INFO / CS 2300 - Project 3: Image Album

Introduction

Project 3 is designed to give you practice building and implementing a database backed website from scratch. As you may have encountered in Project 2, text files can be very difficult to open, read, and write. Databases provide a clean, easy alternative for storing and organizing information. Your task is to develop a personal online image album, using PHP to interact with a MySQL database you will create and populate yourself. It needs to implement a **many to many** relationship. One image can be on multiple albums and one album can have multiple images.

Your content can be something other than an image album but it must follow a similar structure - two entities in a many to many relationship with each other. If you choose entities other than images and albums, you should explain in each rationale you upload to CMS how your entities correspond to the requirements of the assignment as written in the language of images and albums.

Grading

The project has been divided into 3 milestones, with an additional Code Review. **WOW points will only be given for Milestone 3.**

Milestone 1: Due Mar.14th, 15% of P3 grade
Milestone 2: Due Mar. 21st, 25% of P3 grade
Milestone 3: Due Mar. 28th, 55% of P3 grade
Code Review: Starting Mar. 29th, 5% of P3 grade

Important Notes

You will encounter code online using the mysql_connect functions but they are deprecated and you should **not** use them. Use the mysqli functions (or PDO) as taught in class. **If there is any appearance of these deprecated functions in your code, 10 points will be taken off your grade for that milestone.**

If you use file functions, make sure your file permissions are set open to the public for read, write, and execute. **If we receive a file permission error during grading and any part fails to function properly because you will receive a zero for that part**--we will no longer email you to fix your permissions or fix them for you.

Rigorous SQL injection prevention is not required for this assignment. We'll get to that in a later lecture. But do validate user input for numbers and/or appropriate string values.

Please start early and make sure you understand every part of your code, as you may be asked to explain any part of it during the Code Review.

Milestone 1 - Initial Design (Due Mar.14th)

Goal

Set up your database and connect to it from your website

Requirements

Create basic skeleton for your site

- Set up the pages that you think you will need in your website (There is no minimum number of pages. If you do a single page site, you should set up the sections as the pages are described here)
- On each page of your website, include a title and brief description of what will go on that page.
    - e.g. home page, display all albums, display images from specific album, display login form
- Link your pages with a clear and consistent navigation. There should not be any dead-end pages on your site that require the user to click the “back” button in order to continue navigation.
- Organize your files so that it is easy to add new pages. If you have a consistent header on all pages, plan on including a header.php file or a function that creates the header. Same with the footer. Make sure your navigation is created in just one place so that as you change it, you don’t have to change it on every page.
- Have a clear directory structure for your files - css folder, docs folder, images folder, etc. This will make it very easy to keep track of your files as the milestones get more complex. See this reference sheet to get more information on directory structures.

Design your database

- Create a list of all the table information you need to make your site functional. Your two main entities, albums and images, should be in a **many to many relationship** with each other. One album can contain many images and one image can appear in many albums. Note: you will eventually need a user login table but you don’t need to specify that for this milestone. An example of a many to many relationship is the schema for Boats, Sailors and Reserves as we have been using in lecture. For more information, see Different Types of Table Relationships on Piazza.
- Write your schema in a manner which, for each table, shows field names, types (string, integer, date, …), size (if string), allow nulls, primary key). For example:
  table1

name	type	size	allow null	primary key
field_1	int		false	true
field_2	string	32	false	false

- At minimum, your albums should have fields for title, date_created, date_modified and one other field. Your images should have caption and/or title, file_path or file_name, credit. You need enough fields for both albums and images to create a reasonable search.
- Your schema will be part of the P3_1.pdf that you will be uploading to CMS for this milestone.

  Note: Everything else will come much easier if you have a good understanding about what tables you need, and what the attributes / fields in each table should be. More fields are better than too few at this point – you can revise the schema in Milestone 2.

Create and populate your database

- Log in to phpMyAdmin
  - https://info2300.coecis.cornell.edu/phpMyAdmin/
  - Username: your netidsp17 username
  - Password: your password for your username
  - Use your info230_SP17_*username* database for your tables.
- Create your tables using SQL CREATE MyTable statements or use phpMyAdmin's interface. See the activity and notes for **Section (2017-03-10**.)
- Use phpMyAdmin to add records to your tables with **at least 1 album and 3 images** linked to the album.

Add images to your file system

- Your database will keep track of which images exist, but the image files themselves are stored on the file system, likely in a folder for that purpose.
- Eventually, you will need to be able to upload new images using a browser but for this milestone you can add the images using FTP (File Transfer Protocol).

Create a config.php file

- This file defines your DB_HOST, DB_USER, DB_PASS, and DB_NAME variables needed to instantiate a mysqli object to connect to your database. You will use mysqli objects to connect to your database and run SQL queries on your newly created tables. Information on how to do this is in **Lecture 12 (2017-03-08)**. If you copy-paste code, beware of funky errors that appear from incorrectly encoded single and double quotes.

