COMP 3512 Assignment #2

Due Tuesday Dec 11th at noon
Version 1.1, Nov 26 2018, most recent changes in yellow

Overview

This assignment is a group project that expands your first assignment in functionality and scope. It is broken into several milestones with different dates. The milestones are in place to ensure your group is progressing appropriately.

Some of the specific details for some of the milestones and pages will develop over time; that is, this assignment specification is a living document that will grow over the next several weeks.

Composition

You can work in groups of three or four for this assignment. It is also possible to work individually or in a pair, but I do discourage it; please talk to me about this if you are planning on working by yourself or as a pair.

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 any due date if your group is experiencing serious problems in this regard.**

I feel foolish saying this in a third-year university course, but it is your responsibility to read all the assignment instructions thoroughly and carefully. If you are unclear about something, ask me. But before you do, read the material in question again!

Submit

You will be using JavaScript, PHP, and MySQL in this assignment. Eventually this will mean your assignment will *likely* be submitted via Cloud9, since I need a way to view your web pages *and* view your programming code. If you wish to use a different hosting environment than Cloud9, you can do so as long as there is a way for me to view the pages and the source code.

If you do use Cloud9, each group member (and me as well) will have to be given RW access to the workspace via the Share button. Cloud9 allows multiple people to simultaneously edit the same file, so you will have to be careful about overwrites if you have two people editing the same block of code.

Source Files

You can find the most recent version of the database scripts and all the images at the GitHub repo for the assignment: https://github.com/mru-comp3512-archive/f2018-assign2.

There is a revised version of the database script on GitHub as well as revised images (i.e., some images have been added, some have been deleted, and some have been replaced). Re-running the script file (using the MySQL command `source art-medium-2018.sql`) will remove your previous tables and re-create and re-populate them). You will have to remove your existing images and replace them with the newest ones.

GitHub

Each group member will need their own Github account. You will also need to create a private repo on Github for the assignment. You have a couple of ways to do this. One way would be for one member in your group to create it (they would thus "own" the repo), and then add other members as collaborators. The free GitHub account doesn't allow private repos; if you sign up for the Student Developer Pack (https://education.github.com/pack) you can have free private repos while a student. The other way is to email me, and I can create a private repo under our department's github organization (https://github.com/MountRoyalCSIS) for your group. You would need to supply the github names or emails for each member.

You will want to push out updates fairly frequently (1-2 times a day when working on it, or more frequently if lots of work is being done). I will examining the commits when marking. You can push your content from Cloud9 to GitHub via the terminal, using the following commands:

```
git init     [only need to do this one command once for your assignment]
git add *
git commit -m "Fixed the rocket launcher"     [alter message and name as appropriate]
git remote rm origin [just in case your cloud9 workspace still linked to my github]
git remote add origin https:... your-repo-url.git      [specify URL of github repo … do this just once]
git push -u origin master   [login using your own individual github credentials]
```

For more information about Git and GitHub, read pages 571-577 of textbook (2nd Edition). There are many online guides to git (for instance, https://guides.github.com/introduction/git-handbook/).

Milestones

You will need to implement certain functionality by specific dates. These won't be marked but you will incur penalties (deductions in marks) if the milestone is not completed on time. These will be periodically updated. You will show me your completed milestones in the lab on the dates shown below.

Milestone 1. You must decide on your group members, set-up your github accounts, and create your workspace in c9 (if you are using it) that will be used collectively by your group. Create a simple version of your About page (see description below). The design and content of this page will change later as you progress through the assignment. But for now, indicate each the group member's name and github page (as a link). Also provide a link to the main github page for the assignment. *Due November 9.*

Milestone 2. Complete a preliminary design and implement it with CSS. Your design needs to be workable at both mobile and desktop browser sizes. This means you will need to make use of media queries. See textbook pages 290-293 for more information. In Lab 12, look at the provided CSS for lab12-test01 to see simple media queries at work. *Due November 16.*

Milestone 3. Create the following APIs in PHP (place them in folder named `services`).
Due November 23.

`services/genre.php` – with no parameter, return JSON representation of all genres. If supplied with `id` parameter, then return just JSON data for single specified genre.

