# Internet Technology: Project report

Summer:2009

## TibShop

A generic on line shop with product search, shopping cart, order and user management backed by a database of products, users and orders.

Tenzin Dakpa
5751

# Table of Contents

# 1 Introduction.

My Project goal is to produce a simple e-commerce site backed by business logic for small scale retail market. The day to day operation has to be accomplished by non-technical staff in order to reduce running cost. The website presentation itself has to be easily modified according to the wish of the proprietor as he may wish to change his offerings according to market trends and seasonal demands. These requirement suggest a solution based on MVC -II with java beans, and JSP with EL and JSTL tags. As this was my first Dynamic website development my development pattern may not conform to any coding pattern so my apology for any confusions.

# 2 Requirements

Application scope:
A e-commerce solution for small mail-order type of business.

## 2.1 Assumptions:

### Inventory.

These mail-order type business have products that are frequently introduced and discontinued. At a time they usually stock a small variety depending on their specialization. For the purpose of this project we assume that there are about 200 unique stocked products.

### Users.

We assume that we are not interested in building customer base (since there is no such incentives for this kind of business). Our requirement is limited to getting sufficient information regarding where to deliver the product, to whom to charge and ability to integrate with online payment portals like Paypal.

### Invoice.

We need to be able to generate, track and record invoice/order as per Commercial Law.

### Business manager.

The operator of these business would prefer to have a low cost online operation. The technical skill requirement of the online operation should be low. Specifically, the website should be manageable by non-technical staffs without having to hire full time web masters.

## 2.2.1 User roles.

Anonymous. Anyone who is not logged in.

Customer. A person who is registered on this web site with enough information for us to process their payment and deliver their order.

Admin/staff/manager. A person who verifies and does the real world operation of processing payment, posting shipment and adding new products to the website

### 2.2.2 Use cases ( Or things you can do)

We start with what an anonymous person can do. The subsequent roles gets additional privileges.

#### 2.2.2.1 Basic use (for everyone)

- register as new user
- browse product catalog
- search database for product by name, price range, tags, product id.
- add/remove product into shopping cart.

#### 2.2.2.2 Customer use cases.

(In addition to all basic use)

- checkout cart into order page
- confirm/modify/cancel order
- change self details like passwords, address, etc.

#### 2.2.2.3 Admin/manager use cases

(In addition to all basic + customer use cases)

- add/remove products
- add/remove category {automated }
- Grant/upgrade a lower level user to manager status.
- view/search orders based on user id, product id, order id, order date, order status.
- Delete user accounts.{any }

## 3 Design

The web application is designed under the MVC II pattern. The database is base on typical retail business model.

### 3.1 Database:

PostGresql. ER diagram is shown in Appendix.

List of tables.

- user
- address
- product
- order
- line_order
- catalog

### 3.2Webapp Server

- Model. Standard java beans (package tibshop.business )
- View. JSP with EL and jstl tags. CSS for layout. (WEB-INF,Cart,etc )
- Controller. java servlets( package tibshop.cart , tibshop.catalog, tibshop.order, tibshop.util )
- Helper class. (database access package tibshop.data, utility package tibshop.util )
- Container. tomcat 6.0 with connection pooling and JNDI resource lookup

  Client javascript to validate forms.

## 3.3 Additional resource.

1. www.photobucket.com { image host for product picture}{working}
2. re-captcha verification server. {validation does not when run on unibz server}
   {disabled in the controller servlet deployed on unibz server}
3. Paypal {or any other payment processing service}
   {not implemented as business has to be officially verified}

# 4 A Tour of the application.

The website has a main index which serves as short introduction. The text can be replaced with special offers and discount as is typically done in e-commerce. This index page is dynamically generated with a menu that is specific to the role(privileged) of the current user.


A non-logged in user is presented with a side bar menu containing login / register options, and a list of product catalog as click able links. This list is dynamically generated from the database.
There is also a link for searching the product database under various search terms like the id, name, tags, and price range.

Although the user can browse any of the product catalog and add items to his shopping cart, he is not allowed to checkout the cart as we do not have any information about how to process his order / and payment. For this to work the user must be logged in.

The web application verifies that the user is not logged in and will say so. The user will have to log in or register to proceed further, A registration require the user to provide bill-to address and other relevant data. After a successful registration the checkout cart functionality is unlocked for the user. A user who is already registered only needs to login using his own credentials to unlock this functionality

The user can click on the confirm order button on the cart page to get an invoice for his order. This step also optionally provide the user to specify a different address to which the product can be delivered(commonly referred as ship-to address).

This function takes the data provided ( user ship-to address, product ids, quantity, etc) and generates a unique order id and insert all these relevant data into the database (specifically into order, line_order, and optionally the address). After this the user is free to browse and order again .

The admin/manager of this website is provided with additional functionality corresponding to use cases identified before in section 2.2.2.3. When this user with this privilege logs in a special admin menu box is displayed. This menu contains the website management functions like adding new products, reviewing order received, un/banning user or creating new admin user. He is provided with search functionality on orders to facilitate business management like finding out which product was sold most, who ordered certain products, history of all orders by certain user etc. These are available as view Order link, user management link and add product link.

The product category is dynamically generated when the server is started. It will be visible as links on the main web menu when the web app deploys. The new product will show up in the specific catalog only when the web app is redeployed.
( I have made this design because the products are best kept in memory as they are the most frequently used. They are retrieved from the database only the first time the web app is started and made available to all user threads as read only resources. The add product is expected to be used only infrequently compare to browse and searches, so any new product added are made available on web-app restart.)

