Project 2: Dynamic Website Backed by a Database

Create a small catalog website that displays any collection of objects in a database. For this project, 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. 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.

2. Deadlines & Receiving Credit

Milestone	Points	Grading Method	Sip Days	Deadline
Milestone 1 (*p2m1*)	15	Feedback (*completion*)	**Not Permitted**	2/27, 4:00pm
Milestone 2 (*p2m2*)	15	Feedback (*completion*)	**Not Permitted**	3/5, 4:00pm
Final (*p2fin*)	**100**	Rubric (*p2m1* + *p2fin*)	Maximum: 2 days	3/12, 4:00pm

No slip days permitted for milestones. Late submissions will receive a 0.

Milestones are graded *twice*. First for feedback (completion grade only). Lastly, for points at the final submission (via rubric). ***All* work is graded via rubric for points at the final submission.** Use your milestone feedback to improve your final grade; **revise milestone work prior to 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 milestone submissions will receive a 0** for the completion grade.

Milestone feedback is not a *pre-grade*. This feedback is designed to catch large problems (which we sometimes miss). Regardless of the feedback (or lack of feedback) that you get, ***you* are responsible for meeting all of the project's requirements for the final submission**.

Failure to *push* your submission to GitHub is equivalent to not submitting the assignment. You will receive a 0. It is your responsibility to verify that you submitted your assignment.

3. Git Repository & Submission

Clone `git@github.coecis.cornell.edu:info2300-2020sp/YOUR_GITHUB_USERNAME-project-2.git`. Replace **YOUR_GITHUB_USERNAME** in the URL with **your 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.** Never email your submission to the instructor.

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 for partial credit.

4. Documenting Design (Design Journey)

We will grade your **design-journey.md** in VS Code's Markdown Preview. **Everything, including images, must be visible in VS Code's Markdown Preview.** If it's not visible in VS Code's Markdown Preview, then we won't grade it. We won't give you partial credit either. This is your warning.

If you included images in your design journey, they must be visible in VS Code's Markdown Preview. No credit will be provided for images in your repository that are not properly linked in Markdown. No credit will be given for design work not included in the **documents/design-journey.md** file; **do not rename this file or put your answers in another file!** Remember to check and test all assignment submissions!

Requirements

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

Examples of catalogs you might consider include:

- A listing of weightlifting exercises.
- A course catalog for InfoSci majors.
- 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

Important! Image gallery *catalogs* are **prohibited** for this assignment. (Image gallery is for Project 3 when we learn how to do file uploads!)

1. Catalog

- You may pick any type of catalog except those similar to an image gallery.
- Users must be able to view all entries in the catalog at once.
- Users must be able to search your catalog and view the results of that search.
- Users must be able to add entries into your catalog.

2. Database

Planning

- Plan your database schema before implementation.

Constraints

- You must store all catalog data in a SQLite database: **secure/catalog.sqlite**.
- Your catalog must be stored in **exactly** 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 may only store normal/conventional types of data in your database.
  - Numbers (integers, real, numeric) and Text are acceptable.
  - Blob is prohibited (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.

Seed Data

- You must populate your database with seed data with a minimum of 5 complete entries.

3. Design

The goal of Project 2 is to help you understand how to back a website by a database. Design is always important, but just remember that in this project we are focusing a bit more on the technology.

Site Design

- The design of the site should appropriately match the theme of your catalog site.
- The design should be appropriate for your target audience(s).
- Your site's design need not be complex. A fairly basic site is acceptable for this project.

  There is no minimum/maximum number of pages required for this assignment. You need enough to do the job. If you only need 1 page to produce a usable catalog for your target audiences, that is acceptable.
- Your design does not need to be fancy, but it should look nice and be aesthetically pleasing.

- 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.

  Navigation is not required. But if you decide your design needs it, it must be well organized.

- You may assume that your web page will be viewed in a desktop browser.
- Be mindful of common design patterns for your type of catalog.

  Your design should leverage the familiarity of these design patterns to improve your catalog's usability.

Design Process

- You must thoroughly document your design and planning process.
- Your design journey should show the evolution of your design.
- All images in your design journey must be visible in VS Code's Markdown Preview.
- All images must be labeled in your design journey. Labels must be visible in VS Code's Markdown Preview.

4. Website

Planning

- Plan and design your website before coding it.

Functionality

- 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. Seriously, you aren't Google. Databases are good for basic search. If you want *Google-like* search results, you need to use a search engine like Elasticsearch. Search engines are prohibited on this assignment.
    - **Search must be implemented with 1 database query.**

      Algorithmically implemented search in PHP is prohibited; you may not retrieve records from the database and then use a loop in PHP to 'search'.

      Your search must be *entirely* implemented with 1 SQL query. You must take the results of that query and echo them to the screen.
    - 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.

      Note: 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.

Database

- All catalog content must be retrieved from the database. You may not *hard-code* any catalog content in HTML/PHP.
- Database must be named **secure/catalog.sqlite**.
- Your database (*catalog.sqlite*) **must** be committed and pushed to GitHub.

  Seriously. We cannot grade your project 2 without a database. If you "forget" to commit and push your database, you will likely get a 0 for this assignment. It is your responsibility to check your submission.

  Tip: You may check your submission by re-cloning your repository into a *different* folder on your computer and try and run your website. This is **exactly** how the TAs grade your project!
- You are required to use PHP's PDO extension to the database.

  Any other access method is prohibited and will result in a substantial point deduction.

Security

- Your code should be secure against database injection attacks and cross-site scripting attacks.
- **Your code should filter input and escape output.**

Implementation Constraints

- Your code must make effective use partials.

  If your site is 1 page, you probably don't need any partials. Use partials if it makes sense to include them!

- Your code must make effective use of functions. **Minimum of 2 user-defined functions required.**

  You are required to write 2 user defined functions. These functions, like everything in this project must be 100% your own original work.

  Hint: You might want a function to *echo* a record to the screen.

- 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 forms are not required to be sticky.

  To improve the usability of your forms you should make them sticky. However, for this project we'll skip this requirement in the interest of time.

- **init.php** must be included on all pages.
- Remember, cite all images according to the course policy. This includes images associated with the database.
- You may not use an existing site; project 2 should be created from scratch.

5. 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 any 2300 student. 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. Test your add entry and search forms extensively.

- 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.

Milestone 1: Plan your Catalog

Complete the sections labeled "Milestone 1" in the design journey. Plan your catalog, your catalog's target audience(s), and explore design patterns for catalogs similiar to yours.

Note: You need at least 1 target audience but you may have more depending on your catalog/site.

Milestone 2: Design, Plan, & Draft Website

Design and plan your website and database in the design journey; complete the "Milestone 2" sections of the design journey. Code a draft version of your website.

1. Plan your Site

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

2. Plan your Database Schema

Plan out your database schema in the design journey.

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. 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 in the design journey. 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;`

4. Populated Database/Seed Data

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

IMPORTANT! 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.

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 partials, forms, etc.) should mostly be done.

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 **Reflection** in the design journey.

Reminder: All design/planning images must be visible in your design journey when using VS Code's 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.