShipping(0);
125     else
126         order.setShipping(SHIPPING_COST);
127
128     order.setGrandTotal(order.getTotal() + order.getHST() + order.getShipping());
129 }
130
131 public OrderBean orderAddItem(OrderBean order, String number, String qty) throws Exception
132 {
133     int quantity;
134     try
135     {
136         quantity = Integer.parseInt(qty);
137     }
138     catch (Exception e)
139     {
140         throw new Exception("Invalid quantity entries " + qty);
141     }
142
143     try
144     {
145         ItemBean item = isExistingItem(order.getItems(), number);
146         if (item != null)
147         {
148             item.setQuantity(item.getQuantity() + quantity);
149         }
150     }
    else

```

Engine.java

```

151         {
152             item = this.getItem(number);
153             item.setQuantity(quantity);
154             order.getItems().add(item);
155         }
156         updateOrderTotalPrice(order);
157         return order;
158     }
159 }
160 catch (Exception e)
161 {
162     throw new Exception("Fail to add "+ qty +" of item number " + number + " into order");
163 }
164 }
165
166 private void orderDelItems(OrderBean order, List<String> toDelete)
167 {
168     List<ItemBean> items = order.getItems();
169     for (String e: toDelete)
170     {
171         ItemBean item = isExistingItem(items, e);
172         if(item !=null)
173             items.remove(isExistingItem(order.getItems(), e));
174     }
175 }
176 private void orderUpdateQty (OrderBean order, Map<String, String> itemsQty) throws
Exception
177 {
178     List<ItemBean> items = order.getItems();
179     Set<String> toUpdate = itemsQty.keySet();
180     for(String itemNo: toUpdate)
181     {
182         ItemBean item = isExistingItem(items, itemNo);
183         if (item!=null)
184         {
185             int quantity;
186             try
187             {
188                 quantity = Integer.parseInt(itemsQty.get(itemNo));
189                 if (quantity < 0 )
190                     throw new Exception ();
191             }
192             catch (Exception e)
193             {
194                 throw new Exception ("Invalid quantity entries!");
195             }
196             if (quantity == 0)

```

Engine.java

```

197             items.remove(item);
198             item.setQuantity(quantity);
199         }
200     }
201 }
202
203 public OrderBean updateOrder(OrderBean order, Map<String, String> itemsQty,
    List<String> toDelete) throws Exception
204 {
205     orderDellItems(order, toDelete);
206     orderUpdateQty(order, itemsQty);
207     updateOrderTotalPrice(order);
208     return order;
209 }
210
211 public void JAXBObjectToXML(OrderBean order, String id, String path) throws Exception
212 {
213     //Create JAXB Context
214     JAXBContext jaxbContext = JAXBContext.newInstance
215 (OrderBean.class);
216
217     //Create Marshaller
218     Marshaller jaxbMarshaller = jaxbContext.createMarshaller();
219
220     //Required formatting??
221     jaxbMarshaller.setProperty(Marshaller.JAXB_FORMATTED_OUTPUT,
222 Boolean.TRUE);
223
224     //Store XML to File
225     File filePath = new File(path + String.format("%s.xml", id));
226     if (!filePath.exists()) {
227         filePath.createNewFile();
228     }
229
230     //Writes XML file to file-system
231     jaxbMarshaller.marshal(order, filePath);
232 }
233
234 public OrderBean convertFromXMLFileToObject(File file, String user) {
235     JAXBContext jaxbContext;
236     try
237     {
238         jaxbContext = JAXBContext.newInstance(OrderBean.class);
239         Unmarshaller jaxbUnmarshaller = jaxbContext.createUnmarshaller();
240         OrderBean order = (OrderBean) jaxbUnmarshaller.unmarshal(file);

```

Engine.java

```
241         if (order.getAccount() != null && order.getAccount().equals(user))
242             return order;
243     }
244     catch (JAXBException e)
245     {
246         e.printStackTrace();
247     }
248
249     return null;
250
251 }
252 public ArrayList<String> getXMLLinks(String user, String path) {
253     ArrayList<String> names = new ArrayList<>();
254     File[] files = new File(path).listFiles();
255
256     for (File file : files) {
257         if (!file.isDirectory() && file.getName().contains(".xml")) {
258
259             OrderBean order = convertFromXMLFileToObject(file, user);
260             if (order != null)
261             {
262                 String[] f = file.getName().split(".xml");
263                 names.add(f[0]);
264             }
265         }
266     }
267     return names;
268 }
269
270
271 public String getXmlIPOProcessedFolderPath() {
272     return xmlIPOProcessedFolderPath;
273 }
274
275 public String getXmlFolderPath() {
276     return xmlPOFolderPath;
277 }
278
279 public int increment() {
280     count++;
281     return count;
282 }
283
284
285
286 private void initializeFilePath() {
287
288     if (xmlPOFolderPath == null) {
```