# 5 Architecture

*Note:*
*Cases marked with * implies user has to be logged in.*
*Cases marked with * * implies user has to be logged in as admin/manager.*
*Unmarked cases is available to all user without needing to be logged in.*

## Module: User

1. Functions:login/register/change* user details
2. Beans involved: User & Address
3. controller involved: login_validate.java , register_user.java
4. views involved: login.jsp & user,jsp
5. helper class: tibshop.business.data.userdb , tibshop.business.data.addressdb ,re-captcha library
6. figure 1.1 user login, figure 1.2 user registration

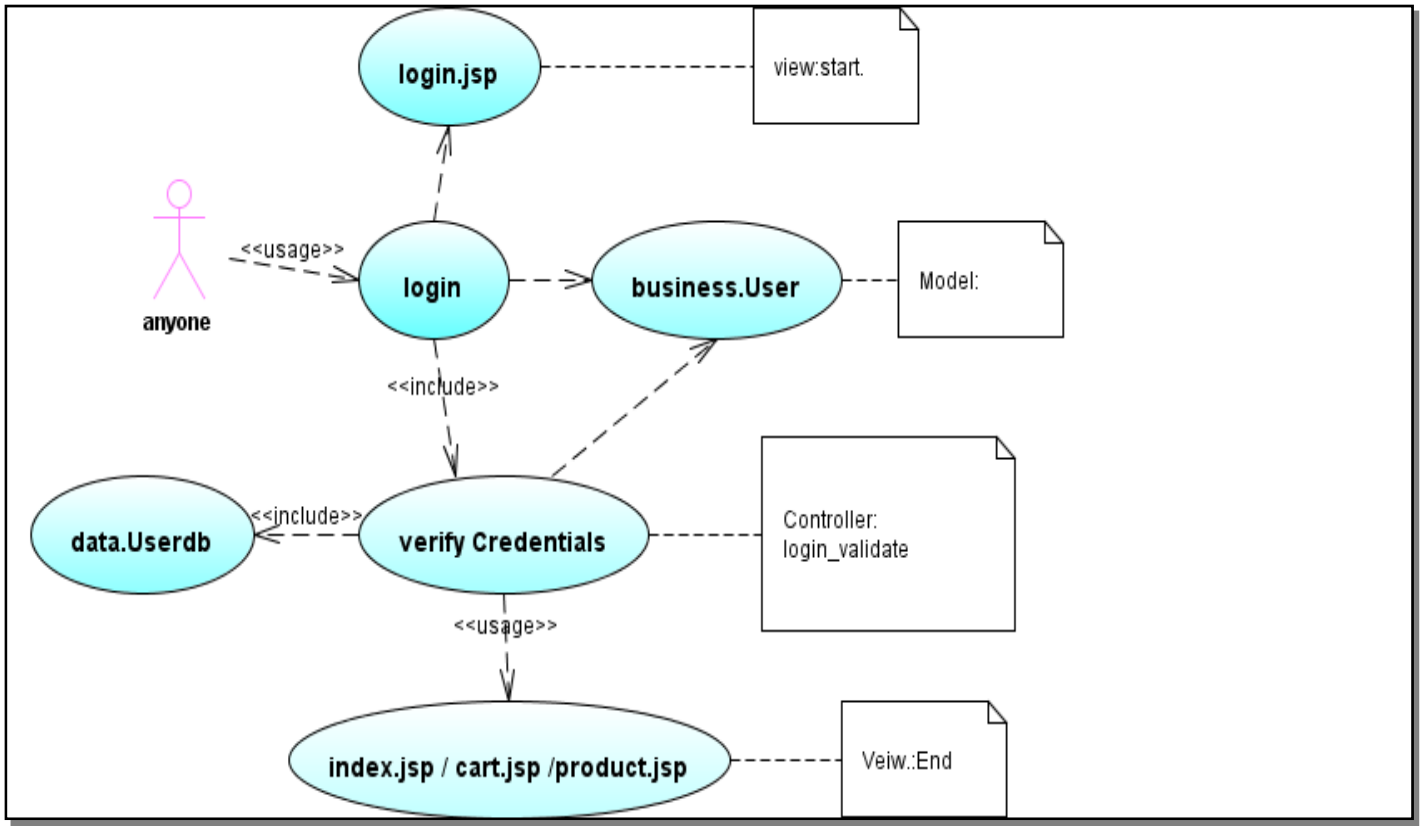


figure 1.1 User Login

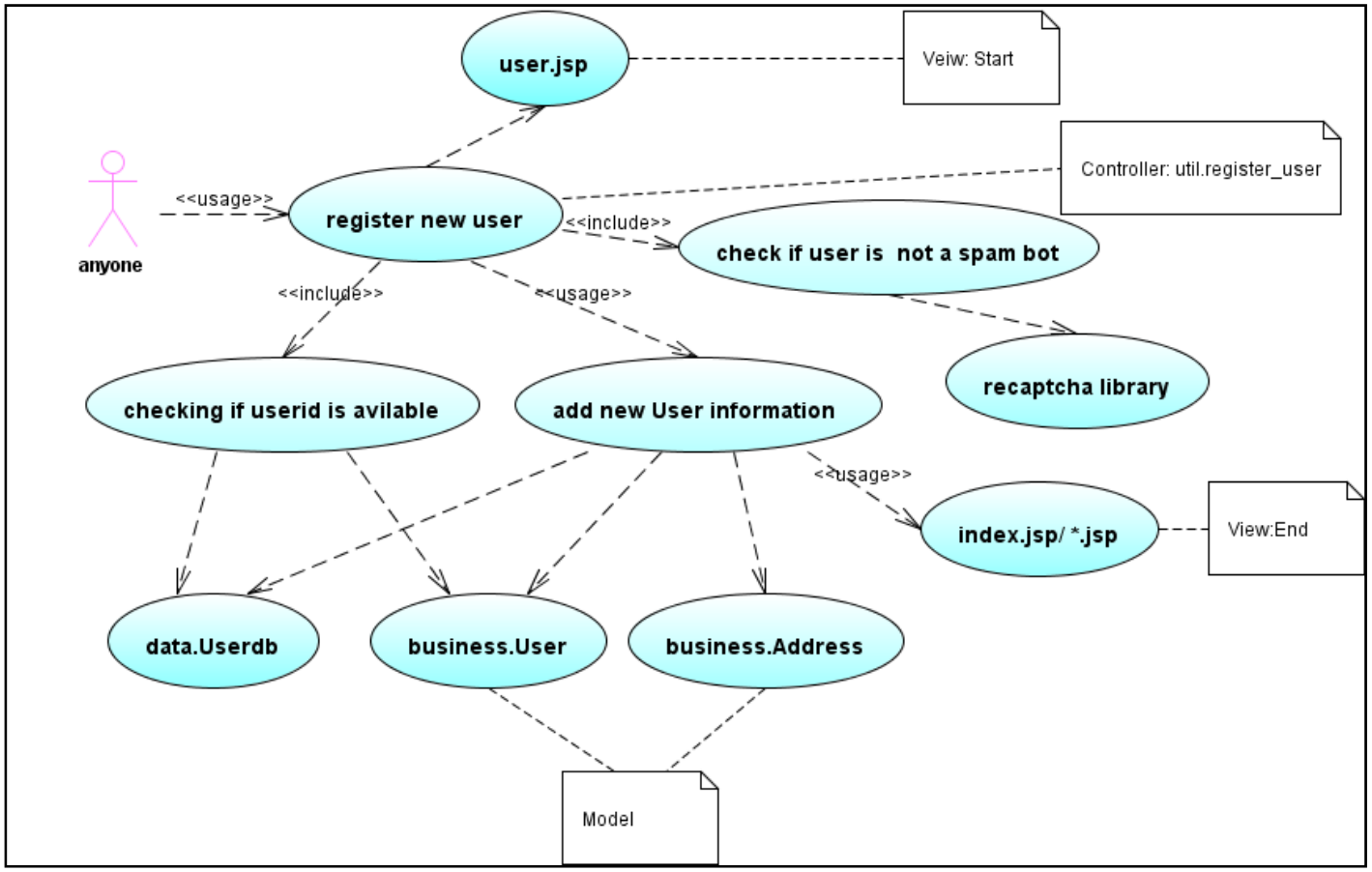


figure 1.2 User registration

## Module: Product

1. Functions:view, search ,add* products and ( browse by ) catalog.
2. Beans involved: Product & catalog
3. controller involved:(Package tibshop.util) index2.java, addProductServlet.java, (Package tibshop.catalog) DisplayProduct.java, advance_search_product.java
4. views involved: menu.jsp , product.jsp, searchProduct.jsp, addProduct.jsp
5. helper class: tibshop.util.searchHelper.java ,tibshop.business.data.productdb, tibshop.business.data.catalogdb

