Assignment 4 - Database Powered Front Ends

Due Dec 8, 2023 by 11:59p.m. **Points** 100 **Submitting** a file upload
File Types txt **Available** until Dec 16, 2023 at 11:59p.m.

This assignment was locked Dec 16, 2023 at 11:59p.m..

Please note that all assignments must be submitted both to CSUNIX and also to Canvas. Except for the filename difference noted below, your files in both submissions must be identical or your work will be graded 0.

REMEMBER THAT THE DATABASE MUST BE INSTALLED ON CSUNIX FOR YOUR CODE TO WORK CORRECTLY. You may have to change your source code credentials in order for this to work if your XAMPP credentials are using the default values.

Please follow the submission instructions carefully to prevent a grade of 0 for incomplete submission.

Do not edit any of the starter code that has been provided. Your server side solution must interact with the provided code correctly.

For this course programs like ChatGPT or Co-pilot will be treated exactly the same as code copied from an internet source. Any evidence of their use will be treated as a level 2 academic integrity violation.

Purpose

The purpose of this assignment is to gain experience creating a database driven application using PHP server side code. The exercise is intended to demonstrate how even a small amount of database driven support can produce significant web site experiences for our users. See
s://csunix.mohawkcollege.ca/~adams/10260/a4/
(https://csunix.mohawkcollege.ca/~adams/10260/a4/) for a working example of the final product.

Starter Code

Please download the **starter code for assignment 4**
(https://mycanvas.mohawkcollege.ca/courses/92989/files/17769358?wrap=1)
(https://mycanvas.mohawkcollege.ca/courses/92989/files/17769358/download?download_frd=1) . As indicated previously, we aren't assessing your abilities in producing client side code this term, and we want you to engage fully with the server side code. As such you are forbidden from altering the front end code. If we download your solutions and put it with the starter code file, it must work without alteration. Similarly we have provided you with the starter code to create the needed database tables and insert the sample data. You are not permitted to alter this database.

To help you understand whether this is just some academic nonsense or if it's a real world restriction that could occur: during team programming one of the first steps is to agree on an interface which allows the teams to work in parallel without frequent interactions. Neither team is permitted to alter the interface once it is created. If a change is required, it must come from the project manager/team lead, which in this setting is your professor. This is a pattern that occurs in many kinds of programming and allows for teams to build software in parallel much quicker than an individual could generate the code. It also allows for the creation of standard tests at the time the specification is written so that we can validate that the created code meets the requirements. If we are going to strive to help you improve your skills as you move through the program with the goal of a professional level of practice, it feels reasonable to set up a restriction such as this.

Getting Started

Please review the **documentation standards (https://mycanvas.mohawkcollege.ca/courses/92989/pages/documentation-standards)** for the homework in this course. All PHP files must follow this standard.

Download the starter code and extract the files to your /xampp/htdocs/10260/a4 folder (or the equivalent on your development machine).

Create a new file called **me.php** that outputs your name and student number and place it in the same folder. Remember to apply the documentation standards even to this file.

Question 1

You will be creating the server side support for an "infinite scroll" web page. Call your solution `quotes.php` On page load and at specific times when the user is scrolling on the page, an event will be triggered that causes an AJAX query to be sent to the server. Note that the AJAX call passes a parameter called "page" - be sure not to trust it. Page must be a positive integer.

Traditionally we would use MVC principles and simply return a JSON encoded array with the values from the database, but in order to give you an opportunity to exercise some personal discretion in the design you will be building an array of HTML elements, specifically BootStrap 5 Cards, as an array of strings. Each card will display the quotation and the author name. Put the code needed to generate each card in a function that returns the finished string.

Each one of your strings will look something like: `'<div class="card mb-3 a4card w-100"><div class="card-header">Quote Author</div><div class="card-body d-flex align-items-center"><p class="card-text w-100">Quote Text</p></div></div>'`

(Repeated here for more clarity)

```
<div class="card mb-3 a4card w-100">
     <div class="card-header">Quote Author</div>
     <div class="card-body d-flex align-items-center">
          <p class="card-text w-100">Quote Text</p>
     </div>
</div>
```

You are not required to make the cards colourful or to embellish them, but you are very welcome to do so if you choose. You do not have to keep the author on the top, you are welcome to modify the card as you see fit except that you are required to not break the responsive design that has been laid out already. You may choose to make the cards narrower or to change the number per row, but at least 3 cards per row are required when the webpage is at its full size.