`services/artist.php` – with no parameter, return JSON representation of all artists. If supplied with `id` parameter, then return just JSON data for single specified artist.

`services/gallery.php` – with no parameter, return JSON representation of all galleries. If supplied with `id` parameter, then return just JSON data for single specified gallery.

`services/painting.php` – with no parameter, return JSON representation of all paintings. If supplied with `id` parameter, then return just JSON data for single specified painting. If supplied with `artist` parameter, then return painting data for specified artist id. If supplied with `gallery` parameter, then return painting data for specified gallery id. If supplied with `genre` parameter, then return painting data for specified genre id.

Functionality

Eventually, your assignment will have roughly the following functionality:

Displaying Images: On the various pages below, you will need to display different paintings, artist images, and genre images. In the GitHub Repo for this assignment, you have been provided with full-size versions (either in portrait or landscape mode) and with square versions. Both are very large in size.

You will thus need to create a PHP script that displays image files at specific sizes, similar to what you did in lab 12 (see `lab12-ex10.php` and `lab12-test02.php`). This script will need to take the following parameters via the querystring: 1) square or full size, 2) width 3) type [painting/artist/genre], 4) identifier [ImageFileName/ArtistID/GenreID]. Note: genre images are only square.

For instance, let's imagine you name your script `make-image.php`. The markup to display a painting might look like this:

```
<img src="make-image.php?size=full&width=600&type=painting&file=018050" alt="…" />
```

or

```
<img src="make-image.php?size=0&width=600&type=0&file=018050" alt="…" />
```

or

```
<img src="make-image.php?file=018050" alt="…" />
```

It is up to you as developers to decide what to name your query strings and their values and their default values if not supplied. In the examples above, the third one provides one possibility of using sensible defaults: if width, size, and type not provided, it defaults to a painting in its actual full size.

In the lab 12, you used the PHP `imagescale()` function. When sizing non-square images you need to preserve the aspect ratio. Unfortunately, the imagescale() function is bugged in the version of PHP installed in Cloud9 (5.5.9). I would recommend you update your PHP in your workspace on Cloud9 to version 7, which solves this bug.

How? Run this command in the Bash Terminal in Cloud9 workspace (it will take about 10 minutes to run):

```
git clone https://github.com/elijahcruz12/PHP-7-Installer-for-C9.IO.git && cd PHP-7-Installer-for-C9.IO/ && python c9.io.php.py
```

Don't want to type this? Go to https://github.com/elijahcruz12/PHP-7-Installer-for-C9.IO and you can find the command for easy copy+pasting.

After you upgrade, your version of PHP in your Cloud9 workspace will be 7.0.X. This allows you to provide a value of -1 for the `$height` parameter of `imagescale()`. The function will then calculate the height based on the new width you have provided.