figure 2.1 view/search Product figure 2.2Add product

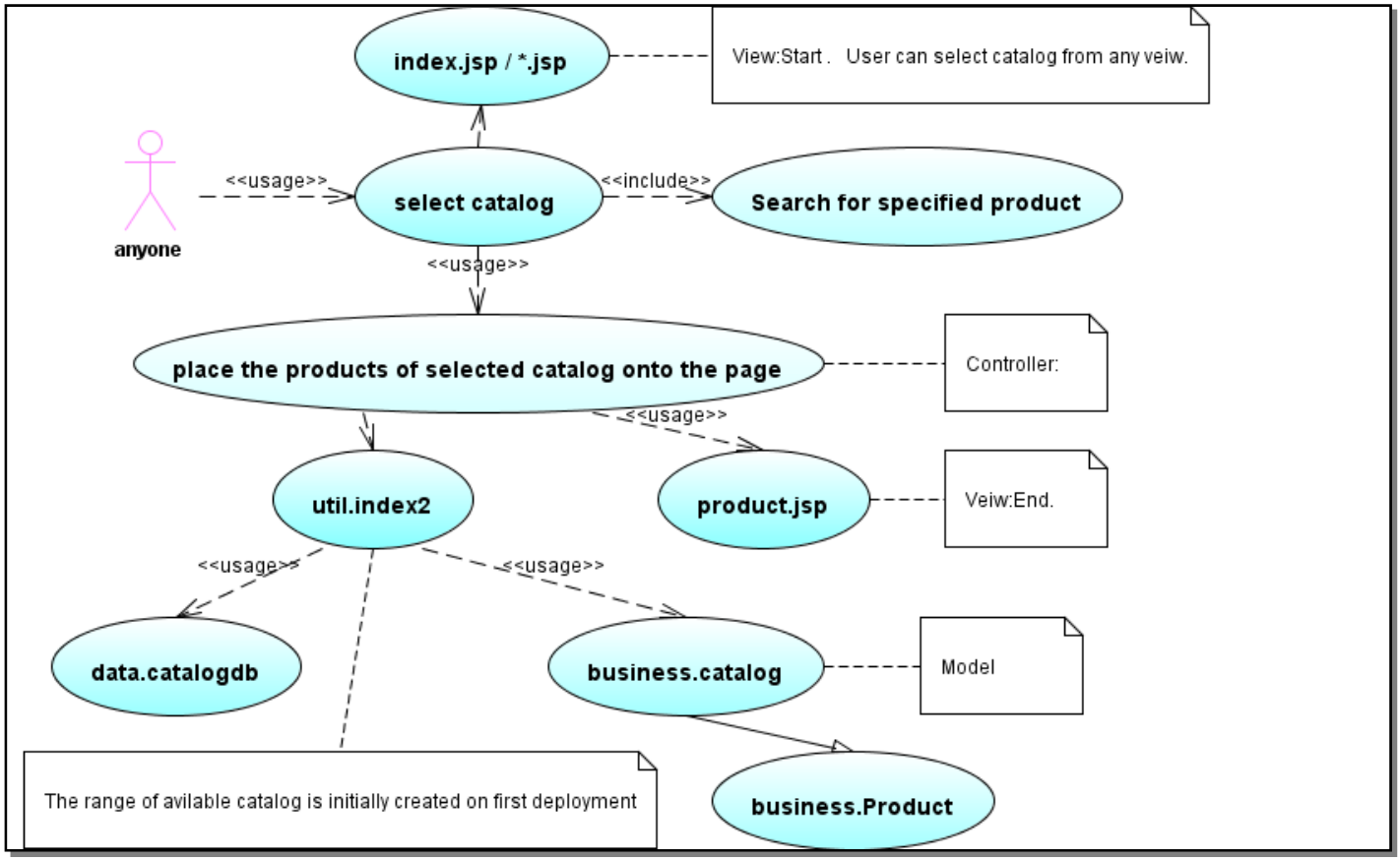


figure 2.1 view/search Product

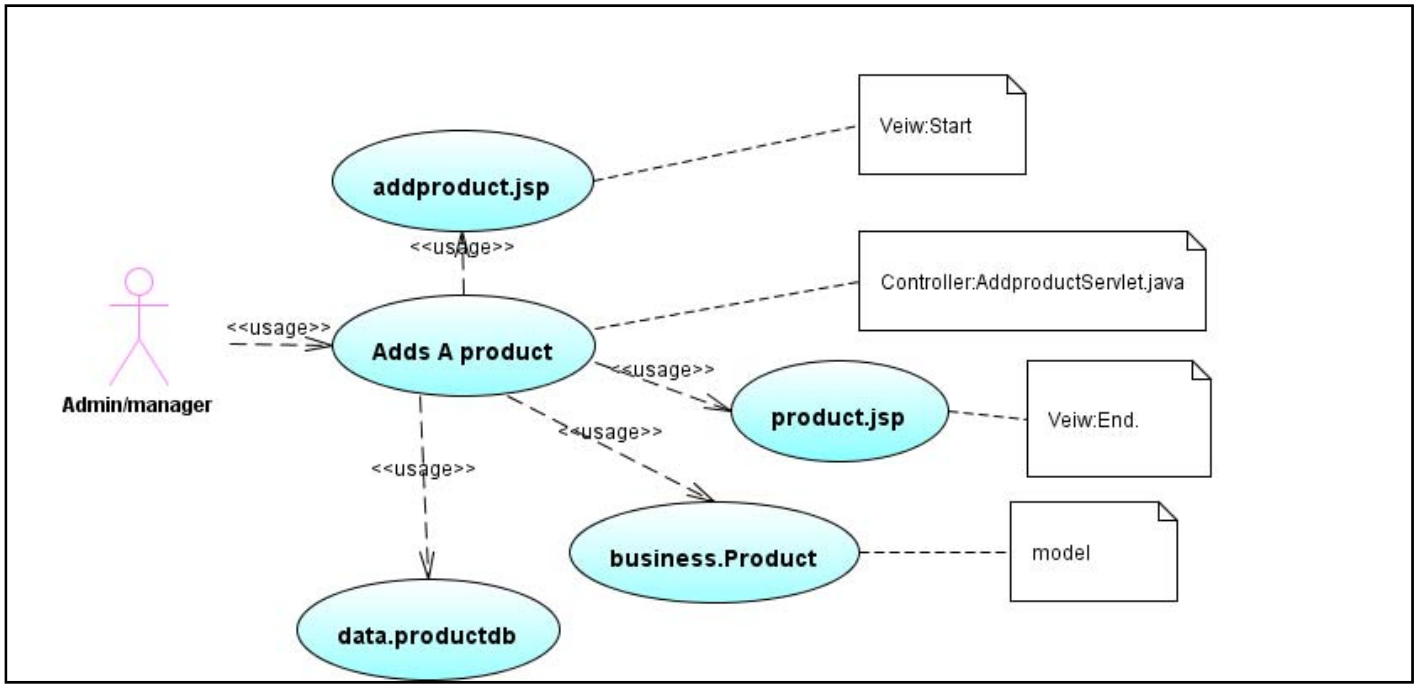


Figure 2.2 add Product

## Module: Cart

1. Functions: add / remove products to cart, view cart.
2. beans involved: Product, catalog & Cart
3. controller involved: tibshop.cart.DisplayCartServlet.java
4. views involved: product.jsp, cart.jsp
5. helper class: none.

Figure 3.1 add product to cart, Figure 3.2 remove product from cart / display cart

