Project 2: Backing a Website with a Database

Create a small catalog website that displays any collection of objects in a database.

1. Overview

Project 2 is about learning how to use a database to render the contents of a web page. To develop these skills you'll build a small catalog website. You'll store the data for your catalog in a SQLite database. You'll then write a PHP web page to read the contents of the database and then present the data in a usable form for the web.

1.1. Learning Objectives

- Continuation of developing your algorithm based problem solving skills.
  - Develop good programming habits: planning first and then implementation.
  - Further development of your *programmer's toolbox* to solve problems with code.
- Exposure to the standard practice of backing website content with a database.
  - Using a database with PHP to populate the content of a web page.
  - Experience with using a development database: SQLite.
  - Basics of database schema design.
  - Exposure to basic SQL queries and security measures to prevent SQL injection.
  - Practice with *filter input, escape output* for HTTP parameters to help secure your web site.

1.2. Deadlines & Receiving Credit

Milestone	Points	Grade	Sip Days	Deadline
Project 2, Milestone 1 (*p2m1*)	20	Feedback (*completion*)	**Not Permitted**	Tue 3/5, 4:00pm
Project 2, Final (*p2fin*)	**100**	Rubric (*p2m1* + *p2fin*)	Maximum: 2 days	Tue 3/12, 4:00pm

Milestones are graded *twice*. First for feedback (completion grade only; no partial credit). Lastly, for points at the final submission (via rubric). **All** work is graded for points (not completion grade) at the final submission.

Completion credit will be awarded so long as you made a good faith effort to complete the milestone's requirements. Very obviously incomplete milestones will receive a 0 for the completion grade.

Use your milestone feedback to improve your final grade; revise milestone work prior to final submission. If you didn't submit a milestone or received negative feedback, make sure you make up or revise the work for the final submission.

1.3. Git Repository & Submission

Clone `git@github.coecis.cornell.edu:info2300-sp2019/YOUR_GITHUB_USERNAME-project-2.git`. Replace **YOUR_GITHUB_USERNAME** in the URL with **your actual GitHub username**. This is usually your NetID.

Submit **all** materials to your GitHub repository for this assignment. **See *README.md* in your Git repository for submission instructions for each milestone.**

Tip: Commit and push your changes every time you work on your project. Every time you commit and push you store your changes on the GitHub server. This acts as a back-up for your work. It also means that if you forget to submit before the deadline, there's something already on the server that the TAs can grade.

2. Requirements

Create a small catalog website that displays any collection of objects in a database.

Examples of catalogs you might consider include:

- A music catalog like Billboard or Discogs.
- A movie catalog like IMDB (but more simplified).
- A video game catalog like VGCollect or Steam (without the download part).
- An online store like Baker Creek, or Rogue Fitness (without the payment part).
- A visual dictionary like a PokéDex or Bird Guide.
- A simple blog or forum like Tumblr, Twitter, or Reddit (but without user accounts).

For example, your collection might look like the following on the *index* page:

Title	Actors	Genre	Rating
Shrek	Mike Myers, Eddie Murphy	Adventure/Comedy	PG
Despicable Me	Steve Carell, Chris Renaud	Comedy/Animation	PG
Toy Story	Tom Hanks, Tim Allen	Fantasy/Adventure	G
Treasure Planet	Joseph Gordon-Levitt, Emma Thompson	Romance/Adventure	PG
The Iron Giant	Vin Diesel, Jennifer Aniston	Action/Adventure	PG

2.1. Design

- You must thoroughly document your design and planning process.
- The design should be appropriate for your target audience(s).
- The design of the site should appropriately match the theme of your catalog site.
- The site should have a consistent look and feel.
- The site should have clear, easy-to-follow navigation, if appropriate.

  If you display results on multiple different pages, you should be able to travel back to the previous page using navigation links or breadcrumbs, instead of relying on the browser's back button.
- Your design does not need to be fancy, but it should look nice and be aesthetically pleasing.
- You may assume that your web page will be viewed in a desktop browser.
- All images in your design journey must be visible in Markdown Preview.
- All images must be labeled in your design journey. Labels must be visible in Markdown Preview.

2.2. Website

- Plan and design your website before implementation.
- Site must contain at least 1 page backed by a database.
- Your website must allow the user to view the entire collection of entries in your catalog at once.
- You must include a search form for returning search results from the database.
  - Basic search is acceptable and encouraged.

    You are not building a search engine.
  - Search must be implemented with 1 database query.
  - Users must be able to search across multiple fields.