```

289         xmlPOFolderPath = System.getProperty("user.dir") + "/appData/
PO/";
290         File filePath = new File(xmlPOFolderPath);
291         //Create folder if they don't exist
292         if (!filePath.exists()) {
293             try{
294                 if (filePath.getParentFile().exists())
295                     filePath.getParentFile().mkdirs();
296
297                 filePath.mkdirs();
298             }
299
300             catch(SecurityException se){
301                 //handle it
302             }
303         }
304     }
305
306     if (xmlPOProcessedFolderPath == null) {
307         xmlPOProcessedFolderPath = System.getProperty("user.dir") + "/ appData/PO_processed/";
308         File filePath = new File(xmlPOProcessedFolderPath);
309         if (!filePath.exists())
310             filePath.mkdirs();
311     }
312 }
313
314
315 private void initializeCount() {
316
317     File[] files = new File(xmlPOFolderPath).listFiles();
318
319     for (File file : files) {
320         if (!file.isDirectory() && file.getName().contains(".xml")) {
321             String[] f = file.getName().split(".xml");
322             int num = Integer.parseInt(f[0].split("_")[1]);
323             if (num > count)
324                 count = num;
325         }
326     }
327
328     files = new File(xmlPOProcessedFolderPath).listFiles();
329     for (File file : files) {
330         if (!file.isDirectory() && file.getName().contains(".xml")) {
331             String[] f = file.getName().split(".xml");
332             int num = Integer.parseInt(f[0].split("_")[1]);
333             if (num > count)
334                 count = num;

```

```
335         }
336     }
337 }
338
339 }
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
```


ItemDAO.java

```
1 package model;
2
3 import java.sql.Connection;
4
5
6
7
8
9
10 class ItemDAO
11 {
12     private static final String DB_URL = "jdbc:derby://localhost:64413/
13     EECS;user=student;password=secret";
14     private static ItemDAO instance = null;
15
16     private ItemDAO() throws Exception
17     {
18         Class.forName("org.apache.derby.jdbc.ClientDriver").newInstance();
19     }
20
21     static synchronized ItemDAO getInstance() throws Exception
22     {
23         if(instance == null)
24             instance = new ItemDAO();
25         return instance;
26     }
27
28     private class GeneratePreparedStatement
29     {
30         private Connection conn;
31         PreparedStatement ps;
32
33         GeneratePreparedStatement(String query) throws Exception
34         {
35             conn = DriverManager.getConnection(DB_URL);
36             ps = conn.prepareStatement("SET SCHEMA ROUMANI");
37             ps.executeUpdate();
38             ps.close();
39             ps = conn.prepareStatement(query);
40         }
41
42         PreparedStatement getPs()
43         {
44             return ps;
45         }
46
47         void closeConnection() throws Exception
48         {
49             if(!ps.isClosed()) this.ps.close();
50             if(!conn.isClosed()) this.conn.close();
51         }
52     }
53 }
```

```

53     }
54     List<ItemBean> getItems(String catalog, String orderBy) throws Exception
55     {
56         GeneratePreparedStatement psGenerator = null;
57         ResultSet rs = null;
58         try
59         {
60             //building sql
61             String sql = "SELECT * FROM ITEM";
62             sql = sql + " WHERE CATID=" + Integer.parseInt
(catalog);
63             sql = orderBy == null? sql : sql + " ORDER BY " + orderBy;
64             System.out.println("sql=" + sql);
65             psGenerator = new GeneratePreparedStatement(sql);
66             PreparedStatement ps = psGenerator.getPs();
67             rs = ps.executeQuery();
68             List<ItemBean> items = new ArrayList<ItemBean>();
69             while(rs.next())
70             {
71                 //ItemBean(String name, String number, double price, int
catalog, String unit)
72                 String name = rs.getString("NAME");
73                 String number = rs.getString("NUMBER");
74                 double price = rs.getDouble("PRICE");
75                 int catalogID = rs.getInt("CATID");
76                 String unit = rs.getString("UNIT");
77
78                 items.add(new ItemBean(name, number, price, catalogID,
unit));
79             }
80             if (items.size() < 1)
81                 throw new Exception("No Items Found");
82             return items;
83         }
84         catch (Exception e)
85         {
86             throw e;
87         }
88         finally
89         {
90             if(rs != null && !rs.isClosed()) rs.close();
91             if(psGenerator != null ) psGenerator.closeConnection();
92         }
93     }
94
95
96     ItemBean getItem(String number) throws Exception
97     {

```

ItemDAO.java

```

98      GeneratePreparedStatement psGenerator = null;
99      ResultSet rs = null;
100     try
101     {
102         String sql = "SELECT * FROM ITEM WHERE NUMBER=" + number + "";
103
104         psGenerator = new GeneratePreparedStatement(sql);
105         PreparedStatement ps = psGenerator.getPs();
106         rs = ps.executeQuery();
107
108         if (rs.getRow() < 0)
109             throw new Exception ("No item with number " + number);
110         ItemBean result = null;
111         if(rs.next())
112         {
113             String name = rs.getString("NAME");
114             String itemNumber = rs.getString("NUMBER");
115             double price = rs.getDouble("PRICE");
116             int catalogID = rs.getInt("CATID");
117             String unit = rs.getString("UNIT");
118
119             result = new ItemBean(name, itemNumber, price, catalogID,
120 unit);
121         }
122         return result;
123     }
124     catch (Exception e)
125     {
126         throw e;
127     }
128     finally
129     {
130         if(rs != null && !rs.isClosed()) rs.close();
131         if(psGenerator != null ) psGenerator.closeConnection();
132     }
133 }
134
135 List<CatalogBean> getCatalogs() throws Exception
136 {
137     GeneratePreparedStatement psGenerator = null;
138     ResultSet rs = null;
139     try
140     {
141         String sql = "SELECT * FROM CATEGORY"; Page

```