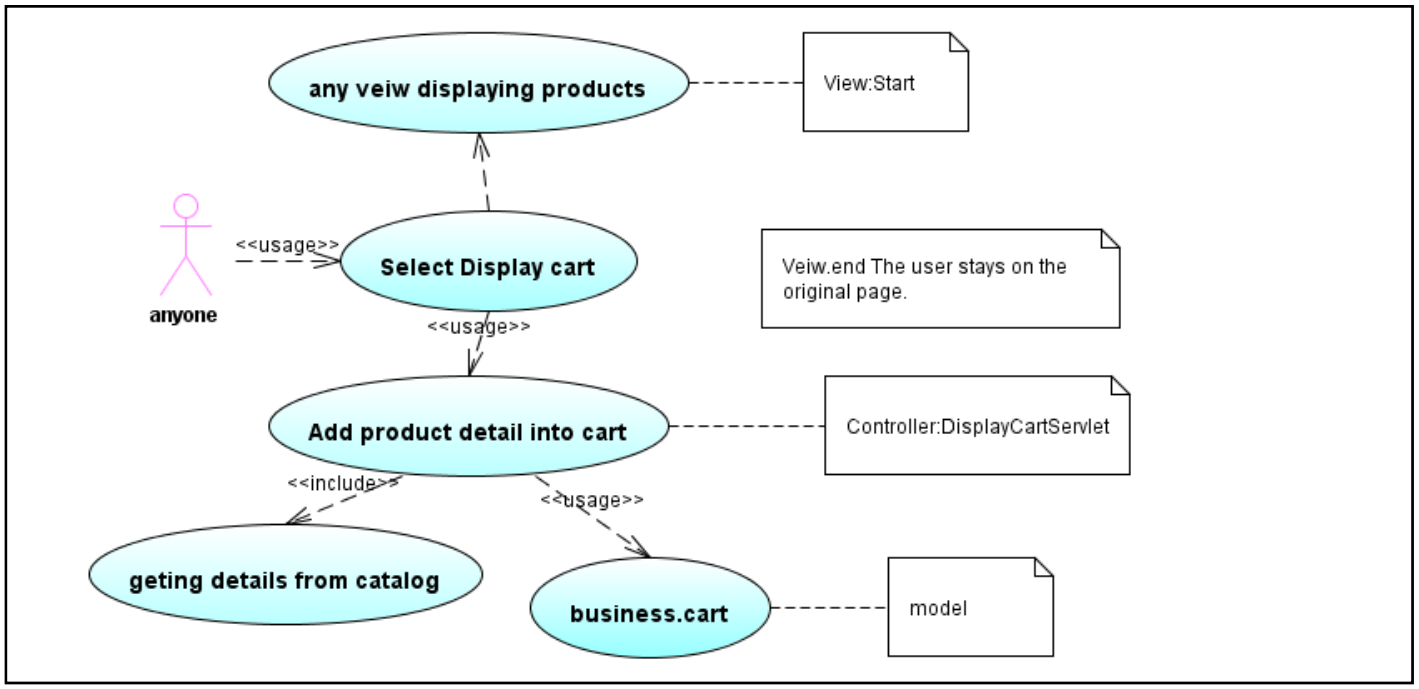


Figure 3.1 add product to cart

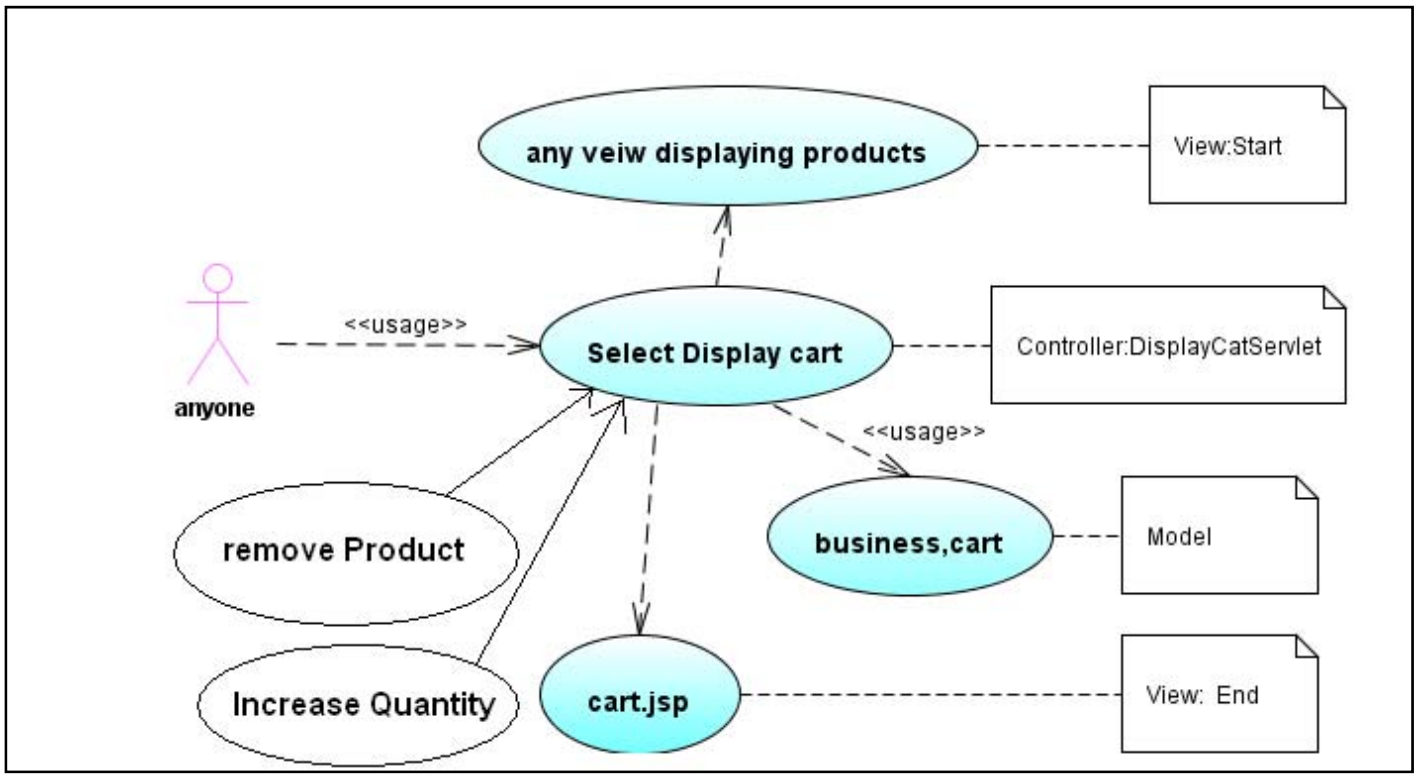


Figure 3.2 remove product from cart / display cart

# Module: Order

1. Functions:confirm order*, search** order
2. beans involved:User, Address, Product, Cart,Order &Line_order
3. controller involved: tibshop.cart.place_Order.java, tibshop.order.DisplayOrder
4. views involved: cart.jsp, orderDetails.jsp, OrderVeiw.jsp
5. helper class: tibshop.data.* ;

Figure 4.1 confirm order from cart page Figure 4.2 Order management (search and display )

