COMP 3512 Assignment #2

Due Midnight-ish Monday Nov 14

Version 1.1, Oct 27 2016 [New changes in yellow]

Overview

You can work in groups of twos/three for this assignment. It is also possible to work individually, but I do strongly discourage it; please talk to me about this if you are planning on working by yourself. This assignment will allow you to apply the concepts and technologies covered in class to a "real" project situation. In this project, you will expand your first assignment.

If working in a group, each member needs to take responsibility for and complete an appropriate amount of the project work. **Be sure to consult the instructor at least one week prior to the due date if your group is experiencing serious problems in this regard.**

Submitting

Put your assignment in a folder named assign2_yourlogin (e.g., assign2_fsmi9876). Only one person in the group needs to submit the assignment. You must also make this site viewable online. I would recommend using Cloud9 or CodeAnywhere (though some you might decide to try something more adventurous such as Heroku or some other web hosting environment). You must then share the URL with me in some manner. If using Cloud9, for instance, you can share your assignment Workspace with me. If using some other hosting option, send me an email with a link that allows me to view your assignment.

Put all resources used by your assignment into this folder. Copy this folder to the submit drive in B215 and to the normal university-wide submit drive. You will lose marks if you do not follow these submission instructions.

Grading

The grade for this assignment will be broken down as follows:

Usability and visual design	20%
Program design **and** documentation	15%
Features	65%

USE CASE NAME:	**Assignment 1 Functionality**
DESCRIPTION:	Expected base functionality
1.	Your system must have ALL the functionality from assignment 1.

USE CASE NAME:	**Program Design**
DESCRIPTION:	Class Approach
1.	Make use of a class-based infrastructure for this assignment. Chapter 14 provides code examples for implementing this infrastructure.

USE CASE NAME:	**Navigation**
DESCRIPTION:	The web site's navigation system.
1.	To your existing navigation in the top row, add a Favorites and a Cart link. These will take the user to **View Favorites** and **View Cart** cases. For each of these links, the label for the link should include a count of the items in the respective lists (e.g., if there are three items in the Favorites list, the link label would be: Favorites (3). Please use the SemanticUI Label class for these numbers. Modify the Browse drop-down menu to include correct Browse Subjects, Browse Artists, and Browse Galleries (see below) links.

USE CASE NAME:	**About Us**
DESCRIPTION:	Display information about this site
1.	The system will display information about this site. Be sure to mention that this site is hypothetical and was created as a term project for COMP 3512 at Mount Royal University taught by Randy Connolly. As well, be sure to list the group member names and roughly what parts of the project each member implemented. If you used any external resources, be sure to cite (or link to) them here.

USE CASE NAME:	**Browse Artists**
DESCRIPTION:	System allows user to search for browse artists
1.	Using the same kind of layout as the Browse Genres page in the first assignment, you need a way to display a visual list of all the Artists. Like the Browse Genres page, each artist will be a link to the appropriate single-artist.php page.

USE CASE NAME:	**Single Artist**
DESCRIPTION:	System allows user to view information about a single artist
1.	Display information about a single artist (determined by a passed query string value). Similar to the Single Genre page from the first assignment, display a list of paintings by the artist (with each painting being the appropriate link to single_painting.php).
2	Add an Add to Favorites button that is operational (see **Add to Favorite** below).

USE CASE NAME:	**Subjects**
DESCRIPTION:	Provide way to browse and view subjects
1.	Using the same kind of layout as the Browse Genres and Single Genre page in the first assignment, you need a way to view subjects. Like the Browse Genres page, each subject will be a link to the appropriate single-subject.php page. Like the Single Genre page, the single subject page will display a list of paintings for that subject.

USE CASE NAME:	**Browse Museums/Galleries**
DESCRIPTION:	Provide way to view list of museums (that is, all the records in the Galleries table).
1.	This list should be sorted by gallery name and include the gallery name on one line as a link and the gallery city and country in a smaller size on a second line. The link should be to single_gallery.php

USE CASE NAME:	**Browse Painting**
DESCRIPTION:	Provide way to add painting to favorite or cart
1.	From assignment one, you already have buttons to add to favorites or add to cart. Make these buttons operational (see **Add to Favorite** and **Add to Cart** below). Assume the quantity = 1, and the glass, frame, and matte options are none.

USE CASE NAME:	**Single Gallery**
DESCRIPTION:	System allows user to view information about a single gallery
1.	Display information about a single gallery (determined by a passed query string value). Similar to the Single Genre page from the first assignment, display a list of paintings in the gallery (with each painting being the appropriate link to single_painting.php). Instead of displaying the latitude and longitude values, display a Google Map (https://developers.google.com/maps/documentation/javascript/) with a marker of the gallery's location on the map. This will require a little bit of JavaScript. As well, your PHP will page will have to generate a few lines of JavaScript to insert the correct latitude and longitude from the database table into the JavaScript. Be sure to set the map zoom sufficiently large so that the maps shows only a few kilometers around the location.

USE CASE NAME:	**Single Painting Changes**
DESCRIPTION:	Changes to the Single Painting page
1.	In the first assignment, the star rating for the painting (underneath the painting title and artist name) was fixed. In this assignment, you will calculate the average rating for the painting (using the Rating field in the Reviews table and round up) and then display the appropriate number of stars.
2	Display the correct subject labels and links for the current painting.
3	Make the museum name a link to the appropriate single gallery page.
4	Make the Add to Favorite and Add to Cart buttons operational (see **Add to Favorite** and **Add to Cart** below).

