THE UNIVERSITY OF SYDNEY

Assignment 2 (20%)
Simple E-Commerce System (Group)

School of Information Technologies
The University of Sydney

SYSTEM FUNCTIONAL DESCRIPTION

The simple E-Commerce System consists of three components: a **web front end** powered by Tomcat server, a **shipping** component running on a separate server within the same administrative domain and an **inventory** component running on third party premises. The inventory component provides basic product information such as product id, name, picture and price. Suppose our simple E-Commerce System specializes in selling photos, we use **flickr.com** to simulate an inventory component and communicate with it through flickr API.

The **web front end** allows registered users to search for product information and to add desirable products in a shopping cart. Once a user is ready to check out, she needs to provide a delivery address. The pending order will be sent to a shipping component to calculate the shipping fee based on delivery address as well as number of items in the shopping cart. An order will be accepted and stored in the database if the delivery address is valid. If the delivery address is invalid, a user can decide to discard the order to provide another address. A user can place multiple orders. She can view all her orders from the web front end. The web front end also allows administrators to view all orders placed by all users. An administrator can update an order's state. A newly placed order is always in "processing" state. It can be moved to "shipped" state by an administrator. A user is able to update an order that is in "processing" state. She may remove product, change the quantity of a product, or change the delivery address of an order that is in "processing" state. She cannot discard the order or remove everything from the order though.

The product information is fetched dynamically from **flickr.com** using **flickr.photos.search** API. For a given keyword, this API returns a large number of photos matching the keyword. You only need to display a few (for instance, 5~10) that can fit in one page. No paging through feature is required in this assignment. Each photo is a product that can be purchased in your E-Commerce System. Please use **photo_id** as the product id, **photo title** as the product title, and the list of tags as product description. The actual photo should be displayed using its flickr based url. Please compute price based on the number of tags a photo has. You can decide your own unit price per tag. You should also set a base price for photos with no tag. For instance, if the unit price is $1.00 and the base price is $2.00, the price of a photo with 5 tags would be $7.00. Photo url, photo_id, photo title and photo tags can be obtained from a single **flickr.photos.search** API call.

The **shipping component** is running on a separate server. It exposes a SOAP or RESTful API for shipping cost calculation. The API accepts a pending order and returns the shipping cost for it. It stores a list of city name and corresponding delivery cost. In addition there is a fixed per item shipping cost. For instance, if the per item shipping cost is $1.99 and the delivery cost to Sydney is $4.99, an order with 10 photos sent to Sydney would have a total shipping fee $24.89. The shipping component returns a shipping fee if the city name of delivery address is in its list. Or it returns an error message indicating the address is not valid. A user can either discard an invalid order or to enter a new delivery address. You can preload the per item shipping cost and a list of city name/delivery cost from a file. The shipping component always prints to console processing information. Below is an example of a series of processing information:

```
Shipping component starting …
receiving order 0001 with 10 items shipped to Sydney
the shipping fee for order 0001 is: $24.89
receiving order 0003 with 5 items shipped to Boston
invalid address
…
```

SPECIAL DEVELOPMENT REQUIREMENT

1. Group size is up to **2** students.
2. You can use **Servlet/JSP** or **Spring MVC** to develop the front end.
3. You are required to apply MVC in your code design.
4. You should to use **Ajax** to implement most of the front end features.
5. You do not need to provide registration mechanism. You can preload username/password/roles in a database or an xml file. Please make sure you have at least two regular users and one admin user in your security realm.

DELIVERIBLE AND SUBMISSION

1. Demo in week 13's lab.
   You should have your component running on different physical servers. The possible combinations are: lab-pc + own-laptop; AWS-cloud + lab-pc/own-laptop; ICT-teaching-server + lab-PC/own-laptop. If you decide to run one component on your own laptop, make sure it has good network connection. Using your mobile phone as personal hotspot may not be sufficient.
2. Submit your source code (including jave source, html/jsp source, CSS source and configuration files) to Blackboard Learn before midnight 2nd of June, 2015.
3. Submit a hard copy report (up to 6 pages) and a signed group assignment cover sheet in week 13's demo as well. In the report, please describe briefly the design of your front end system. You should describe the responsibilities of your controllers, models and views and how they interact with each other. You may use UML class diagrams to show main classes in the front end system.
4. Mark distribution: 15 marks for functional requirements, 5 marks for report.