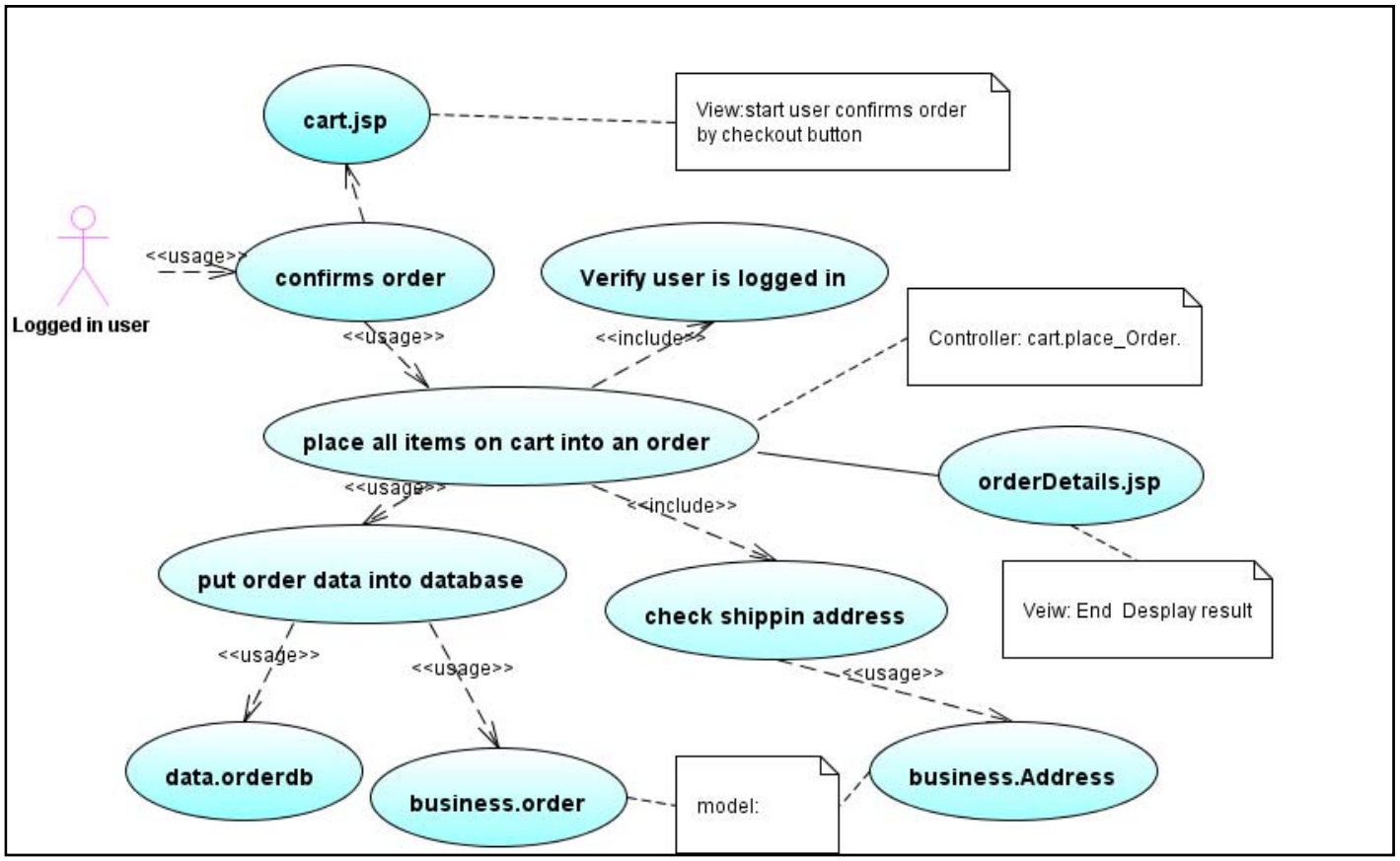


Figure 4.1 confirm order from cart page

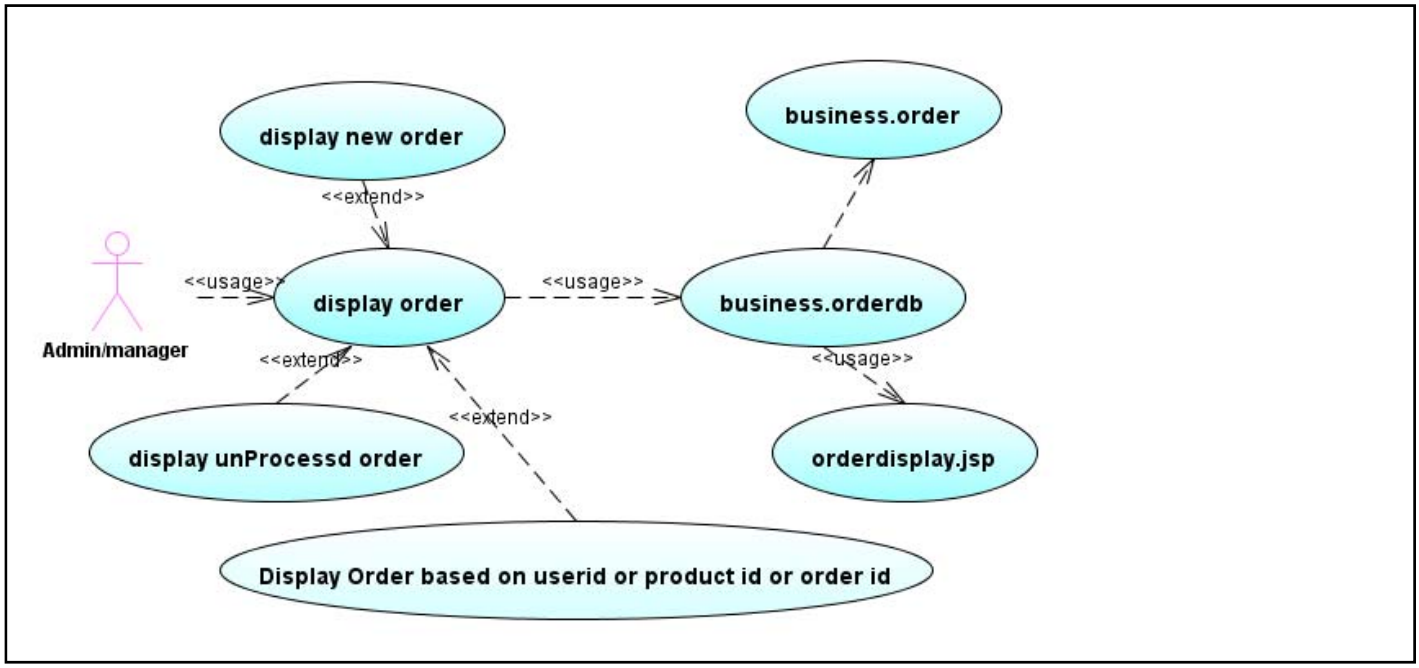


Figure 4.2 Order management (search and display)

## Module: Admin

1. Function: ban user**/ make user admin**
2. beans involved: User
3. controller involved: tibshop.util.ManageUser.java
4. views involved: manageUser.jsp
5. helper class: tibshop.data.Userdb