```
145
146     psGenerator = new GeneratePreparedStatement(sql);
147     PreparedStatement ps = psGenerator.getPs();
148     rs = ps.executeQuery();
149     List<CatalogBean> result = new ArrayList<CatalogBean>();
150     while(rs.next())
151     {
152         String name = rs.getString("NAME");
153         String description = rs.getString("DESCRIPTION");
154         int id = rs.getInt("ID");
155         result.add(new CatalogBean(name, description, id));
156     }
157     return result;
158 }
159 catch (Exception e)
160 {
161     throw e;
162 }
163 finally
164 {
165     if(rs != null && !rs.isClosed()) rs.close();
166     if(psGenerator != null ) psGenerator.closeConnection();
167 }
168 }
169
170 }
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
```

ItemDAO.java

193
194
195
196
197
198
199
200
201
202

ItemBean.java

```
1 package model;
2
3 public class ItemBean
4 {
5     private String name;
6     private String number;
7     private double price;
8     private int quantity;
9     private double extended;
10    private int catalog;
11    private String unit;
12
13    public ItemBean() {
14
15    }
16
17
18    public ItemBean(String name, String number, double price, int catalog, String unit, int quantity,
19        double extended)
20    {
21        super();
22        this.name = name;
23        this.number = number;
24        this.price = price;
25        this.quantity = quantity;
26        this.extended = extended;
27        this.catalog = catalog;
28        this.unit = unit;
29    }
30    public ItemBean(String name, String number, double price, int catalog, String unit)
31    {
32        this(name, number, price, catalog, unit, 0, 0.0);
33    }
34
35    public String getUnit()
36    {
37        return unit;
38    }
39
40    public void setUnit(String unit)
41    {
42        this.unit = unit;
43    }
44
45    public String getName()
46    {
```

```
47         return name;
48     }
49
50     public void setName(String name)
51     {
52         this.name = name;
53     }
54
55     public String getNumber()
56     {
57         return number;
58     }
59
60     public void setNumber(String number)
61     {
62         this.number = number.toUpperCase();
63     }
64
65     public double getPrice()
66     {
67         return price;
68     }
69
70     public void setPrice(double price)
71     {
72         this.price = price;
73     }
74
75     public int getQuantity()
76     {
77         return quantity;
78     }
79
80     public void setQuantity(int quantity)
81     {
82         this.quantity = quantity;
83     }
84
85     public double getExtended()
86     {
87         return extended;
88     }
89
90     public void setExtended(double extended)
91     {
92         this.extended = extended;
93     }
94
```

ItemBean.java

```
95     public int getCatalog()
96     {
97         return catalog;
98     }
99
100    public void setCatalog(int catalog)
101    {
102        this.catalog = catalog;
103    }
104
105    @Override
106    public String toString()
107    {
108        return "ItemBean [name=" + name + ", number=" + number + ", price=" + price + ", quantity=" + quantity
109            + ", extended=" + extended + ", catalog=" + catalog + ",
110            unit=" + unit + "];"
111    }
112
113 }
```


CatalogBean.java

```
1 package model;
2
3 public class CatalogBean
4 {
5     String name;
6     String description;
7     int id;
8     public CatalogBean(String name, String description, int id)
9     {
10         super();
11         this.name = name;
12         this.description = description;
13         this.id = id;
14     }
15     public String getName()
16     {
17         return name;
18     }
19     public void setName(String name)
20     {
21         this.name = name;
22     }
23     public String getDescription()
24     {
25         return description;
26     }
27     public void setDescription(String description)
28     {
29         this.description = description;
30     }
31     public int getId()
32     {
33         return id;
34     }
35     public void setId(int id)
36     {
37         this.id = id;
38     }
39     @Override
40     public String toString()
41     {
42         return "CatalogBean [name=" + name + ", description=" + description + ", id=" + id + "]\n";
43     }
44
45 }
```

```
1 package model;
2
3 import java.util.ArrayList;
4
5
6
7
8
9 @XmlElement(name = "product")
10 public class OrderBean
11 {
12     // CustomerBean customer
13     private List<ItemBean> items;
14     private double total;
15     private double shipping;
16     private double HST;
17     private double grandTotal;
18     private String id;
19     private int orderNumber;
20     private Date submitted;
21     private String account;
22
23     public int getOrderNumber() {
24         return orderNumber;
25     }
26     public void setOrderNumber(int orderNumber) {
27         this.orderNumber = orderNumber;
28     }
29     public String getAccount()
30     {
31         return account;
32     }
33     public void setAccount(String account)
34     {
35         this.account = account;
36     }
37     public OrderBean(List<ItemBean> items)
38     {
39         super();
40         this.items = items;
41     }
42
43     public OrderBean()
44     {
45         super();
46         this.items = null;
47     }
48     public List<ItemBean> getItems()
49     {
50         return items;
51     }
```

