**Foods R US**

**An E-Commerce Website**

Designed and Developed By

**Wook Cho (cse: cho0708a)**

**Yang Bai (cse: baiyang1)**

**Jessica Sardinas Rodriguez (cse: invigil)**

*Submitted as the group project for the course*

*EECS4413 – Building E-Commerce Systems*

*Prof. H. Roumani – Fall 2018*

TABLE OF CONTENTS…

i. Design………………………………………………………………………………………1 - 2

Overview of data flow, architecture, key issues of design and phases of design progression. Describes any problems in design and any issues caused by previously flawed designs and their solutions.

ii. Implementation…………………………………………………………….……………….3 - 5

Overview of any implementation issues, testing methods and limitations with respect to the implementation and shortcomings due to implementation.

iii. The Team…………………………………………………………….……………………..6 - 7

Overview of scheduled meeting times, how work was kept up to date and consistent, lessons learned from the group project. Member names and signatures and member contribution is outlined in this section.

iv. Output Samples…………………………………………………………….…………………..8

Sample P/Os and their location on disk and sample completed P/O reports and their location on disk are shown here.

v. The Source Code…………………………………………………………….……………9-##

i. Servlets

ii. Analytics

iii. jspx

iv. Filters

v. Model

Design

System Architecture and Dataflow (highly simplified) see next page for details..

Provides a filter attribute

Apache Tomcat Server

links page to

Serves

Item

population

Links back to

User identification

Renames processed POs by adding a \* prefix

Serves

Main page

/efoods

OAuth Server

Poreport\_#.xml

List<String> POs

Add an item

List<String> POs

Changes quantites

PO Report Directory

PO Directory

Order checkout by button press

List<String>POs

Viewable PO

Generates new PO

Order confirmed by button press

Linked to by Catalog and Shopping Cart

Order cancelled by button press

Redirects to

Type class

Model Package:

Displays PO in HTML format

Type class

Generates new PO

The model package is responsible for all calculations in the B2C module, it returns all results by contacting the Engine class.

Dark squares are abstracted and not directly modified. Lightly colored ovals call the Model Package (the Engine class in particular)), while the darker ovals in the middleware rely on their own methods where Report Generator (main method) references the Unionizer (singleton).

Design Key Issues

1. How do we make sure the client is logged in when they checkout?
2. How do we keep a unique session login?
3. How do we maintain one shopping cart per client?
4. How do we make sure the client doesn’t add an invalid quantity to the shopping cart?
5. How can the client view P/Os without it effecting the middleware?
6. How can the middleware process a P/O only once without effecting the client’s ability to view a P/O?
7. How can we improve the search facility so it returns relevant searches?
8. What information should the P/O report actually have?
9. How do we allow the client to view items in a comfortable way?

Design Decisions

1. During checkout the user is immediately checked for an attribute called account, if that attribute is null then the user has never entered a login screen and authenticated so they are immediately asked to login.
2. We decided to add an attribute called “account” this account attribute is necessary to access any features that require being logged in such as viewing account info (displays a list of P/Os specific to the account) or checking out. The account attribute is given the value user which is obtained from an open authorization to a server that returns both “user” and “name”, this prevents the user from wrongly accessing things from other accounts for privacy reasons.
3. Each session has a specific attribute called order that is modified whenever the client changes their shopping cart in some way (updating it, adding an item, deleting an item, checking out), as such the shopping cart is maintained on a per session basis.
4. If the client tries to add an invalid entry or updates the cart with an invalid entry that is less than 1 or not numeric then an exception is thrown and an error below the shopping cart is displayed that states that the shopping cart could not be modified (either it could not be updated, or an item could not be added).
5. A P/O is displayed as long as it has the specified account name captured by a regular expression, as such it does not effect the middleware as long as the middleware does not remove the account name.
6. The middleware adds a prefix \* to the file name if the P/O has been processed, since the client can read P/Os like this still then it doesn’t effect B2C. The middleware does not process P/Os that have a prefix \* to the beginning, this cannot occur in unprocessed P/Os making this an effective marker.
7. The search facility was made more relevant by sorting the results in a lexicographically meaningful way, the entries that matched the search parameter closest (without assuming the search parameter is a prefix) were displayed first alongside any others that contained the parameter. This allows the user to quickly find all “sirloin” meats even if they do not know the brands.
8. The B2B requires knowing what to purchase and how much of something thus the Middleware’s P/O report only displays the item’s item code (since it is a unique label unlike the name of an item which could have subtyping) and the quantity of the item to be ordered. This was done to save space and traversal times by only keeping relevant information.
9. Clients can view their P/Os in their account info pages, this keeps a relevant page for clients to see all their past purchase orders that have not been removed yet (both unprocessed and processed P/Os), account info pages require authentication but offers a list of all P/Os the client has made. This was done to ensure that clients had privacy for their own accounts to a degree, the client should be able to keep secret how many P/Os they have and not see the P/Os of others without a direct link. This prevents accounts from being easily searchable and the invasion of privacy of other users. Of course, however direct URLs of P/Os are accessible without authentication and thus clients are able to access P/O orders even if they are not logged in with the account if they were given the URL by another client or bookmarked the URL, as per requirements.

