THE UNIVERSITY OF SYDNEY

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

School of Information Technologies
The University of Sydney

## SYSTEM FUNCTIONAL DESCRIPTION

A simple E-Commerce System consists of three components: a **web front end** powered by Tomcat running on local host, 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 and price. Suppose our simple E-Commerce System specialises in selling photos. We use **flickr.com** to simulate an inventory component and communicate with it through flickr API.

The web front end allows **regular users** who have registered to search for product information and to add/remove 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. 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 regular users, and look at the details of each order such as order line items, quantity, shipping address and fee, and final cost. An administrator can update an order's status. A newly placed order is always in "processing" status. It can be moved to "shipped" or "delivered" status by an **administrator**.

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 keywords. You only need to display a few (for instance, 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 each photo has. Set a base price for photos with no tags. For instance, if a photo has 6 tags and each tag costs $1.00, with a base price of $3.00, this photo's price would be $9.00. 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 $0.99 and the delivery cost to Sydney is $7.95, an order with 10 photos sent to Sydney would have a total shipping fee of $17.85. 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. When invalid address error is returned, a user can either discard the order or enter a new delivery address. Discarding the order will reset the shopping cart and return user to the query form. 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 its current states. Important states include: starting, receiving order and finishing cost computation. It should print out order details when it receives an order and the total cost or invalid delivery address exception after it computes the cost.

**SPECIAL DEVELOPMENT REQUIREMENTS**

1. Group size is up to 3 students.
2. You can use Servlet/JSP or Struts to develop the front end.
3. You are required to apply MVC in your code design.
4. You need to use **Ajax** to implement the front end ordering feature to ensure that the shopping cart will be updated without refreshing the whole page each time you add/remove a product. You also need to use **Ajax** to implement administrator's functions: view order details and update order state. You can use Ajax to implement other front end functions as well.
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 three registered users: two regular users and one administrator, in your security realm.

**DELIVERABLES AND SUBMISSION**

1. Demo in week 13's lab.
    a. You can demo the assignment on your laptop or on lab PC. If you decide to demo on your laptop, make sure it has good network connection. Using your mobile phone as personal hotspot may not be sufficient.
    b. During the demo, you can deploy your shipping component and your front end on the **same physical machine**, but they have to be running on different servers. For instance, you may have two Tomcat servers running on the same machine, listening on different ports. One Tomcat server can host the front end, while the other hosts the shipping component.
2. Submit a zip file to Blackboard Learn **before 10 pm 3th of June, 2014** containing source code, including Java source, HTML/JSP, CSS, SQL and configuration files.
3. Submit a hard copy design document (2-4 pages report) and a signed group assignment cover sheet in week 13's demo as well. In the document, please describe briefly the design of your web front end component. You should describe the responsibilities of your controllers, models and views and how they interact with each other. You should also describe the interaction between the three main components of the system. You may use UML class diagrams to show main classes in the front end system and component diagrams for the main components.
4. Mark distribution: 15 marks for functional requirements, 5 marks for report.

| **Section I General Information** | |
|---|---|
| Demo Environment ☐Lab PC ☐Laptop | Group No<br>Name SID<br>Name SID<br>Name SID |
| **Section II Preparations** | |
| **1. Check the security realm and Database** | |
| Students show tutor where and how the system stores authentication information. Make sure there are at least three registered users: two regular users and one administrator. Check if a clean Database is used. | ☐UserDatabaseRealm ☐DataSourceRealm<br>☐Other Realms/Authentication Mechanism<br>☐No proper configuration (DO NOT proceed further)<br>☐No authentication mechanism (DO NOT proceed further)<br>☐A clean Database is used e.g. no order stored<br>☐No Database implementation |
| **2. Start the shipping component on an SIT server (e.g. pgrad.it.usyd.edu.au)** | |
| The shipping component should show some startup message e.g. "starting…" on the console | ☐Pass<br>☐No shipping component<br>☐No console message<br>Others: |
| **Section III Regular User Functions** | |
| **3. Start the application and login as regular user (the first user)** | |
| After login, the browser should show a query form and a link (or tab/button) for the user to check his orders | ☐**Pass** ☐No login mechanism<br>☐No query form ☐No link for order checking<br>Others: |
| **4. Check the user's order** | |
| A message or page should show that the user currently has no orders | ☐**Pass** ☐**Ajax call** ☐No message or page to show no order<br>☐Showing orders not from this user<br>Others: |