```
52     public void setItems(List<ItemBean> items)
53     {
54         this.items = items;
55     }
56     public double getTotal()
57     {
58         return total;
59     }
60     public void setTotal(double total)
61     {
62         this.total = total;
63     }
64     public double getShipping()
65     {
66         return shipping;
67     }
68     public void setShipping(double shipping)
69     {
70         this.shipping = shipping;
71     }
72     public double getHST()
73     {
74         return HST;
75     }
76     public void setHST(double hST)
77     {
78         HST = hST;
79     }
80     public double getGrandTotal()
81     {
82         return grandTotal;
83     }
84     public void setGrandTotal(double grandTotal)
85     {
86         this.grandTotal = grandTotal;
87     }
88     public String getId()
89     {
90         return id;
91     }
92     public void setId(String id)
93     {
94         this.id = id;
95     }
96     public Date getSubmitted()
97     {
98         return submitted;
99     }
```

OrderBean.java

```
100     public void setSubmitted(Date submitted)
101     {
102         this.submitted = submitted;
103     }
104     @Override
105     public String toString()
106     {
107         return "OrderBean [items=" + items + ", total=" + total + ",
shipping=" + shipping + ", HST=" + HST
108             + ", grandTotal=" + grandTotal + ", id=" + id + ",
submitted=" + submitted + "]\n";
109     }
110
111     public boolean isEmpty() {
112         if (items == null)
113             return true;
114
115         return items.size() == 0;
116     }
117     public void clear() {
118
119         items = new ArrayList<>();
120         total = 0;
121         shipping = 0;
122         HST = 0;
123         grandTotal = 0;
124         submitted = null;
125     }
126 }
127
128
129
130 }
```

Account.jspx

```
1 <?xml version="1.0" encoding="UTF-8" ?>
2 <jsp:root xmlns:jsp="http://java.sun.com/JSP/Page" version="2.0"
3     xmlns:c="http://java.sun.com/jsp/jstl/core"
4     xmlns:fmt="http://java.sun.com/jsp/jstl/fmt">
5 <jsp:directive.page contentType="text/html; charset=UTF-8"
6     pageEncoding="UTF-8" session="false"/>
7 <jsp:output doctype-root-element="html"
8     doctype-public="-//W3C//DTD XHTML 1.0 Transitional//EN"
9     doctype-system="http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd"
10    omit-xml-declaration="true" />
11 <html xmlns="http://www.w3.org/1999/xhtml" lang="en">
12 <head >
13     <title>Foods R US</title>
14 </head>
15 <body>
16 <c:if test="${not empty order}">
17
18     <div>
19
20         <h1>Details of Order:</h1>
21
22         Account Name: ${order.account}<br/>
23         Order number: ${order.orderNumber}<br/>
24         Order date: ${order.submitted}<br/>
25     </div>
26     <div>
27         <h2>Your order items:</h2>
28         <table border="1">
29             <tbody>
30                 <tr>
31                     <td>Product ID</td>
32                     <td>Name</td>
33                     <td>Price</td>
34                     <td>Quantity</td>
35                 </tr>
36
37                 <c:forEach var="e" items="${order.items}">
38                     <tr>
39                         <td>${e.number}</td>
40                         <td>${e.name}</td>
41                         <td>${e.price}</td>
42                         <td>${e.quantity}</td>
43                     </tr>
44                 </c:forEach>
45
46             </tbody>
47         </table>
48     </div>
```

Account.jspx

```

49
50     <div>
51         <table>
52             <tbody>
53                 <tr>
54                     <td></td>
55                     <td>Sub-Total:</td>
56                     <td>${order.total}</td>
57                 </tr>
58                 <tr>
59                     <td></td>
60                     <td>HST:</td>
61                     <td>${order.HST}</td>
62                 </tr>
63                 <tr>
64                     <td></td>
65                     <td>Shipping:</td>
66                     <td>${order.shipping}</td>
67                 </tr>
68                 <tr>
69                     <td></td>
70                     <td>Grand Total:</td>
71                     <td>${order.grandTotal}</td>
72                 </tr>
73             </tbody>
74         </table>
75     </div>
76
77
78     <div>
79
80         <button onclick="javascript:window.print()">Print Window</button>
81
82     </div>
83
84 </c:if>
85
86 <c:if test="${empty order}">
87     <jsp:include page="nav.jspx"/>
88     <h1>New Orders (Not Processed)</h1>
89 </c:if>
90
91 <c:if test="${not empty fileNames}">
92     <c:forEach var="e" items="${fileNames}">
93         <a target="_blank" href="Account?view=${e}">${e}</a><br/>
94     </c:forEach>
95 </c:if>
96

```

Account.jspx

```
97 <h1>Old Orders (Processed)</h1>
98 <c:if test="${not empty processedfileNames}">
99     <c:forEach var="e" items="${processedfileNames}">
100         <a target="_blank" href="Account?view=${e}">${e}</a><br/>
101     </c:forEach>
102 </c:if>
103
104 <c:if test="${not empty error}">
105     <h1 style="color:red;">${error}</h1>
106 </c:if>
107
108 </body>
109 </html>
110 </jsp:root>
111
112
113
114
115
116
117
118
119
120
121
122
123
```

Add.jspx