Section I General Information	
Front end server ☐Lab PC ☐Laptop ☐AWS Shipping component server ☐Lab PC ☐AWS ☐Laptop ☐ICT teaching server	Group Name Student Name SID: Student Name SID:
Section II Preparations:	
1. Check the security realm	
Students show tutor where and how the system stores authentication information. Make sure there are at least two regular users and one admin user.	☐Database ☐File ☐Not proper configuration ☐No authentication mechanism Comments:
2. Start the shipping component on an server	
The shipping component should show some startup message on the console	☐Pass ☐ No Shipping Component ☐ No Console Message Others:
Section III Regular User Functions:	
1. Start the application and login as user	
The browser should show a query form and a link (or tab/button) for the user to check his orders	☐**Pass** ☐ No query form ☐ No link for order checking Others:
2. Check the user's order	
A message or page should show that the user currently has no orders	☐**Pass** ☐**Ajax call** ☐Showing orders not from this user Others:

3. Query for product, query term	
Type a query term to find photos from flickr and display them as products The photo itself, its title, tags should be displayed and the photo price be computed. An empty shopping cart should be displayed as well.	☐ **Pass** ☐ **Ajax call** ☐ Not showing photo image ☐ Missing title or tags ☐ Price is wrong Others:
4. Order/Remove a few products	
The shopping cart items should be updated accordingly; the total price should be updated as well. If possible, try to test cases such as removing a product that is not in the shopping cart.	☐ **Pass** ☐ **Ajax call** ☐ Cannot order product ☐ Cannot remove product from shopping cart ☐ Shopping cart item update is not correct ☐ Shopping cart total price update is not correct Others:
5. Display new list products using another query term	
The product catalogue should be updated with the shopping cart showing items previous products added. Add and remove a few products from this list.	☐ **Pass** ☐ **Ajax call** ☐ Not able to run another query ☐ Show an empty shopping cart ☐ Same error as in step 4 Others:
6. Place the order – filling in the address	
A new form should appear to allow users to type in the delivery address. Type in a valid address.	☐ **Pass** ☐ **Ajax call** ☐ No form to take delivery address Others:

7. Place the order – computing the shipping cost, finishing the order	
Click submit button to submit the order. Your shipping component should receive the order and print out progressive information. Your front end should display the order details including the shipping cost, and final cost.	☐ **Pass** ☐ No shipping component ☐ Nothing happens on the shipping component side ☐ The shipping costs do not match Others:
8. Check this user's order	
There should be a separate page showing this newly created order in "processing" state. Check that the order is stored in database as well.	☐ **Pass** ☐ Not able to see any order ☐ Order details are not correct ☐ Order is not stored in database Others:
9. Update the newly created order	
Click to show details of this order. There should be links or other editing mechanisms to allow removing, changing quantity of the order and changing delivery address of the order. The updated order should be sent to the shipping component to calculate the new shipping cost and saved in database.	☐ **Pass** ☐ Able to remove product ☐ Able to increase or decrease the quantity of a product ☐ Able to update the delivery address ☐ Not able to remove everything in the order Others:
10. Go back to the product catalogue page and make another order	
Type a new query term to get a list of products. The shopping cart should be cleared for the new order. In step 7, try an invalid address and then fix it. In step 8, the user should have two orders.	☐ **All Pass** ☐ Step 2-6 has same error(s) as those with the first order ☐ In step 7 shipping component accepts invalid address ☐ In Step 7 user is not allowed to fix the invalid address Others:

11. Log out	
Logout the current user. The browser should go back to the home page.	☐ **Pass** ☐ No proper logout mechanism ☐ Previous user's information still shows. Others:
12. Login as another user **repeat the steps (2-8) using a different query term**	
In step 8, the user should just see the newly created order.	☐ **All Pass** ☐ Step 2-7 has same error(s) as those from the previous user ☐ In step 8 the user is able to see all others placed so far including orders placed by another user. Others:
13. Make another order by repeating step 2-8	
In step 7, try an invalid address then allow user to discard the order.	☐ **All Pass** ☐ Step 2-7 has same error(s) as those from the previous order ☐ In step 7, shipping component accepts invalid address ☐ In step 7, user is not allowed to discard order ☐ Discarded order is stored in database Others:

Section III Admin function	
1. Log in as admin	
Admin user should only be able to view orders. He cannot view catalogue or place order.	☐ **Pass** ☐ Admin cannot do anything ☐ Admin can perform regular user functions Others:
2. View all orders	
There should be three orders from two different users.	☐ **Pass** ☐ Not able to view all orders ☐ Order information is not correct Others:
3. Update state of orders	
Update state of one of the first user's order to "shipped". The admin should not be able to other information except for the state.	☐ **Pass** ☐ Not able to change state ☐ Other order information can be updated as well Others:
4. User check the order states	
Logout the admin user and login as the first user to check his orders. There should be two orders, one with "shipped" state. The "shipped" order would not allow any further change.	☐ **Pass** ☐ Not able to logout admin ☐ Order information not correct ☐ "Shipped" order is still editable Others:

Section IV Implementation details:
Framework used: ☐ **Servlet/JSP** ☐ **Spring** ☐ **Other** **Proper Session Management** ☐ **Yes** ☐ **No** **Communicating with flickr** ☐ **REST** ☐ **SOAP** **Other** **Communicating with shipping component** ☐ **REST** ☐ **SOAP** **Other** **Other features**