CS 1530 - SPRINT 2 DELIVERABLE

Coffee-specific Import and Inventory Tracking System

Karnbir Singh – Karnsingh214

Adam Rison – AdamRison

George Totolos – get13

Chris Meier – chrismeiercs

Due Date: June 24, 2015

**Accomplishments**

This sprint team communication improved. We decided that stand-ups were best held after each class. We were able to discuss the state of the project, any extra information we needed, and what each of us were doing. We held meetings on Slack to help each other through any issues and had an in-person meeting to map out the project. This gave us a better idea of what needed to be done and plan better.

There were some disagreements on how the system should be designed. This arose during our in-person when we discussed how the system should be designed. Some did not think that the pipeline approach would be the best way to go. During our discussion, each idea was heard and drawn out on the white board. By doing this, everyone was able to see that the pipeline approach was best and come to an agreement.

Since the last sprint, we decided to communication needed to be improved for this sprint. We held more meetings and more discussions about the project. We were able to get any questions we had answered faster, which allowed us to get a better understanding of the customer’s needs.

With this sprint, we found that we had more interaction with the customer. We felt more comfortable contacting the customer with any issues we had. This led to a better understanding of their needs and how to move forward with designing the system.

All of us are new to SpringMVC, so learning the ins and outs was difficult. Fguring out how the whole thing fit together was particularily frustrating. While some members have worked with similar frameworks, such as Django, Spring felt very bloated. We were able to figure it out by spending a lot of time with tutorials and documentation.

Testing is very new to us as well, which is where we struggled the most. We had trouble determining what tests were appropriate to write and how to find edge cases. The approach taken was writing a test for each method that was not a setter or a getter. Once we figured out how tests were written in the Spring framework, it became easier to figure out what should be tested and

Due to this being a web application and that our framework enforces this paradigm, we heavily focus on using the MVC pattern for development. This helps us separate our code which improves readability and usability. We also discussed how access to our back-end should be implemented. For now, we have settled on having the connection and updates as a singleton. This will help reduce cluttering from having multiples of this object around the application. It will also prevent any problems from having the same user initialized repeatedly throughout the application, as we are using a BaaS.

During this sprint, we were able to complete some of the “New Shipment” module of the application. The shipping calculator was implemented. During the next sprint, we will begin developing the remainder of the module. The inventory system will also be a major part of the next sprint. While we haven’t implemented a back-end to store data yet, we have found a library to help with that and that will be the highest priority for the next sprint.

**Completed User Stories**

<https://github.com/chrismeiercs/CS1530CoffeeImporter>

1) As a user, I want to keep track of the total shipping cost of an order by using an import calculator so that I know how much coffee to buy to get the best pricing (shipping cost decreases per bag as number of bags goes up).

2) As a vendor/user, I want the web app to calculate the specific shipping cost for importing a product into the country, separate from the rest of the shipping cost, so that I know exactly how much money I am spending out of pocket for the product.

3) As a vendor/user, I want a web app to create a calculated number representing the distributed shipping cost across the weight of the product that I buy so that I can apply it to the product that I sell in order to break even or turn a profit.

**User Story Decisions**

After talking with the customer more, we were able to get a better idea of what they wanted. We gathered that parts of the system where to fit together into a pipeline. The customer preferred to enter in all of the data by hand and have calculations performed and stored. We decided that is how the development of the application should be approached. We manually went through the process the customer gave us and determined the best way to perform tasks in a step-wise fashion. Our main focus this sprint was the shipment calculator. User stories were selected in a way that fit the pipeline-like process described to us by the customer. The three with the highest priority satisfied this.

**Testing**

Though test-driven development had a high time cost at the beginning, it did help with being able to structure the code properly to handle different situations. It mainly helped with determining edge cases. For example, in the WeightCalculator class, tests were written when developing the actual calculator portion. Since user needs to input the total weight of the shipment, and the default in the form is zero, a divide by zero error can occur if the user forgets to fill in the input.

System testing also helped with the process. Here, we were able to catch more defects than with unit testing. Going through the same process that the customer was helped point out some issues. When manually testing the cost calculator for the shipments, it was found that it is possible to enter in that the cost of the products was larger than the total cost of the shipment. After finding this, the method that calculates the shipping cost throws an exception letting the user know that this isn’t allowed. It was then determined that the user should see this message and be able to respond appropriately. When combined, system and unit testing greatly helped.

Before the end to this sprint, some team members looked over the code to find defects. We found that having others review your tests and code helped with finding more defects. Since this was the end of the sprint, they were added to the product backlog. The one issue was form validation. It was found that the user could enter in strings instead of dollar amounts in the calculation forms. When this happens, it is supposed to be caught through form validation. However, this was not implemented yet. Instead of alerting the user, Spring returns a 404 error along with a validation error for the form backing object. We decided to prioritize this next sprint since it was not caught until near the end of the sprint.