```
1 <?xml version="1.0" encoding="UTF-8" ?>
2 <jsp:root xmlns:jsp="http://java.sun.com/JSP/Page" version="2.0"
3     xmlns:c="http://java.sun.com/jsp/jstl/core"
4     xmlns:fmt="http://java.sun.com/jsp/jstl/fmt">
5 <jsp:directive.page contentType="text/html; charset=UTF-8"
6     pageEncoding="UTF-8" session="false"/>
7 <jsp:output doctype-root-element="html"
8     doctype-public="-//W3C//DTD XHTML 1.0 Transitional//EN"
9     doctype-system="http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd"
10    omit-xml-declaration="true" />
11 <html xmlns="http://www.w3.org/1999/xhtml" lang="en">
12 <head>
13     <meta charset="utf-8" />
14     <meta content="width=device-width, initial-scale=1" name="viewport" />
15 <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/
16     bootstrap/3.3.7/css/bootstrap.min.css" />
17 <link href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/ bootstrap.min.css"
18     rel="stylesheet" />
19     <link href="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/
20     jquery.min.js" rel="stylesheet" />
21 </head>
22 <body>
23     <jsp:include page="nav.jspx"/>
24     <h2>Shopping Cart</h2>
25     <form action="Add" method="POST">
26         <table class="table">
27             <thead><tr>
28                 <th>Product ID</th>
29                 <th>Name</th>
30                 <th>Unit Price</th>
31                 <th>Quantity</th>
32                 <th>Unit</th>
33                 <th>Total</th>
34                 <th>Remove Item</th>
35             </tr></thead>
36             <c:forEach var="e" items="{order.items }">
37                 <tr>
38                     <td>${e.number }</td>
39                     <td>${e.name }</td>
40                     <td><fmt:formatNumber type="currency" value="${e.price }"/>
41                     <td><input type="number" step="1" name="${e.number}"
42                     value="${e.quantity}"/></td>
```


Add.jspx

```

44         <td>${e.unit }</td>
45         <td><fmt:formatNumber type="currency" value="${e.quantity
* e.price }"/></td>
46         <td><input type="checkbox" name="delete_${e.number}"
value="${e.number}"/> Delete</td>
47     </tr>
48 </c:forEach>
49 </table>
50 <div><input type="submit" class="btn btn-info" name="update" value="update" /></div>
51 <br/>
52 <div><input type="button" class="btn btn-info" value="Continue Shopping"
onclick="location='Browse'"/></div>
53 <br/>
54 </form>
55
56 <form action="Checkout" method="POST">
57     <input type="hidden" id="force" name="force" value="true"/>
58 <input type="submit" class="btn btn-info" name="checkout" value="${checkout}" />
59 </form>
60
61 <div id="pricing">
62     <div>Total: <fmt:formatNumber type="currency" value="${order.total }"/>
63 </div>
64     <div>HST: <fmt:formatNumber type="currency" value="${order.HST }"/></div>
65     <div>Shipping: <fmt:formatNumber type="currency"
value="${order.shipping }"/></div>
66 <div>Grand Total: <fmt:formatNumber type="currency" value="$
{order.grandTotal }"/></div>
67
68
69 <div id="advertisement"><c:if test="${not empty advertise }">
70     <h4>You might want to get ${advertise.name }</h4>
71     <div>Item No.: ${advertise.number }</div>
72     <div>${advertise.name }</div>
73 <div><fmt:formatNumber type="currency" value="${advertise.price }"/> / ${advertise.unit } </div>
74     <form action="Add" method="POST">
75         <div><label for="qty">Qty: </label>
76             <input type="number" name="qty" value="1" step="1"/></div>
77             <input type="hidden" name="number"
value="${advertise.number }"/>
78         <div><input type="submit" name="select_item" value="Add To Cart"/>
79     </div>
    </form>

```

Add.jspx

```
80     </c:if></div>
81
82     <c:if test="${not empty error}">
83         <div class="alert alert-danger">${error }</div>
84     </c:if>
85
86
87
88 </body>
89
90 </html>
91 </jsp:root>
```

Analytics.jspx

```
1 <?xml version="1.0" encoding="UTF-8" ?>
2 <jsp:root xmlns:jsp="http://java.sun.com/JSP/Page" version="2.0"
3       xmlns:c="http://java.sun.com/jsp/jstl/core"
4       xmlns:fmt="http://java.sun.com/jsp/jstl/fmt">
5 <jsp:directive.page contentType="text/html; charset=UTF-8"
6   pageEncoding="UTF-8" session="false"/>
7 <jsp:output doctype-root-element="html"
8   doctype-public="-//W3C//DTD XHTML 1.0 Transitional//EN"
9   doctype-system="http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd"
10  omit-xml-declaration="true" />
11 <html xmlns="http://www.w3.org/1999/xhtml" lang="en">
12 <head >
13   <meta content="width=device-width, initial-scale=1" name="viewport" />
14 <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/
15   bootstrap/3.3.7/css/bootstrap.min.css" />
16 <link href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/ bootstrap.min.css"
17   rel="stylesheet" />
18 <link href="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/ jquery.min.js"
19   rel="stylesheet" />
20 <title>Foods R US</title>
21 </head>
22 <body><div class="container">
23   <div class="panel-heading"><h2>Analytics Report</h2></div>
24   <div class="panel-group">
25     <div class="panel panel-info">
26       <div class="panel-heading">Average Time Users Add Items to Cart:
27     </div>
28     <div class="panel-body">
29       <c:choose>
30         <c:when test="${ not empty rpt_AddItem}">${rpt_AddItem }</
31       <c:when>
32         <c:otherwise>
33           <fmt:formatNumber maxFractionDigits="3"
34           value="${ aveAddItem}" /> seconds
35         </c:otherwise>
36       </c:choose>
37     </div>
38   </div>
39   <div class="panel panel-info">
40     <div class="panel-heading">Average Time Users Checkout: </div>
41     <div class="panel-body">
42       <c:choose>
43         <c:when test="${ not empty rpt_Checkout}">${rpt_Checkout }
44       </c:when>
45       <c:otherwise>
```

Analytics.jspx