Figure 5 manage user

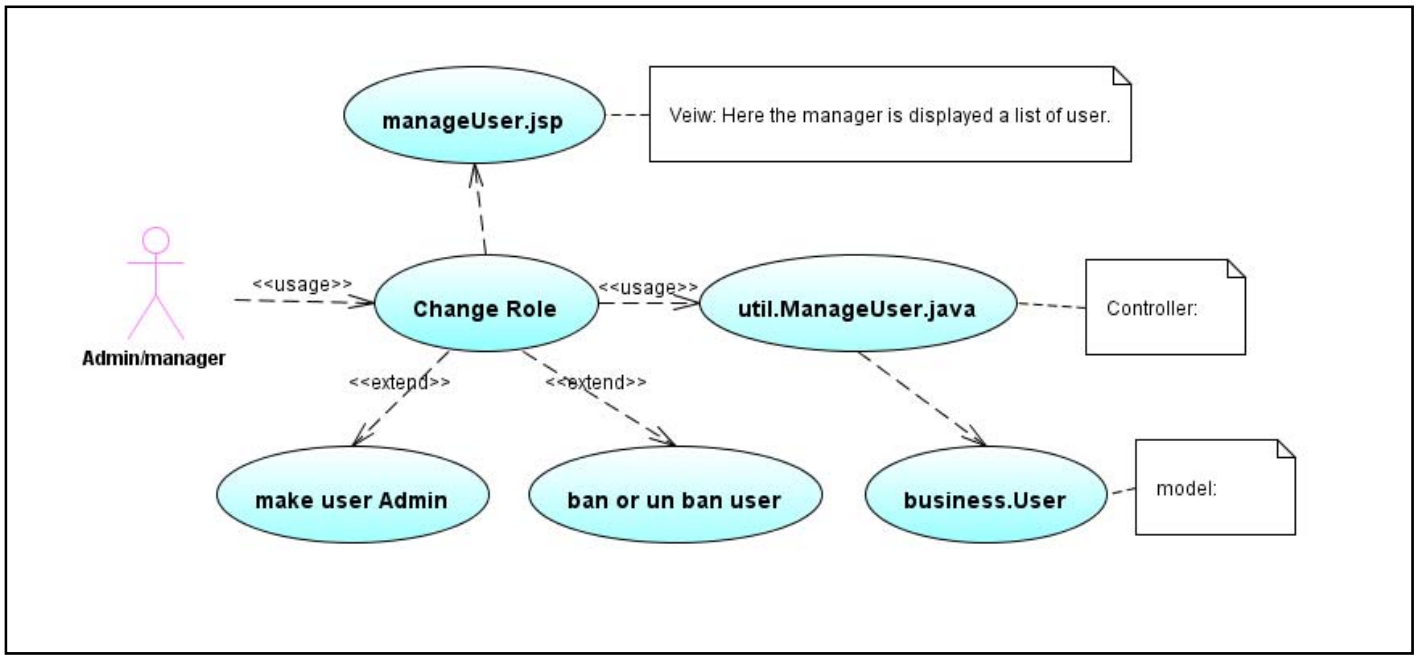


Figure 5 manage user

# 6 Details of Main Modules.

All the module follow the standard pattern of getting information from the browser, a servlet processing the information, and calling additional helper class (if necessary), generating some appropriate response and forwarding to a jsp view which formats and display the response.