Display an album on your site

- You'll need a page on your site to display a single album. For this milestone it can simply display fields from your only album and all the images in your images table (either as thumbnails or at full size).
- Implement this by accessing the database dynamically using PHP (don't hardcode it). In other words, your image filenames should **not** appear in your PHP. You should be retrieving image filenames from the database.
- In the next milestone, you will need to modify this to only show images from a specific

(user-selected) album.

Create basic styles for your site with CSS

- Make sure that the pages are user friendly and aesthetically pleasing.
- Consider your layout, colors, and fonts.
- It does not need to be a fully polished product and can be changed later, but note that the more design you accomplish now, the less you will have to do in later functionality-heavy milestones.

What to Submit

- P3_M1.pdf to CMS and the course server. It should include:
    - Your database name, username, and password so TAs can log in to your database
    - Your relational schema
- Implemented tables to your course server database.
- Basic website to course server inside a P3_M1 folder.
- Reminder: Just because it works on your local host does **not** mean the copied files will work on the course server. **Test it on the server**.
- **IMPORTANT:** If you are going to start on Milestone 2 before M1 is graded, make a copy of the files on the course server, and put it in a different directory (P3_M2). Don't modify your P3_M1 files until you have your grade for P3_M1 or else your files for P3_M1 look like they were submitted late. You shouldn't have to create a copy of the database but be careful not to mess up your database on the server until you've been graded for M1.

Grading Rubric

- **NOTE:** As stated above, we are not fixing file permission errors. Grading will be done as is.
- Initial schema (__/40)
    - Workable schema included in P3_M1.pdf on CMS (__/8)
    - Username and password included in P3_M1.pdf on CMS (__/2)
    - Implementation of the described schema in MySQL on the INFO 2300 server (__/20)
    - Album and Image tables exist and have a many to many database relationship? (__/6)
    - Tables are populated with at least one album and three images (__/4)
- Initial functionality (__/40):
    - Does the code work and display images from the album in the database? (__/20)
        - 100% works - 20pts; few errors - 15pts; somewhat works but many errors - 10pts; does not work but code present - 3pts.
    - Done via PHP calls to the database? (__/20)
- Initial design and navigability of pages (__/20):
    - Does the site have functional navigation? (__/10)
    - Is the site navigation consistent and easy to use? (__/3)

    - Is the site styled appropriately (all content is legible and easy to read, uses appropriate color, typography, layout, and positioning)? (__/7)

-10pts if mysql_connect functions are used.

Milestone 2 - More Functionality (Due March 21)

Goal

Create HTML forms to upload images and modify your database

Requirements

Add another album and three more images

- It is okay to do this using phpMyAdmin and FTP (File Transfer Protocol)
- You should now have **at least 2 albums and at least 3 images** in each album
- At least one image should appear in multiple albums. At least one image should appear in only one album.

Display album list

- Users should have a means of showing a list of all albums. The list shows the title for each album and possibly another field or two but **not** the images
- Users can click on an album in the list to display that album. You'll need to create a unique URL for each album (i.e. album.php?album_id=1)

Display user selected album

- When the user clicks the album in the list you created above, that album should display with all its appropriate fields and the photos that are in that album.
- Modify the album page you created in Milestone 1 so that it responds to a parameter in the URL (i.e. album.php?album_id=1) to display the appropriate album. This step combines what you learned in Project 2 (data validation) with what you learned in Homework 2 (SQL queries). You'll need to modify your SQL statement(s) to safely use the parameter from the URL (e.g. If you are expecting the parameter to be an integer, make sure it is an integer before using it).

Add an album using the browser

Create an "add album" form that allows the user to add a new album to your database. This means that the user should be able to specify field values such as an album name and an album description (assuming name and description are fields in your albums table). Upon form submission, the user's input should first be validated, then used to form a SQL INSERT statement. This INSERT statement should then be given to your mysqli connection. To test your SQL statement independent of your website, you can use the SQL tab in the phpMyAdmin tool and see if the new album record shows up in your albums table. In the final version of this project, this will need to be an admin-only feature available only to a logged in user.

Add an image from the browser

Create an upload photo form that allows the user to upload a new image. Information on how to do this is in **Lecture 13 (2017-03-13**.) The image itself should be stored in an images folder and the database should keep track of the filename. Make sure the user can upload as many images as they want without getting any errors. You don't need to support multiple file uploads at once, but there shouldn't be a limit to the number of files that can be uploaded over time. A best practice would be to restrict excessively large images or create appropriately sized copies of large images but that isn't necessary for this project. The user should have the option of whether or not to add the photo to zero or more albums. In the final version of this project, this will need to be an admin-only feature available only to a logged in user.