```
42                                     <fmt:formatNumber maxFractionDigits="3"
    value="${ aveCheckout}" /> seconds
43                                     </c:otherwise>
44                                     </c:choose>
45                                 </div>
46                            </div>
47                    </div>
48
49</div></body>
50</html>
51</jsp:root>
```

Browse.jspx

```
1 <?xml version="1.0" encoding="UTF-8" ?>
2 <jsp:root xmlns:jsp="http://java.sun.com/JSP/Page" version="2.0"
3   xmlns:c="http://java.sun.com/jsp/jstl/core"
4   xmlns:fmt="http://java.sun.com/jsp/jstl/fmt">
5   <jsp:directive.page contentType="text/html; charset=UTF-8"
6     pageEncoding="UTF-8" session="false" />
7   <jsp:output doctype-root-element="html"
8     doctype-public="-//W3C//DTD XHTML 1.0 Transitional//EN"
9   doctype-system="http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd"
10   omit-xml-declaration="true" />
11   <html xmlns="http://www.w3.org/1999/xhtml" lang="en">
12 <head>
13 <meta charset="utf-8"></meta>
14 <meta name="viewport" content="width=device-width, initial-scale=1"></meta>
15 <link rel="stylesheet"
16 href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/
17 bootstrap.min.css"></link>
18 <link rel="stylesheet"
19 href="https://use.fontawesome.com/releases/v5.5.0/css/all.css"
20 integrity="sha384-B4dIYHKNBt8Bc12p+WXckhzcICo0wtJAoU8YZTY5qE0Id1GSseTk6S +L3BIXeVIU"
21 crossorigin="anonymous"></link>
22 <link rel="stylesheet" href="res/Style.css"></link>
23 <title>Foods R US</title>
24 </head>
25 <body>
26   <jsp:include page="nav.jspx"/>
27
28   <div class="container-fluid padding">
29     <div class="row welcome text-center">
30       <div class="col-12">
31         <a href="Browse"><button type="button" class="btn btn-info
32 btn-lg">Browse Categories</button></a>
33         <br></br>
34         <hr class="my-3"></hr>
35       </div>
36     </div>
37
38     <div class="container-fluid padding">
39       <div class="row text-center padding">
40         <c:forEach var="e" items="{catalogs }">
41           <div class="col-xs-12 col-sm-6 col-md-3">
42             <p class="category-image"
43               style="background-image: url
44               ({pageContext.request.contextPath}/res/images/{e.name}.jpg);"> Page 1
```

Browse.jspx

```

44         </p>
45         <h3>
46             <a href="Browse?cat=${e.id}" >${e.name }</a>
47         </h3>
48         <p>${e.description }</p>
49     </div>
50 </c:forEach>
51 </div>
52 </div>
53 <hr class="my-1"></hr>
54 </div>
55
56 <c:if test="${not empty items}">
57     <div class="container-fluid padding" id="items">
58         <br></br>
59         <h2 class="text-center">Browse Items Listing</h2>
60         <form action="Browse" method="POST">
61             <input type="hidden" name="cat" value="${cat }" />
62             <input type="hidden" name="number" value="${number }" />
63             Sort By: <select name="orderBy" class="form-control">
64                 <option selected="any" value="NONE">Select One:</option>
65                 <option value="PRICE">Price</option>
66                 <option value="NAME">Name</option>
67             </select>
68             <input type="submit" name="sort" value="Sort" />
69         </form>
70         <br></br>
71         <hr></hr>
72         <div class="row">
73             <c:forEach var="e" items="${items}">
74                 <div class="col-sm-4 col-md-3">
75                     <div class="panel panel-primary item col-md-10 col-
76                         sm-6 col-xs-12">
77                         <div class="col-md-10 col-sm-10 col-xs-10">
78                             <div class="panel-heading item-name">
79                                 ${e.name }</div>
80                                 <div class="panel-body item_no">Item No.:
81                                 ${e.number }</div>
82                                 <div>
83                                     <fmt:formatNumber type="currency"
84                                     value="${e.price }" />
85                                     / ${e.unit }
86                                 </div>
87                                 <form action="Add" method="POST">
88                                     <div>
89                                         <label for="qty">Qty: </label> <input
90                                         type="number" name="qty"
91                                         value="1" step="1" />

```

Browse.jspx

```
87                                     </div>
88                                     <input type="hidden" name="number"
value="${e.number }" />
89                                     <div>
90                                     <button type="submit"
name="select_item" value="Add To Cart" class="btn btn-success btn-md">Add To
Cart</button>
91                                     </div>
92                                 </form>
93                             </div>
94                         </div>
95                     </div>
96                 </c:forEach>
97             </div>
98         </div>
99     </div>
100 </c:if>
101
102     <div id="error">
103         <strong>${error }</strong>
104     </div>
105
106
107
108 </body>
109 </html>
110 </jsp:root>
```

Checkout.jspx