| | |
|---|---|
| | |
| **5. Query for product, query term** | |
| Type a query term to find photos from flickr and display them as products.<br>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**<br>☐ Not showing photo image ☐ Missing title or tags<br>☐ Price is wrong ☐ Shopping cart is not empty<br>Others: |
| **6. Add/Remove a few products** | |
| The shopping cart items should be updated accordingly; the total price should be updated as well.<br>If possible, try to test cases such as adding/removing multiple different products and removing a product that is not in the shopping cart. | ☐ **Pass** ☐ **Ajax call**<br>☐ Cannot add product into shopping cart<br>☐ Cannot remove product from shopping cart<br>☐ Shopping cart item update is not correct<br>☐ Shopping cart total price update is not correct<br>Others: |
| **7. Place the order – filling in a delivery address** | |
| Check out to place the order. A new form should appear to allow users to type in the delivery address. Type in a valid address. | ☐ **Pass** ☐ **Ajax call** ☐ No proper check out mechanism<br>☐ No form to take delivery address<br>Others: |
| **8. 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 to console a message about its current state and the order details. Print out another message when it computes the total shipping cost. Your front end should display the | ☐ **Pass** ☐ **Ajax call** ☐ No shipping component<br>☐ Nothing happens on the shipping component side<br>☐ No console message for either order details, shipping state or cost<br>☐ The shipping costs do not match<br>☐ Order details, shipping or final cost not displayed on the front end<br>☐ Final cost is incorrect<br>☐ Same error as this user's previous step: |

| | |
|---|---|
| order details including order line items, shipping cost and final cost. | Others: |
| **9. Check the newly created order** | |
| There should be a separate page showing this newly created order in "processing" state.<br>Check that the order is stored in Database as well. | ☐ **Pass** ☐ Not able to see any order<br>☐ Order details are not correct<br>☐ Order details are not stored in Database<br>Others: |
| **10. Log out** | |
| The browser should return to the login page after logout. | ☐ **Pass** ☐ No proper logout mechanism<br>☐ Previous user's information still shows<br>Others: |
| **11. Login as another regular user (the second user)**<br>**repeat the steps (3-10) using a different query term** | |
| In step 9, the user should just see the newly created order. | ☐ **All Pass**<br>☐ This user is able to see previous user's order in step 9<br>☐ Same error as the previous user's step:<br>Others: |
| **12. Login as the first regular user**<br>**repeat the steps (3-10) using a different query term** | |
| In step 4, this user should already have one previous order in "processing" state.<br>In step 7, 8 try an invalid address and then fix it.<br>In step 9, the user should see two orders both in "processing" state. | ☐ **All Pass** ☐ Shipping component accepts invalid address<br>☐ Shipping component does not print invalid address and return error<br>☐ Not able to fix the invalid address<br>☐ This user is not able to see her previous order in step 4 or 9<br>☐ This user is not able to see two orders including the new one<br>☐ Same error as this user's previous step: |

| | |
|---|---|
| | Others: |
| **13. Login as the second regular user**<br>**repeat the steps (3-10) using a different query term** | |
| In step 4, this user should already have one previous order in "processing" state.<br>In step 7, 8 try an invalid address and then discard the order.<br>In step 9, the user should see one previous order only. | ☐ **All Pass**<br>☐ The order with invalid address is stored in the system<br>☐ Shipping component accepts invalid address<br>☐ Shipping component does not print invalid address and return error<br>☐ Not able to discard the order or return to query form after discard<br>☐ Shopping cart is not reset after discard<br>☐ Not able to see the only one previous order from step 11<br>☐ Same error as this user's previous step:<br>Others: |
| **Section IV Admin Functions** | |
| **14. Restart the server and log in as administrator** | |
| Admin user should only be able to view all orders. He cannot view catalogue or place order. | ☐ **Pass**<br>☐ Admin cannot do anything<br>☐ Admin can perform regular user functions e.g. search catalogue<br>Others: |
| **15. View all orders and the details** | |
| There should be three orders in "processing" state from two different regular users. View the orders details. | ☐ **Pass** ☐ **Ajax call**<br>☐ Not able to view all three orders<br>☐ Not able to view order details<br>☐ Order information is not correct<br>Others: |

| **16. Update state of orders** | |
|---|---|
| Update state of one of the first regular user's orders to "shipped". Display confirmation on state update.<br>The admin should not be able to amend other order information except for the state. | ☐**Pass** ☐**Ajax call**<br>☐Not able to change state<br>☐No confirmation to show state has been updated<br>☐Other order information can be updated as well<br>Others: |
| **17. Check the updated order** | |
| Logout the admin user and login as the first regular user to check his orders. There should be two orders, one with the updated state | ☐**Pass**<br>☐Not able to logout admin<br>☐First regular user's order information is not correct<br>Others: |
| **Section V Implementation Details** | |
| **Framework used** ☐**Servlet/JSP** ☐**Struts**<br>**Proper session management** ☐**Yes** ☐**No**<br>**Proper Database model** ☐**Yes** ☐**No**<br>**Communicating with flickr**<br>☐**REST** ☐**SOAP** **Other**<br>**Communicating with shipping component**<br>☐**REST** ☐**SOAP** **Other**<br>**Other features** | |