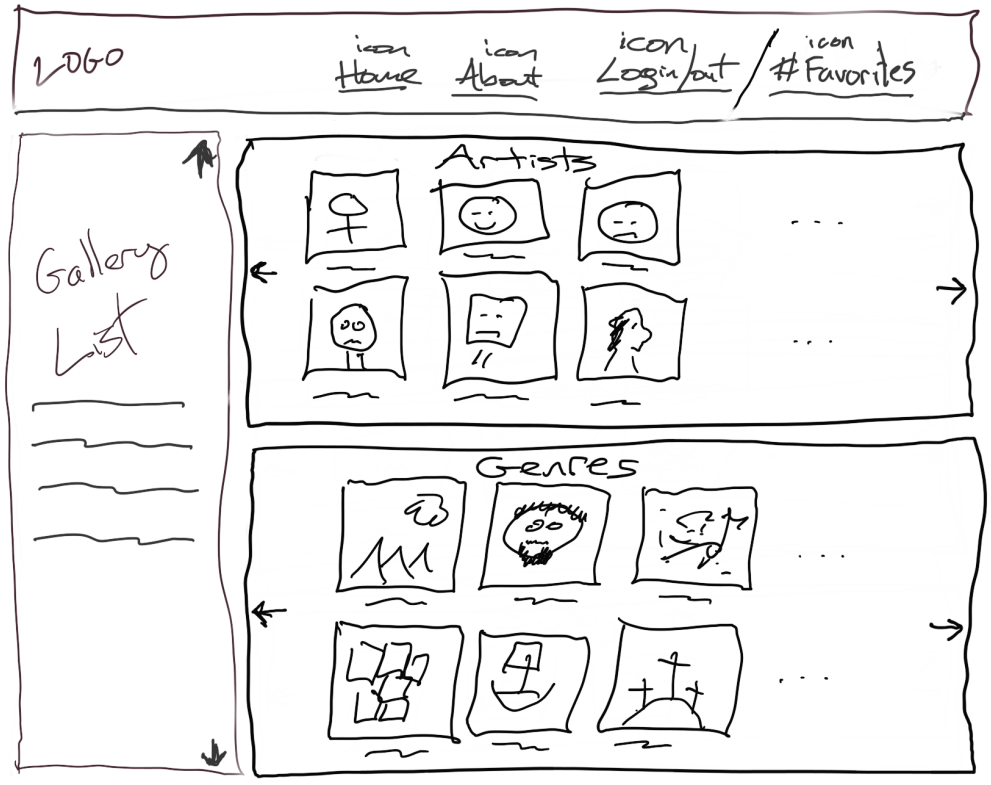
Home Page: will display list of all galleries, list of all artists, and a list of all genres. *I recommend making this page quite simple initially and then add in the horizontal scrolling towards the end.*

The Gallery List **must** be populated with the Gallery API (and thus JavaScript as well).

Artists and Genres lists will be horizontally scrolling lists of links (to get the best marks, this scrolling will be implemented in JavaScript and the Artist and Genre APIs; worst case scenario, implement the scrolling simply via the CSS `overflow` property for less marks). When the user clicks a gallery, artist, or genre, the user will be taken to the appropriate single item view.

Your home page must display a loading animation while the API data is being retrieved. This simply requires using javascript to show a gif animation before the fetch, and use javascript to hide the element after the fetch receives its data.

Your assignment will need some type of consistent header at the top of each page that contains a logo/title, and a menu consisting of links for home page, about page, to login/logout, and to view the favorite list. The options in this list will vary depending if logged in. If not logged in, the options will be: Home, About, Login. If logged in, the options will be: Home, About, Logout, Favorites. It might be nice to use icons and text for these menu labels.



Single Artist Page: This page will be populated using PHP and JavaScript. It will display information for a single artist. Which artist? The artistID must be passed as a query string to the page.