```
1 <?xml version="1.0" encoding="UTF-8" ?>
2 <jsp:root xmlns:jsp="http://java.sun.com/JSP/Page" version="2.0"
3     xmlns:c="http://java.sun.com/jsp/jstl/core"
4     xmlns:fmt="http://java.sun.com/jsp/jstl/fmt">
5 <jsp:directive.page contentType="text/html; charset=UTF-8"
6     pageEncoding="UTF-8" session="false"/>
7 <jsp:output doctype-root-element="html"
8     doctype-public="-//W3C//DTD XHTML 1.0 Transitional//EN"
9     doctype-system="http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd"
10    omit-xml-declaration="true" />
11 <html xmlns="http://www.w3.org/1999/xhtml" lang="en">
12 <head >
13     <title>Foods R US</title>
14 </head>
15 <body>
16 <jsp:include page="nav.jspx"/>
17 <c:if test="{not empty order}">
18     <h2>${order.account}, your order has been processed</h2>
19 </c:if>
20
21 <h1>Not Processed files</h1>
22 <c:if test="{not empty fileNames}">
23     <c:forEach var="e" items="{fileNames}">
24         <a target="_blank" href="Account?view=${e}">${e}</a><br/>
25     </c:forEach>
26 </c:if>
27
28 <h1>Processed Files</h1>
29 <c:if test="{not empty processedfileNames}">
30     <h1>Old Orders (Processed)</h1>
31     <c:forEach var="e" items="{processedfileNames}">
32         <a target="_blank" href="Account?view=${e}">${e}</a><br/>
33     </c:forEach>
34 </c:if>
35 </body>
36 </html>
37 </jsp:root>
38
39
40
41
42
43
44
45
46
47
48
```


Checkout.aspx

49
50
51

Login.jspx

```
1 <?xml version="1.0" encoding="UTF-8" ?>
2 <jsp:root xmlns:jsp="http://java.sun.com/JSP/Page" version="2.0"
3     xmlns:c="http://java.sun.com/jsp/jstl/core"
4     xmlns:fmt="http://java.sun.com/jsp/jstl/fmt">
5 <jsp:directive.page contentType="text/html; charset=UTF-8"
6     pageEncoding="UTF-8" session="false"/>
7 <jsp:output doctype-root-element="html"
8     doctype-public="-//W3C//DTD XHTML 1.0 Transitional//EN"
9     doctype-system="http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd"
10    omit-xml-declaration="true" />
11 <html xmlns="http://www.w3.org/1999/xhtml" lang="en">
12 <head >
13     <title>Foods R US</title>
14 </head>
15 <body>
16     <jsp:include page="nav.jspx"/>
17
18     <div>${error }</div>
19     <c:if test="${not empty user }">
20         <div>${name }, You have logged in</div>
21         <div><a href="Browse">Continue Shopping</a></div>
22         <div><a href="Add">View Cart and to Checkout</a></div>
23     </c:if>
24 </body>
25 </html>
26 </jsp:root>
```

nav.jspx

```
1 <?xml version="1.0" encoding="UTF-8" ?>
2 <jsp:root xmlns:jsp="http://java.sun.com/JSP/Page" version="2.0"
3   xmlns:c="http://java.sun.com/jsp/jstl/core"
4   xmlns:fmt="http://java.sun.com/jsp/jstl/fmt">
5   <jsp:directive.page contentType="text/html; charset=UTF-8"
6     pageEncoding="UTF-8" session="false" />
7   <jsp:output doctype-root-element="html"
8     doctype-public="-//W3C//DTD XHTML 1.0 Transitional//EN"
9     doctype-system="http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd"
10    omit-xml-declaration="true" />
11   <html xmlns="http://www.w3.org/1999/xhtml" lang="en">
12 <head>
13 <meta charset="utf-8"></meta>
14 <meta name="viewport" content="width=device-width, initial-scale=1"></meta>
15 <link rel="stylesheet"
16 href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/
17 bootstrap.min.css"></link>
18 <link rel="stylesheet"
19 href="https://use.fontawesome.com/releases/v5.5.0/css/all.css"
20 integrity="sha384-B4dIYHKNBt8Bc12p+WXckhzcICo0wtJAoU8YZTY5qE0Id1GSseTk6S +L3BIXeVIU"
21 crossorigin="anonymous"></link>
22 <link rel="stylesheet" href="res/Style.css"></link>
23 <title>Foods R US</title>
24 <body>
25   <nav class="navbar navbar-expand-sm bg-dark navbar-dark sticky-top">
26     <div class="container-fluid">
27       <div class="navbar-header">
28         <a class="navbar-brand" href="Browse">Foods R Us</a>
29       </div>
30
31       <button class="navbar-toggler" type="button" data-
32 toggle="collapse"
33         data-target="#navbarResponsive">
34         <span class="navbar-toggler-icon"></span>
35       </button>
36       <div class="collapse navbar-collapse" id="navbarResponsive">
37         <ul class="nav navbar-nav ml-auto">
38           <li>
39             <form class="form-inline" action="Browse"
40               method="post">
41               <input class="form-control mr-sm-2" type="string"
42                 placeholder="Enter the Item No."
43               name="number" />
44               <button class="btn btn-success" type="submit">
```

nav.jspx