What to Submit

- P3_M2.pdf to CMS and the course server. It should include:
  - All content from your P3_M1.pdf
  - Any updates to your schema. If there are just a couple small changes, you can name them. If the changes are significant you should include the new schema. If you aren't sure whether changes are small or significant, do both. If there were no changes, just add that fact as a sentence.
  - Name an image that is on more than one album and which albums it is on.
  - Name an image that is not on one album but not others.
- Updated website to course server under P3_M2.
- **IMPORTANT:** If you are going to start on the final version before M2 is graded, make a copy of the files on the course server in a folder named something like P3_M3. Don't modify your P3_M2 files until you have your grade for that part or else your files look late. Until you are graded for M2, be careful about changes to the database that you are probably sharing between your M2 and M3 files.

Grading Rubric

P3_M2.pdf in CMS (_/5)

Display album list (_/10)

- Page has at least 2 albums listed (_/1)
- Albums are displayed by correctly accessing the database dynamically using PHP (not hardcoded) (_/5)
- Page works and albums display without errors (_/2)
- Albums in the list have appropriate links to a unique album (_/2)

Display a single album and the images in it (_/25)

- The correct album displays based on the URL parameter (_/2)
- At least 2 albums contain at least 3 images each (_/2)

- At least one image is shared between two albums (_/2)
- At least one image is on an album by itself (_/2)
- Page works and images display without errors (_/7)
- Images are displayed by correctly accessing the database dynamically using PHP (not hardcoded) (_/10)

Add a new album (_/30)

- The form allows user to submit successfully and functions without error (_/5)
- The new album is added by correctly accessing the database dynamically using PHP (not hardcoded) (_/15)
- The new album is displayed on the albums page without errors and persists when page is closed and reopened (_/10)

Upload an image and add it to zero or more albums (_/30)

- An image can be uploaded successfully without errors (_/10)
- The image can be added to an album and displays on the page of that album after being added (_/10)
- The image persists when website is closed and reopened (_/10)

-10pts if mysql_connect functions are used.

Milestone 3 - The Finished Product (Due March 28)

Goal

Polish your website for a better user experience

Requirements

Search form

Add a search form that allows the user to search for photos whose text (varchar datatype) fields contain the user's input (after validation, of course). This functionality is just like what we asked for in Project 2, and should support searches across a single field as well as across multiple fields. The easiest approach is to use a general search form (one text field and a submit button), and to search for matching values within any of the fields in your photos and albums tables. If you're not sure how to form this SQL query, try experimenting with the Search tab in phpMyAdmin.

Image detail

When viewing the photos of an album or looking at search results, the user should be able to click on the image's thumbnail and be able to see the full size image along with the image's caption, image credit, along with any other fields for the image (e.g. date_taken if your images have that field). It should show what album(s) this image is in. An image can be in one or many albums.

Login functionality

- Add a login form (username and password) to allow users to use the site as an admin. You don't need to create a new user. Just tell us the username and password in your rationale.pdf. You will need to create a $_SESSION when the user logs in, and destroy it when the user logs out. You can check if a user is currently logged in as an admin by checking if this $_SESSION variable is set. Password must be stored in a hashed form. Information on how to do this is in **Lecture 14 (2017-03-15**.)
- This needs to be implemented using a user table as seen in lecture. You only need one user, but a user login table would allow more.
- A user must be **logged in as an admin** in order to add a new photo or album, edit an existing photo or album, and delete an existing form or album. It may be a good idea to hide the add/edit/delete options from the user if they are not logged in as an admin. You should also display a "You must be logged in to use this feature" message (and don't show any of the add/edit/delete forms) if the user tries to access your add/edit/delete forms without logging in as an admin.

Edit album

Have an edit album form that allows the user to update the fields of an existing album record, but only if the user is currently logged in as an admin. Allow adding an existing image to this album. Allow removing an image from one album while leaving it available

for other albums. **Don't forget to validate the user input before forming your SQL queries.**

Edit image

Have an edit photo form that allows the user to update the fields of an existing photo record, but only if the user is currently logged in as an admin. **Don't forget to validate the user input before forming your SQL query.**

Delete album

Allow the user to delete an existing album, but only if the user is currently logged in as an admin. It may be a good idea to ask the user to confirm their desire to delete the album. Make sure that the related database records are appropriately handled. Images can continue to exist even if they are no longer associated with any existing albums. There should not be any lingering references to the deleted album.

Delete image record

Allow the user to delete an existing image, but only if the user is currently logged in as an admin. Make sure that any albums that this image belonged to are updated appropriately so that no lingering references to the image remain. It is sufficient to delete the image record from the database without deleting the associated file though deleting the associated file would be a more complete implementation.