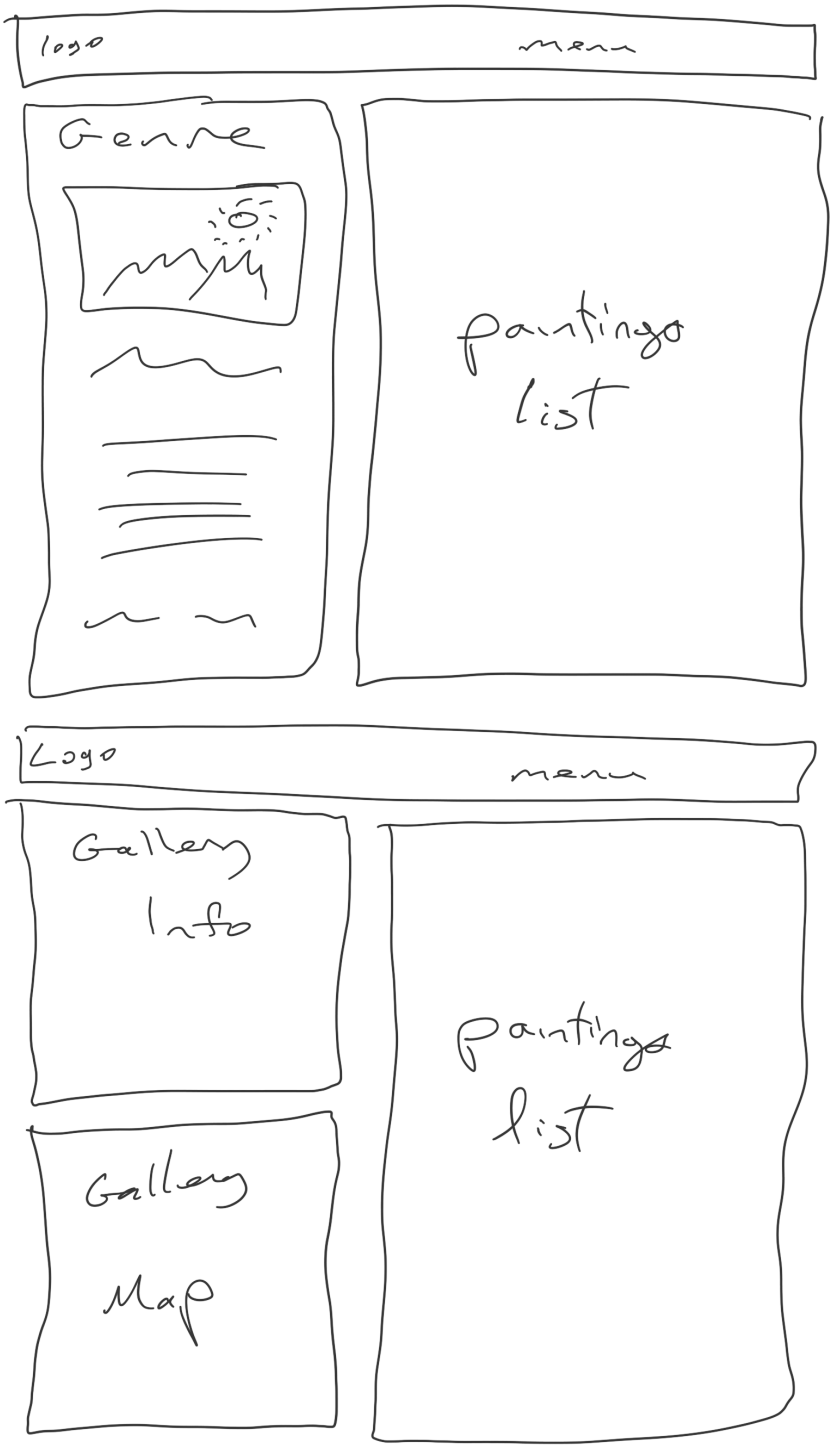
Information from the Artists table in the database will be displayed in the information section. This section will be populated by PHP (i.e., generated by `echo` statements).

This list of paintings will show up on multiple pages and will populate its list using JavaScript and the `services/painting.php` API. The title and thumbnail image will be links to the Single Painting view; the artist name a link to the Single Artist Page. There should be some way to dynamically sort the list by title, artist, or year. This sorting will be performed using JavaScript.

When the user moves the mouse pointer over the square thumbnail of the painting, use JavaScript to dynamically display a larger pop-over version (say a width of about 200 pixels) of the painting; this pop-over version should move with the mouse and then disappear after you move the mouse outside of the thumbnail. This will require using JavaScript setting event handlers for the `mouseenter`, `mouseleave`, and `mousemove` events of each thumbnail.

In the `mouseenter` event, simply unhide a <`div`> and dynamically add an <`img`> to that <`div`> with the larger version of the image. In the `mouseleave` event, re-hide the <`div`> with the image. For the `mousemove` event, the event argument can be used to retrieve the current x,y position of the mouse pointer; you can then programmatically set the CSS `left` and `top` properties of the <`div`> with the larger image.

Single Genre Page and Single Gallery Page: will display information for a single genre or gallery. Information to be displayed comes from the Galleries and Genres tables. The Genre and Gallery info sections will be populated by PHP (i.e., generated by `echo` statements). The paintings list will be generated via JavaScript.



Single Painting Page: will display information for a single painting. Which painting? The `paintingID` will be passed as a query string to the page. Information from the Paintings table in the database will be displayed in the information section. All the data on this page will be populated by PHP (i.e., generated by `echo` statements).

There is quite a bit of data to be displayed so being able to do so effectively will impact the design mark of the assignment. Each painting has a single artist, but multiple genres and reviews.