    This requirement is vague to give you as much freedom as possible for your target audience. This means you could have an input for each field in your search form. It also means you could have 1 input in your search form and then search for that input across multiple fields in your database. The choice is yours.
  - Your search results will probably display in a similar manner as the full collection. Make sure it is clear to your target audiences when they are viewing the full collection and when they are viewing the results of a search.

- You must include an HTML form to add entries to the catalog's database.
    - Inserting entries in the database must be completed with 1 database query.
    - File uploads are not permitted.

      If you want images associated with an entry, you can have a set of images on the server that the user can select from a drop-down list, etc.
- Form input validation and corrective feedback is **not** required.

  You should always validate the user's input and provide corrective feedback. However, there isn't time to work on this for this project. Instead make sure you filter your inputs to secure your web site.
- Your code must make effective use templates (includes).
- Your code must make effective use of functions. Minimum of 2 user-defined functions required.
- **init.php** must be included on all pages.
- Remember, cite all images according to the course policy. This includes images associated with the database.

2.3. Database

- Plan your database schema before implementation.
- You must store all catalog data in a SQLite database: **secure/site.sqlite**.
- Your catalog must be stored in 1 database table.
- Your catalog's table must include a minimum of 4 fields (columns).

  You may not count the **id** field towards these 4.
- You must populate your database with seed data with a minimum of 5 complete entries.
- You may only store normal/conventional types of data in your database.
    - Numbers (integers, real, numeric) and Text are acceptable.
    - Blob is not acceptable (i.e. no images, movies, files, etc.)
- Each field must be properly constrained.
- Your table must include a *primary key* titled **id**.
    - **id** must be an integer.
    - **id** cannot be null.
    - **id** must be unique.
- PHP's PDO extension must be used to access to the database.

2.4. Security

Your code should be secure against database injection attacks and cross-site scripting attacks.

Your code should filter input and escape output.

- Filter/sanitize all input.
- Escape all output for SQL and HTML.

2.5. Coding Requirements

- All code must be your own work. You may not use code from other classes, including INFO 1300.
- Site must be coded in valid HTML, CSS, PHP, and SQL.

  JavaScript is prohibited for this assignment.
- Your site should display reasonably well (not necessarily identically) across Firefox and Chrome.
- You may assume that your web page will be viewed in a desktop browser.

  You do not need to design a web page for mobile browsers. You do not need to use media queries.

2.6. Best Practices

Your assignment should follow the coding standards, conventions, and expectations of this class. See Project 1 if you need a reminder for some of them.

All code should be easy to read and understand by anyone. Use functions and includes to help organize your code. Make use of comments to explain things that aren't obvious. Name your functions and variables appropriately. Your code should be indented properly so that it's easy to see which lines belong to each element (HTML/CSS) or function (PHP).

Test your add entry and search forms extensively.

3. Milestone 1: Design, Plan, & Draft Website

In Milestone 1, you'll design and plan your website and database.

Complete the **Milestone 1** section of the design journey: **documents/design-journey.md**.

3.1. Plan your Site

In the design journey thoroughly document your design process for your site.

You will need to identify your target audiences for this project. You need at least 1 but you may have more depending on your site.

3.2. Plan your Database

Plan out your database schema.

Think about the following questions.

- What columns/fields will you have in your table?
- What type will each column/field be?
- What constraints will your fields have? (primary key, not null, auto increment, etc.)

3.3. Plan your Database Queries

You'll need at least three (3) database queries: 1) retrieve all results, 2) search the results, and 3) insert a record into the database. Describe your plan for these queries. You may use natural language, pseudocode, or the SQL queries themselves.

Example:

> **Natural Language/Pseudocode**: Select all records and all fields from the movies table.
> **SQL**: `select * from movies;`

3.4. Populated Database

Using **DB Browser for SQlite**, populate your **secure/data.sqlite** database with your seed data.

Do not run the PHP Server and DB Browser for SQLite at the same time! This can cause your database to become corrupted. If you need to modify the database, stop the PHP server and then open DB Browser. When you are done with DB Browser you must exit it before you restart the PHP server.

3.5. Draft Website

Code up your design and produce a draft website. You don't need to implement any calls to the database yet. But the website (HTML, CSS, PHP Templates, Forms) should mostly be done.

4. Final Submission: Complete & Polished Website

For the final submission, finish implementing your designed and planned website. Your final site should be complete and polished. All catalog content should be populated from the database and both forms should be fully functional.

Complete the **Final Submission** section of the design journey: **documents/design-journey.md**.

All design/planning images must be visible in your design journey when using Markdown Preview. No credit will be provided for images in your repository that not properly linked in Markdown.

Test your final web site thoroughly, especially your forms. We will try to break your web page during grading. Make sure that we can't inject HTML or SQL.