## 6.1 Product.

The Product bean is stored in the database with all its details. Upon web site deployment, the main index controller loads and sort all the product into their specified category (this information is stored in the catalog table ).The controller makes this products available as catalogs with specific names. These catalogs are just a collection of product beans. The names of these catalogs are dynamically displayed and linked in the side menu of the website. When a user clicks a particular catalog name on the menu, the controller DisplayProduct.java identifies that catalog and send it to product.jsp. This view retrieves the catalog passed to itself and dynamically generate all the html code necessary to display all the products inside that catalog. The product.jsp also provide a mechanism to ask the server to record the customer's request of adding items to a shopping cart. The view uses EL to get all the data corresponding to the products. It user JSTL core tag to dynamically access the collection of product from the catalog.

On requesting a search for a product, the controller advance_search_product.java is invoked. It process the search parameters and with the help of searchHelper.java performs an in-memory search for the result. The result is saved as a catalog since it can contain more than one products. It is not stored in the database, but directly forwarded to the view product.jsp from which a user can add it to his cart as usual.

## 6.2 Order.

This module serves two of the user role under different use case. For a normal user it allows the confirmation of order along with additionally providing a shipping address. For a manager it allows performing search based on parameters like the date of order, whether the order has been processed, The id of the order, the product id or user id. Additionally he can make those order which has not been processed into processed order.

The Order bean contains a list of products along with their quantity. Additionally it has a status indicator, a date of order, etc. When a user confirms his order from the cart page, the list of product and their quantity in inserted into the database table Line_order and Order. In this process a order id is generated which is unique. If a different ship-to address is provided, then that order is linked to the new address by inserting the new address in the address database. When all these are successfully done, the user is displayed a summary of this order through the orderDetails.jsp. The controller involved with this side of order is tibshop.cart.place_Order.java

The Manager can specify different parameters for searching and sorting the orders. These are done through the orderVeiw.jsp. The search parameters are collected and parsed by tibshop.order.DisplayOrder. I used a byte field to provide a single access method from the controller to the helper class. This will help me to extend the search helper class without having to change anything in the controller. The search parameters are: date of order, order status, id based search on order, user, and product. The various combination of these parameters is difficult to satisfy completely. So, I have restricted the search to having only one id based search parameter at a time. Thus in the helper class I have implemented the 6 most common and useful combination of search. The other search

combinations will simply print out that such search pattern has not been implemented.
All these search are SQL search and performed using PreparedStatement objects.
Once the search is performed the result is provided to the controller in a form of ArrayList.
The controller will forward this collection to the orderveiw.jsp ( the same page where we started this search)
The view will dynamically display the order details like who made the order, where it was supposed to be shipped and all the items and quantity etc. Additionally if the order is not processed the manager is provided with a button that makes the order a processed order. see figure 6.

The other module follows the same pattern.