The color scheme (i.e., the dominant colors in the painting) information resides within a new field named `JsonAnnotations` in the `Paintings` table. The name of the color should either be displayed beside the color (or be available via a tooltip when you mouse over the color). As demonstrated in class, this will require using the `json_decode()` function to turn the string field data into a PHP object.

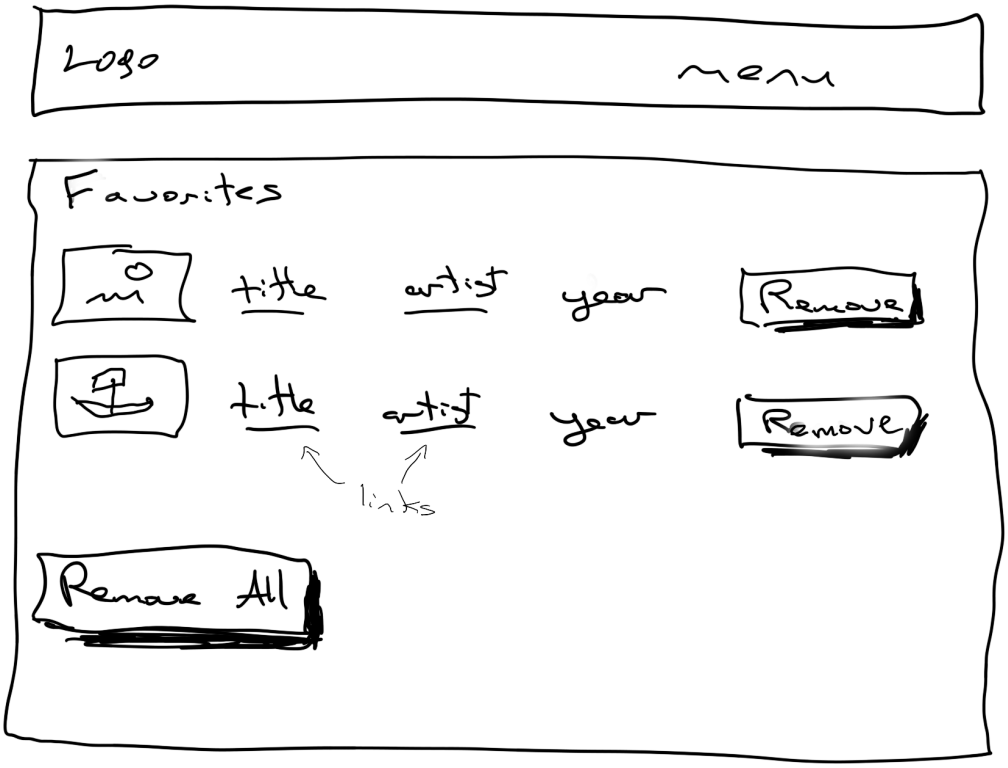
The rating can be calculated from the Reviews. If the user is logged in, they will be able to see a Vote link, but it doesn't need to actually do anything. In this assignment, users won't be able to write a review or add a rating.

The user can add a painting to her favorites list. When the user does click the Add to Favorites link/button, the site will make a request of a file called add_to_favorites.php, update the PHP session value, and then redirect back to the single-painting page, via:

```
header("Location: {$_SERVER['HTTP_REFERER']}");
```

Gallery, artist, and genres should be links to the appropriate single gallery, artist, and genre pages.

Favorites Page: will display list of logged-in user's favorited paintings (implemented via PHP sessions). The user should be able to remove paintings singly or all at once from this list. This page will be entirely in PHP. The title and artist links should be to the appropriate single painting/artist page.



