**COM 428 Object Oriented Design and Development (AE2)**

**Student Number and Name: 15071855 Justyn McKenna**

**Introduction**

This report will cover the design choices made in my diagrams and drawings and my reasoning behind changes and what I need to improve my application. The application referenced in this report can be found in the github repository JustynMckenna/COM528\_AE2 in the folder shoppingCartApplication. The diagrams and test plan mentioned can be found in the same repository in the location Documentation/ Diagram.

**Domain Model Drawing and Use Case Decisions**

The domain model drawing in the documentation folder covers the classes, attributes and methods in the application and the relationships between these. The decisions made on the model were greatly influenced by the existing attributes and classes inside the provided application for example the User, Invoice and shopping items were all already referenced within the existing application and thus the decision was made to use these attributes that were predefined in order to make the implementation process faster and easier. The decision behind the credit card attribute was based on the previous work in AE1 in which we had to design a credit card application. The attributes from the previous application were used in this application.

**Diagram Drawing Decisions**

When drawing up the use case, robustness and sequence diagrams, many design decisions were made. These decisions being how each user type will interact with the application, whether it be a Customer, Admin, Anonymous user or deactivated account. While drawing the robustness diagram, I decided to separate how the admin and the customer view the order list, with having the customer view it in page called “previous orders” while the admins orders in the admin drop down box called “view all orders”.

While drawing the sequence diagram no decision was made, however when drawing the use case diagram, the decision to keep the admin and customer accounts similar to some degree was made. This means that both the customer and the admin could use the site as if they were both customers, the original design had the admin account work only as an administrator and thus their ability to make transactions would not exist.

**Differences between Code and Design and why**

One of the key differences the codes and the design is the lack of limitations applied to the anonymous account. The anonymous account in the designs was unable to do anything until the customer or admin had signed in, however in the actual application, a user can create a shopping cart and even access the basket until it is hit with a restriction. This could easily have been changed by using the c:if test= "${sessionUser.userRole =='CUSTOMER'}" line of code, however the decision was not implemented as the lack of restriction was similar to that of actual working sites on the internet. Most sites don’t require you to sign in to create a basket, only requiring you to do so when you wish to pay, examples of this can be seen in popular shopping sites like eBay and amazon.

**Rationale for Test Strategy and Test Plan**

The test strategy used in my test plan was made with a simple goal in mind, to test as many or all the aspects of the user’s interaction with the application as possible. Testing each interaction and checking if it failed or not helps easily point out the errors within the code. The test plans were separated between the user roles, one plan for Customer and another for the Admin. The admin test plan covering the admin specific features like the modification of the users and catalogue while the customer test plan covered the transactions and the orders.

**Evaluation of Code**

With the implementation of the application, there were many aspects that could have been made differently or been implemented that were not and that is due to my poor management of time. The first part of the code I was unable to implement was the admin order page. This page exists in the application, but it currently is unable to gather all the users orders for the admin to see, this is because the MVCController lacked the ability to check all user orders and display them as I did not implement it.

Another part of the code that I was unable to implement was a working transaction between the bank and the shopping list. This was a major problem as this not only affected the transactions, but also effected the ordering as well as there was no way to test or create an order without a successful transaction taking place. This however was worked around with some code but the overall problem of a lack of transaction left the checkout page empty and unusable and left the orders list unable to be updated.

The final part of code that wasn’t able to be implemented was the modify catalogue option for the admin. In my attempt to implement it, the application lost the ability to update the shopping cart and basket and thus caused more problems without solving the issue I was trying to solve. I abandoned the code and reverted to a previous version before the code was implemented and then ran out of time to go back and implement a working modify catalogue feature.

Overall, most of the problems described above are due to my poor time management, there was no fault in the coding or received application, it was purely human error that caused all the problems.