```

name="select_item"
43                                     value="search">Search</button>
44                                     </form>
45                                     </li>
46
47                                     <li><a class="nav-link" href="Add">Cart</a></li>
48                                     <li><a class="nav-link" href="Account">Account</a></li>
49                                     <c:if test="${empty user}">
50                                     <li><a class="nav-link" href="Login?login=true">Log
In</a></li>
51                                     </c:if>
52                                     </ul>
53                                     </div>
54                                     </div>
55                                     </nav>
56
57     <div class="container-fluid">
58         <div class="row jumbotron">
59             <div class="col-xs-12 col-sm-12 col-md-9 col-lg-9 col-xl-10">
60                 <h1>Fast, Reliable, Convenient</h1>
61                 <p class="lead">Foods R Us provides a wide variety of gourmet
62                     grocery items at great prices. Our products are all of the
63                     highest
64                     and simply
65                     to your
66                     door!
67                 </p>
68                 <br></br>
69             </div>
70         </div>
71     </div>
72
73 </body>
74 </html>
75 </jsp:root>
76
77
78
79
80
81
82
83
84
85

```

nav.jspx

86
87
88

Style.css

```
1 @CHARSET "UTF-8";
2 @import url('https://fonts.googleapis.com/css?family=Lato|Pacifico|Righteous|Roboto+Condensed|
  Ubuntu|Muli');
3
4
5 html,
6 body {
7     color: #212121;
8     background-color: white;
9     margin: 0px 0px;
10    padding: 0px;
11    font-family: 'Lato', 'sans-serif';
12 }
13
14
15 .category-image{
16
17     height:200px;
18     width:300px;
19     margin-left:auto;
20     margin-right:auto;"
21 }
22
23 .form-row{
24     padding: 10px 0;
25     display: flex;
26 }
27
28 .form-row label{
29     padding-right: 10px;
30 }
31
32 .form-row input{
33     flex: 1px;
34 }
35
36 .btn-round {
37     margin-left: 5%;
38     width: 90%;
39     padding-left: 5%;
40     padding-right: 5%;
41     background: red;
42     color: white;
43     border: 1px solid white;
44     font-weight: bolder;
45     font-size: 20px;
46     border-radius: 24px;
47 }
```

```
48
49 .btn-round:hover {
50     border: 1px solid red;
51 }
52
53 .navbar.navbar-default {
54     margin-bottom: 0px !important;
55     background-color: black;
56     font-weight: 700;
57 }
58
59 .navbar-brand{
60     font-family: 'Pacifico';
61     font-size: 35px;
62
63 }
64
65 .jumbotron{
66
67     background-image: url('images/bg.jpg');
68     background-size: cover;
69     color:white;
70 }
71
72
73 .logo
74 {
75     height:100px;
76 }
77
78 .category-banner {
79     height: 20%;
80     width: 100%;
81     padding: 10% 0% 10% 5%;
82     color: white;
83     background: #3F51B5;
84     background-repeat: no-repeat;
85     background-size: 100%;
86     background-position: center;
87 }
88
89 .category-banner h1 {
90     font-size: 48px;
91     font-weight: bolder;
92     text-transform: uppercase;
93 }
94
95 .category {
```

```
96     padding: 16px;
97 }
98
99 .category-name {
100     font-weight: bold;
101     margin-bottom: 8px;
102 }
103
104 .category-description {
105     font-weight: light;
106 }
107
108 .filter {
109     background: white;
110     border: 1px solid #607D8B;
111     border-top: none;
112     padding: 16px;
113     margin-bottom: 16px;
114 }
115
116 .item {
117     margin-top: 4px;
118     margin-bottom: 4px;
119     border: 1px solid #ddd;
120     background: white;
121     padding: 16px;
122     padding-right: 0px;
123     transition: background ease-in-out 250ms;
124 }
125
126 .item:hover {
127     background: #eee;
128 }
129
130 .item img {
131     margin-top: 12px;
132     width: 32px;
133     height: 32px;
134     border-radius: 50%;
135 }
136
137 .item .item-name {
138     font-weight: bolder;
139     width: 118%;
140 }
141
142 .item .item-number {
143     font-size: 11px;
```



```
144     font-weight: lighter;  
145     color: gray;  
146 }  
147  
148 .cart-btn {  
149     margin-left: 105% !important;  
150 }  
151  
152 .cart {  
153     margin-top: 8px;  
154     color: gray;  
155     font-size: 24px;  
156 }  
157  
158 #cart-count {  
159     transition: all ease-in-out 500ms;  
160 }  
161  
162 #cart-count.cart-count-glow {  
163     background: #3F51B5;  
164     padding: 8px;  
165     font-size: 18px;  
166 }  
167  
168 .cart-item-name {  
169     font-weight: bolder;  
170     font-size: 20px;  
171 }  
172  
173 .cart-item-number {  
174     font-size: 14px;  
175     font-weight: lighter;  
176     color: gray;  
177     margin-bottom: 8px;  
178 }  
179  
180 .cart-item-price {  
181     font-size: 18px;  
182 }  
183  
184 .cart-item-amount {  
185     width: 32px;  
186 }
```