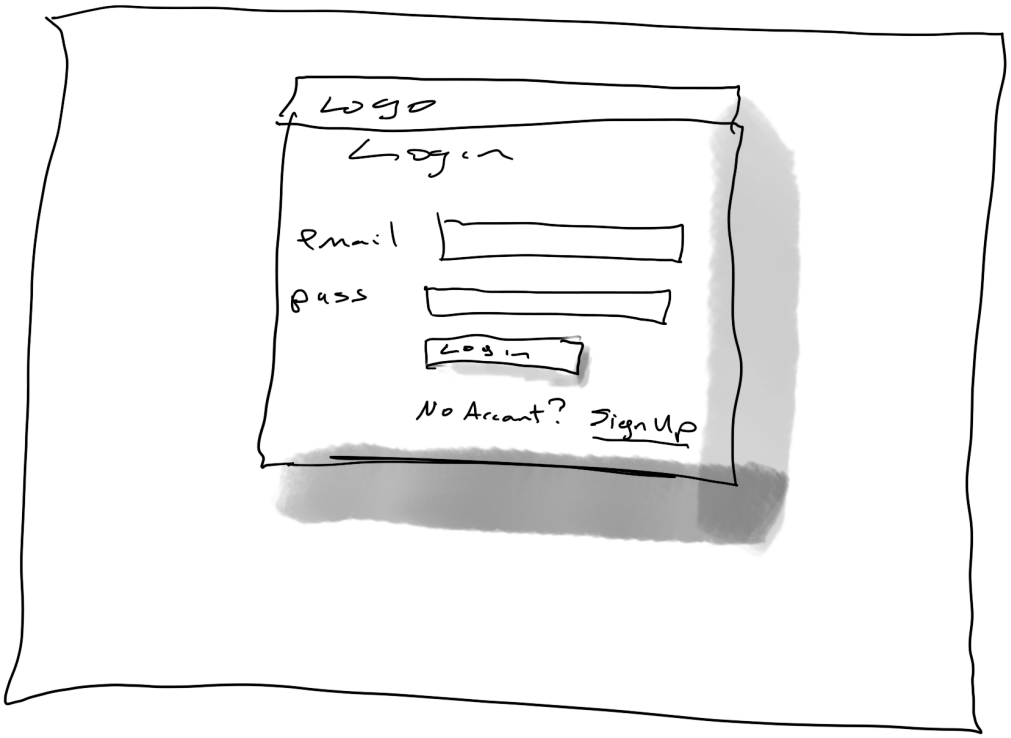
Login Page: will display a login screen. Some of the information below about salts will be explained in class when we cover security.

Your database has two new tables. The first is called `Customers` and contains the following fields: `CustomerID`, `FirstName`, `LastName`, `Address`, `City`, `Region`, `Country`, `Postal`, `Phone`, `Email`. The other new table is called `CustomerLogin` and contains the following fields: `CustomerID`, `UserName`, `Password`, `Salt`, `State`, `DateJoined`, `DateLastModified`.

The actual password for each user is **abcd1234**, but has been salted and subjected to an `md5()` hash function. That means to check for a correct login, you will have to perform two queries: the first to retrieve the Salt field for the entered `UserName` (e.g., tgoyer@apple.com) from the `CustomerLogin` table, and if that user exists, concatenate that salt value and the entered password, and run them through the MD5 function: :

```
$hashed_pass = md5($_POST['pass'] . $salt_for_this_user);
$sql = "SELECT … WHERE … UserName=? AND pass=?"
$pdo->bindValue(2, $hashed_pass)
```

If the result of that function matches the result in the Password field, then log the user in.



Registration/Signup Page: will display a registration form. Perform the following client-side validation checks using JavaScript: first name, last name, city, and country can't be blank, email must be in proper format (use regular expressions), and the two passwords must match and be 6 characters long. Be sure to display any error messages nicely.

When user clicks signup, you will have to check to ensure that the email doesn't already exist. If it does, return and display nice error message. If email is new, then add new record to both `Customers` and `CustomerLogin` tables. The email field from the form is used as well for the `UserName` field.

You will **not** store the password as plain text in the database. Instead, you will have to generate a random 32 character salt (which will be saved in the `Salt` field) of the `CustomerLogin` table. The value you save in the `pass` field will be the result of the concatenation of the plain text entered password plus the salt after it has been hashed via the `md5()` function:

```
$pass = md5($_POST['pass'] . $salt);
```

Note: since we are working with existing customer login info (which was created with MD5), we are obliged to keep using it. If we were starting from scratch, however, we would not use MD5 since it is not secure enough, and would use crypt() instead.