USE CASE NAME:	**Simple Search**
DESCRIPTION:	System allows user to search within the paintings from a single search box.
1.	This use case is initiated when a user enters text into search box and hits some type of submit button or link. Note: this search box should be accessible everywhere in the site.

USE CASE NAME:	**Search Results**
DESCRIPTION:	System displays a listing of search results
1.	This use case is initiated after the user makes a search request.
2.	This system will show the matching paintings within `browse-paintings.php`. The results will be those paintings whose title or description contains the search text (i.e., `Like "%searchvalue%"`).

USE CASE NAME:	**Add To Favorites**
DESCRIPTION:	Way to add a painting or artist to the user's favorites.
1.	Initiated when user clicks the Add item to favorites link/button.
2.	The system will add the item to the appropriate favorites (i.e., painting or artist) list and then either go to the **View Favorites** use case or indicate visually somehow that the item has been added to the favorites. The favorites list only needs to last as long as that session. You should check to see if the item is already in the favorite list. If it is already a favorite, do not add a duplicate entry.

USE CASE NAME:	**View Favorites**
DESCRIPTION:	Way to view and modify the contents of user's favorites list.
1.	Initiated when user adds an item to the favorites list, or when user clicks the view favorites link/button.
2.	The system will display a listing of the contents of each of the two favorites list. For favorite paintings, display small square image and painting title; for favorite artist, display small image and artist name. Allow the user to modify the favorites list's contents (i.e., remove an individual item as well as way to quickly remove all items from the list).
3.	For each artist and painting in the favorites list, make them links back to the appropriate single artist / single painting page.

USE CASE NAME:	**Add To Cart**
DESCRIPTION:	Way to add a painting to the user's Cart.
1.	Initiated when user clicks the Add item to cart button.
2.	The system will add the item to the cart list and then either go to the **View Cart** use case or indicate visually somehow that the item has been added to the cart. The Cart list only needs to last as long as that session.
3.	You should check to see if the item is already in the cart. If it is already in the cart, do not add a duplicate entry. You will take one of the following strategies based on context when the user adds a painting that is already in the cart. If the user added the painting from Browse Paintings, then one strategy is to simply increment the quantity; as well, in the View Cart, display some type of message that painting was already in the cart and that the quantity has been increased. An even better option from a usability perspective is within Browse Painting, for that painting, replace the Add to Favorites button with a View Favorites button (use different icon). This way, the user can never add an existing painting to the cart from the Browse Painting page. If the user added the painting from Single Painting, then replace existing cart item for that painting (that is, its quantity, glass, frame, and matt options) with the new values from the form in Single Painting; also, in the View Cart, display some type of message that the existing painting was already in the cart and that it has been replaced. Ideally from a usability perspective, if a painting is already in the cart, then the page should display the current quantity, glass, frame, and matt settings in the form.

USE CASE NAME:	**View Cart**
DESCRIPTION:	Way to view and modify the contents of user's cart list
1.	Initiated when user adds a painting to the cart. For each painting in the shopping cart, make them links back to the appropriate single painting page.
2.	Allow the user to remove an item or empty the entire cart.
3	From the first assignment, you already have been provided with the markup for specifying the quantity, matte, frame, and glass options. The view cart page must also provide a mechanism for changing any of these four items. Provide an Update Cart button that saves these changes to the cart.
4	Display the base cost for each painting in the cart based on the following formula: MSRP*quantity. However, the matte, frames, and glass selections will modify this base amount (see next steps).
5	For each painting in the cart, there is no additional charge if the matte is None. For any of the other mattes, the additional cost is $10 per quantity. For instance, if the user selected bright yellow matte and quantity of 3, then the matte cost would be $30 ($10x3).
6	For each painting in the cart, use the Price field in the TypesFrames and TypesGlass tables to determine the extra cost per quantity. For instance, if the user selected the Lave Black frame (Price = $400) and the Clear glass (Price = $60), and the quantity was 2, then the additional cost for the frames and glass would be $920 ([$400 x 2] + [$60 x 2]).
7	Display a subtotal (the sum of each total) and the shipping cost. The shipping cost varies depending upon the shipper and the amount of the order. The shipping options are Standard Shipping and Express Shipping. Standard shipping is $25 per quantity of painting and express shipping is $50 per quantity. If the total order before shipping is $1500 or above, then standard shipping is free; if the total order is over $2500, then express shipping is also free.
8	Add an Order button (which does nothing), a Continue Shopping button (which returns to home page) and an Update button that will recalculate the totals based on user selections on this page. The update and continue shopping buttons will preserve the user's choices in session state. This means if I move to another page after clicking either of these buttons, my quantity, stock, frame, and glass settings will be preserved.

Guidance

This is substantial assignment and your group will likely need to invest 50-70 hours into it. Some of you will need much less but some groups might need more. I would recommend the following process.

1. This is a data-driven site, so I would recommend beginning by constructing the data access layer. To begin, create data access objects for Artists, Genres, Subjects, Reviews, and Paintings. The architecture provided in chapter 14 gives you `GetAll()` and `GetByID()` functionality already. You will need to add additional methods that match the functionality required by the use cases. I'm not sure I would bother implementing a complex business layer since this assignment is mainly just displaying data. Your business layer will likely just encapsulate the shopping cart and the favorites list.
2. Retrofit your assignment one pages to use this new infrastructure.
3. Add search results functionality to browse-paintings.php.
4. Add session abilities for favorites and cart lists.
5. Implement the cart. There is a reasonable amount of business logic in the view cart use case and this logic needs to be encapsulated within one or more classes. Most of the usability mark will reside in how you implement this page. I would recommend using SemanticUI PopUp and/or modal dialog to help explain the different cart options to the user.