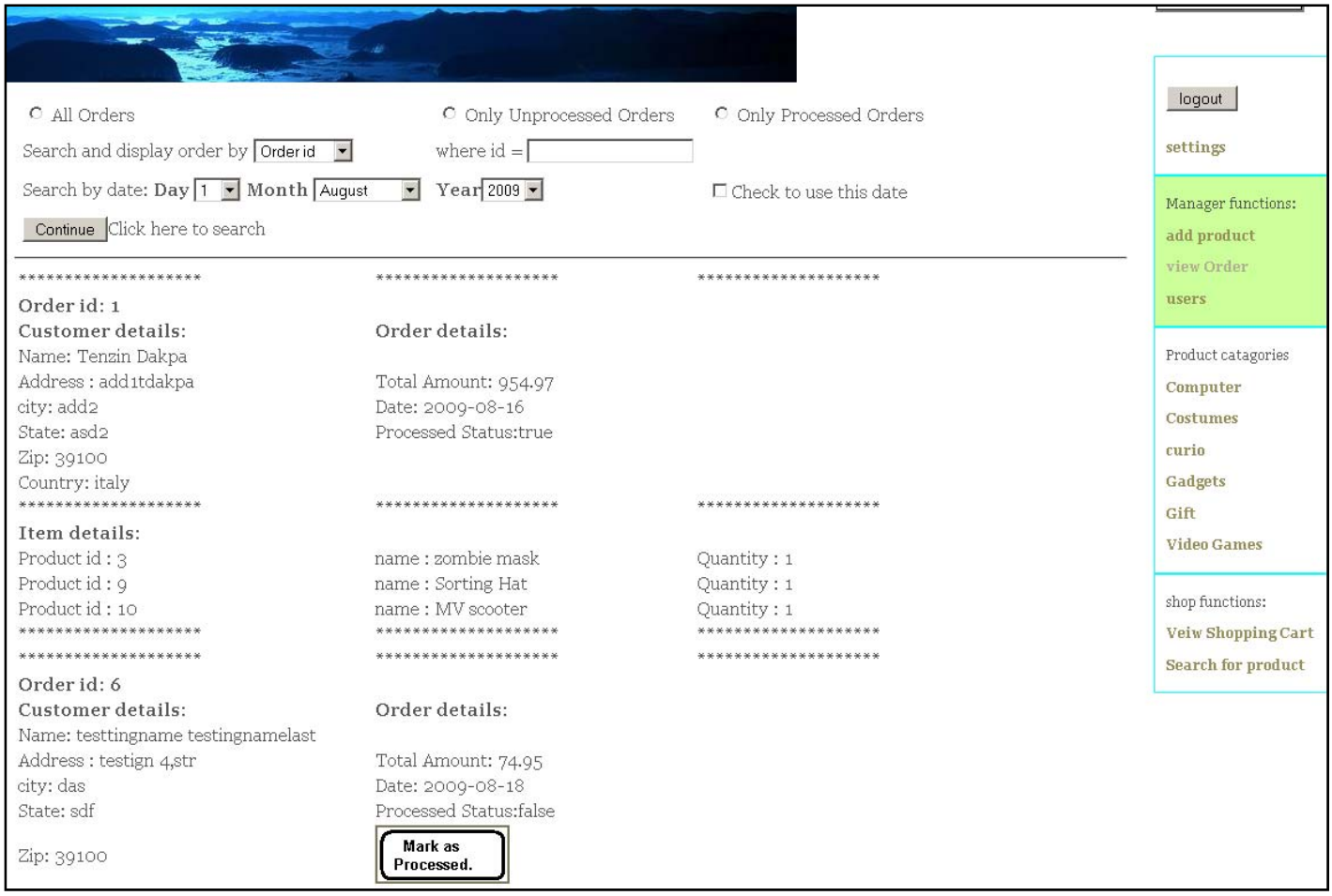


Figure 6 A Screen capture of Order management view

# 7 Design justification.(FAQ)

1. How is user identified and sessions maintained?
2. Why is the product catalog maintained in memory?
3. Where do you record the credit card numbers?
4. How does the re-captcha validation works ?
5. How do user pay for their order?
6. Why is some search performed with sql while others are done in memory ?
7. Where can I add comments?
8. Why does the website have such weird design?

Ans.1 The user is identified by using a standard cookie with jssesion id. We do not record or store anything in this cookie other then the session id. The session expires in 30 min of inactivity. A user who logs in to the web application is associated with this cookie on the browser side and with a HttpSession object matching this cookie on the server side. This Session object stores the users settings and role, any items put in the cart and any search performed for him. On logout the session object is invalidated and the cookie with that jsession id can not be used.

Ans2.

The Product is the main item used in this website. It is largely immutable as only occasionally new products are introduced. The customer and non-logged in user needs to obtain the product details as fast as they can while browsing the catalogs. And also when performing search with regard to their specific search parameters like product name, tags, product id, price range etc. Therefore, It is more memory efficient to keep all the products as data available to all user thread, and not have to obtain it from the database for each user. Since it is already in memory , all the search is performed faster leading to better response time.

Ans 3

Currently I am not recording or validating the credit card number at all. The PayPal SDK recommend not to store these if you are not using an encrypted connection like SSL. Since it is not possible for me to get these security certificates I chose not to read implement them. The Paypal SDK recommends immediate redirection onto their portal for validation of the payment after which they will redirect back to my Thank you page while simultaneously sending me a token containing payment details and confirmation. As such it is operationally cheaper and more secure to not implement this part of the web-app.

Ans 4

The recaptcha validation on the server side needs outbound access to the server api-verify.recaptcha.net through port 80. Since I do not have sufficient privilege on the university net to open these ports through firewalls, I have simply disable this service in the controller. On my local machine it send the words typed by the user to that address with a token. Their server validates the user submitted word and returns the token with a message saying passed or failed. On my Server controller I just check this token and if it contains the passed parameter I process to the user registration. Otherwise I display the error message asking the user to try the recaptcha again.

Ans 5
The payment for the orders has to be handled by a third party service like paypal or other banking service. It is not possible for me to attempt this because of security consideration and my inexperience in this matter. However, I have tried my best to make this web application compatible with paypal sdk guidelines. It should not be too difficult to integrate this web application with paypal or other online payment processing service.

Ans 6
The most frequently used objects is maintained ( Explicitly. the products in catalogs) in memory. So search involving these items are done in memory as it is lot more faster. Other items are saved in the database, so searches involving them needs to be performed on the database.

Ans 7
This web application aims to be a online shop with minimal maintenance overhead. Instead of posting comments the user can submit orders.

Ans 8
I regret that I have no design/art skills. However I have made the front end in pure JSP (read html) with EL and tags. There is no scripting (except for some javascript to validate user input) so any web designer can quickly and efficiently change the presentation into what they want.

# 8 Summary

This web application has its limitation due to its primary scope being small business. It is run able by regular non technical people requiring technical assistance only when doing database backup and redeployments. As it was developed under MVC II, it is modular enough to allow extension to its business logics and other areas of the application. The code itself shows some irregularity in style as I started this application as a complete novice and learned a lot while doing this project. I faced typical problems like Installation issues with the IDE, class library and integration with Tomcat. One java problem was that I get an unchecked cast when I try to retrieve a collection of bean stored in the session. It is an ArrayList of beans. I had no solution but to add a compiler suppress flag to it. Another difficulty was that there were too many different technology involved in this project which made this project very hard to manage. To learn to use jstl tags, and EL syntax on top of javascript and CSS style was quite confusing at times.

# Appendix.

## Reference

Software used:
Operating system: Windows XP sp3
IDE: Netbeans 6.5
java: SDK 1.6
Tomcat: version 6.0
PostGreSql: Version: 8.3.7
CSS tool: Style Master 4.6 demo
SQL utility: PGAdmin III
browser: Firefox 3.1, IE 8, google chrome.
Library: recaptcha4j-0.0.7.zip java library for re-captcha service, JSTL core tag library

Books reference:
Head First Servlets & JSP Second Edition, O`Reilly
Murach's Java Servlets and JSP, 2nd Edition.
The Zen of CSS Design: Visual Enlightenment for the Web

Web reference:
www.w3schools.com For HTML , javascript reference
www.sun.com For guidance on netbean ide , java api for servlets , jsp, EL ,Tags etc
http://javascript.internet.com/ For the calender javascript.
http://www.csszengarden.com
www.databaseanswers.org For the Database design base for e-commerce

Project Code svn : Anonymous read access
http://cetravex.svn.beanstalkapp.com/tibshop/

Project war file overview.

- Cart
  - Cart.jsp
  - orderDetails.jsp
- Images
- Meta-inf
- Web_-inf
  - classes
  - jspf
    - header.jsp
    - footer.jspf
    - logout.jspf
  - Lib
    - jstl.jar
    - recaptcha4j-0.0.7.jar
    - standard.jar
  - Order
    - OrderVeiw.jsp
  - product
    - addproduct.jsp
    - product.jsp
    - searchProduct.jsp
- User
  - manageUser.jsp
- tibShop.css
- favicon.ico
- index.jsp
- login.jsp
- menu.jsp
- user.jsp

Note: The Database is based on this Model with suitable reductions made to fit our goal.