Populate your site

You should have **at least 3 albums, each with at least 8 images**. It may be a good idea to do this with your Add Album and Upload Photo forms, to test that they both work correctly before submission. Making this work on your local computer and uploading your files to the server is not good enough. **Test everything on the server**.

What to Submit

- rationale.pdf to CMS and the course server. It should include:
  - Provide your **login username and password**.
  - Explain your aesthetic design choices (e.g. color, layout, navigation, theme).
  - Explain the functionality design choices (e.g. how pages are implemented clearly, where functionality is placed, additional functionality added).
  - Explain improvements from milestone feedback (either how feedback was implemented or why it was not).
  - Explain any WOW features you implemented.
- Updated website to course server under P3.

Grading Rubric

Design - (__/15)

- Content is legible, easy to read and understand (__/3)
- Navigation is easy to use and no dangling pages (__/3)
- Appropriate color, typography, layout, and positioning for the chosen theme (__/5)

- Pleasant to look at; Creative and interesting (__/4)

Design Rationale - (__/10)

- Provided login username and password for the website and the database (__/2)
- Explanation of Design Choices (e.g., the choice of color, layout, nav, and chosen theme) (__/3)
- Response to feedback from milestones 1 and 2 (__/5)

PHP and MySQL Functionality (__/60)

- Search form (__/ 14)
    - Site should include a form to search on the text in captions, title, album etc.
    - This form is visible to all users
    - Should include partial search
    - Should be able to search across multiple fields
    - Error checking for malicious and incorrect input
- Image detail (__/ 5)
    - Shows the image, credit, other fields, and albums to which this image belongs when it is clicked on
- Login system (__/ 12)
    - Must utilize $_SESSION variables as discussed in class (the password should be hashed)
    - Add and Update Forms only visible to a single administrative user who logs into the site via a username and password
    - Error checking for malicious and incorrect input
- Edit album (__/ 7)
    - Only the admin is able to edit an existing album
    - A form is present to edit albums
    - The changes are persistent and the database is updated with the relevant information
    - Error checking for malicious and incorrect input
- Edit image (__/ 7)
    - Only the admin is able to edit an existing image
    - A form is present to edit images
    - The changes are persistent and the database is updated with the relevant information
    - Error checking for malicious and incorrect input
- Delete album (__/ 5)
    - Images in the album still persist on the website (i.e. they are not deleted from the database after the album is deleted)
- Delete image record (__/ 5):
    - All related database records are appropriately handled
- Content (__/ 5)
    - There are at least 3 albums, each with at least 8 images
    - User is able to see all albums and images in the database

Content (__/10)

- All album entry in the database includes a title, date created and date last modified (__/6)
  (date last modified worth 2; others 1)
- A image entry in the database should include a caption, image url, and date taken (__/2)
- Database contains at least (3) different albums with at least (8) images in each (__/2)

Organization (__/5)

- readable and clear PHP, HTML, and CSS with comments in appropriate places(__/2)
- HTML validates (for complete pages, not for include() or require() pages) (__/1)
- pages are well organized in server directories (images are in an img folder, etc) (__/2)

Deductions

- Use of deprecated mysql_connect functions -10pts
- Anything from earlier milestones that no longer works - up to 10 points
- Failure to implement feedback from earlier milestones could result in a loss of up to 5 more points (besides those that may have been lost in the rationale section.) Yes, this is a big deal. If a client or supervisor gives you feedback, you take it very seriously.

Wow Points (up to 5)

This is reserved for websites that go above and beyond the requirements. These 5 points will be applied to other parts of this assignment where you may have lost points, but will not take your overall score for this project over 100. Elements of wow factor include, but are not limited to:

- Awesome design
- Excellent use of JavaScript, jQuery, or Ajax to simplify user interactions
- A means to display images not in any album
- Delete image files not just the database record when deleting
- Allowing an image to be added to or removed from albums when editing the image
- Image resizing and management of multiple images sizes (thumbnails in list view and full size images in detail view)
- Buttons that sorts the album list by fields

Code Review (Starting March 29)

Goal

In a one-on-one conversation with a TA, show us how much you understood everything you did for this project.

Following the submission of Milestone 3, each student will participate in a code review. The purpose of the review is to show off, explain, and answer questions about your project.

Students will be asked to sign up for a 15 minute slot via CMS with one of the course staff. Sign ups will begin between the due date for M2 and M3, and will be scheduled on a first come, first serve basis. Code reviews will begin March 29, and will continue after Spring Break.

What to Expect: Students will be asked to demo their sites to the TA, making sure to cover core functionalities. Students should be prepared to answer general questions about course material relevant to the Project 3, as well as more specific questions regarding their individual implementations. Note: there is no advantage in trying to hide flaws or bugs. Instead, it would be best to explain why issues occur, as well as what steps were taken to attempt to address them.