Implementation

Implementation Key Issues

1. Should we use JSP or SPA?
2. How do we avoid unnecessary javascript?
3. How do we make sure the engine isn’t a superman class?
4. How can we manage servlets so they’re not doing any calculations and only handling requests and responses?
5. How do we handle errors?

Implementation Decisions

1. We decided to use JSP as most of us were more well trained in it and we valued the benefit of separation of concerns.
2. Often times it was hard to figure out how to implement something with only JSP, avoiding javascript was difficult at the beginning however code later completely migrated from javascript to coincide with the foundations of JSP. This was solved through more exposure and resolving problems amongst the group with communication (providing possible alternatives).
3. The engine by default plays a lot of roles, however adding too many methods makes it difficult for team members to read and understand and add their own methods if they had to introduce a new servlet, it also makes debugging code significantly harder. Many engine methods were migrated to different classes in the Model package in order to reduce this difficulty.
4. Due to the above reason migrating roles to the engine can be tricky and there was often, most notably the PODisplay servlet that made the blur between model, view and controller more apparent, due to the above problems it was easier in some cases to not fully migrate roles to other classes and blur the lines. This is thus both an issue with implementation and design (there was not a clear cut place for some servlets or features in our original plan and thus they had to be added haphazardly).
5. Errors on the server side were not shown to the user but quantity errors were given exceptions in their corresponding methods in the engine class and thus displayed to the user.

Testing

The project was tested without any testing units and was essentially done with a feature by feature testing practice. The team would continuously test when they added an additional feature the specific feature and then later would test the entire system again. Thus, the testing was only for functionality as apparent to the user, no special test cases were provided only manual testing was done.

Status

The Team

The work was not an official clear-cut division many parts of the project were done procedurally by everyone in the group. The group project was uploaded to a repository on GitHub and thus the group was allowed to update the project procedurally and fix past versions of the project that they might not have previously worked on. Whenever a teammate would upload a new version the teammate would inform the others by messaging them on the mobile messaging platform, What’s App, thus all group members were expected to update their version to the “latest” version as needed. As such, all group members have seen all parts of the implementation and are thus familiar enough with it to edit the code as all at one point have edited or fixed errors in previous versions of the project. However, it is the case that members did focus on certain areas even if other members would later come in to check and make necessary changes, this will be discussed in detail in after the next part.

Group projects can be difficult, to manage, as students we all have a lot of responsibilities outside the scope of one course and thus we do not all have equal time to put into the project and we work on the project at different times. Some members have jobs and a full course load and thus may not have a consistent schedule to work on the project. Thus, one lesson is that one needs to divide tasks as best they can and provide reasonable time frames for completion depending on the teammate, this is however difficult as finding the right balance for each person is nontrivial. Another lesson is that every groupmate has a different set of skills and style of coding, coding throughout the project should optimally follow a specific format so it is easy to read and understand, since we all come from different backgrounds it is a powerful lesson to have an arranged format, so all code can be in an easy to understand style. Our different set of skills should also be looked at and appropriately accounted for, for instance if one person has a lot of experience in CSS and user-friendly design it is a reasonable conclusion that they should have a major role in the design. If we were to start over, we might want to try and more closely define our roles and look into this in a deeper way, if everyone has the right balance then surely the project produced would be more impressive.

Contributions:

Wook Cho

Yang Bai

Jessica Sardinas Rodriguez (B2C-B2B Middleware, Search Functionality, Code Clean-Up and Bug Fixes in B2C, Majority of the Report)

I hereby attest to the accuracy of the information contained in "The Team" section of this Report.

Wook Cho \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Yang Bai \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Jessica Sardinas Rodriguez \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Output Samples

Sample P/O (saved in the Home directory in a folder called PO)

﻿<order id="1" submitted="Sun Nov 25 14:27:07 EST 2018">

<customer account="invigil">

<name>Jessica Sardinas Rodriguez</name>

</customer>

<items>

<item number="0905A044 ">

<name>Minced Rib Meat by VX </name>

<price>6.49</price>

<quantity>1</quantity>

<extended>6.49</extended>

</item>

</items>

<total>6.49</total>

<shipping>0.0</shipping>

<HST>0.8437</HST>

<grandTotal>7.3337</grandTotal>

</order>

Sample Report P/O (saved in the Home directory in a folder called POReport)

﻿<orders generated="Sat Nov 24 17:21:12 EST 2018" total="3">

<items>

<item number="0905A306 ">

<quantity>2</quantity>

</item>

<item number="0905A044 ">

<quantity>11</quantity>

</item>

<item number="0905A365 ">

<quantity>2</quantity>

</item>

</items>

</orders>

The Source Code