Your script will output the JSON encoded array of these cards to the AJAX call. Note: you should be returning a maximum of 20 quotations at a time. You may hard code this limit. You will need to create a way to skip quotations that are on previous pages. The formula `(pages - 1) * limit` should help you.

Similarly you will need a query in order to retrieve information from the database. This query should be helpful:

```
    $query = "SELECT quotes.quote_text, authors.author_name
    FROM quotes
    JOIN authors ON quotes.author_id = authors.author_id
    LIMIT :per_page
    OFFSET :offset
    ";
```

All PHP code should use the best practices demonstrated in this course including validation, sanitization, and PDO database access. Assignments that do not use the required PDO format will be graded 0.

As always, if there is any discrepancy between the instructions and the sample solution, the instructions are assumed to be correct. I encourage you to ask any questions you may have to the mycanvas discussion topic.

Submission Instructions

Programs are meant to be run, not read. Therefore in this course you must submit your work to CSUNIX so that your instructor can grade the functional performance of your work. This assignment must be uploaded to your **public_html/private/10260/a4** directory on CSUNIX (you will probably have to create this directory). If you submit this to the wrong folder it will be treated as though it was not submitted. Searching around your directory structure to find the folder is not a good use of your professor's assignment grading time. If your professor cannot find your work on CSUNIX you will be given 1 week to correct the folder name, and to email them to alert them that you have fixed the problem. After the 1 week period the grade of 0 will be permanent.

Your program must also be submitted to Canvas. First and foremost this establishes a time that you submitted your work that we can all agree on. Secondly it allows your work to be inspected for plagiarism. You will not be judged based on the simple score from TurnItIn, but on your professor's interpretation of whether the identified code is common and reasonably should appear in many student's work (i.e. **<?php** will show up a lot) or whether the code suggests copying from an unauthorized source. We are suggesting you shouldn't fully trust AI tools, and we won't either.

You will submit your files to Canvas in the following way:

For each .php file (not the provided starter code):

- Copy the file and add .txt to the end of the filename. For example: a4.php gets copied to quotes.php.txt
- Upload the .txt files to Canvas

Do not archive (zip) your files, or upload files that are not requested.

You must submit **me.php.txt**, **a4.php.txt**, and **any other helper php files (like db-connect.php.txt)** to Canvas and uploading the starter code and your .php files to CSUNIX for credit.

REMEMBER THAT THE DATABASE MUST BE INSTALLED ON CSUNIX FOR YOUR CODE TO WORK CORRECTLY. You may have to change your source code credentials in order for this to work.

10260 Assignment 4 Rubric

Criteria	Ratings				Pts
me.php as required me.php is present, provides the required information, and is correctly documented.	**5 pts** **Full Marks** me.php provides the correct information and is well documented.	**3 pts** **Satisfactory** me.php provides the correct information but is not correctly documented.	**2 pts** **Poor** me.php provides incomplete or incorrect information but some effort is present.	**0 pts** **No Marks** me.php is not submitted, does not meet the technical requirements, or is not available on csunix	5 pts
question 1: non-technical merits File name is as specified, file header is present and as required, functions are documented as required. Client side starter code files and support files are unmodified.	**20 pts** **Excellent** Solution is well documented and has not modified the starter code. The solution is named as required.	**12 pts** **Good** The solution's documentation is incomplete or missing. Solution is named as required and the starter code is not modified.	**8 pts** **Satisfactory** Solution is well documented but has modified the starter code.	**4 pts** **Poor** The solution is insufficiently documented and has modified the starter code.	**0 pts** **No Marks** Solution is missing, inconsequential, or not hosted on CSUNIX.	20 pts
question 1: technical merits Code is written in a manner that is consistent with course instruction and includes appropriate levels of commenting, modularity,	**75 pts** **Excellent** Code contains appropriate internal documentation, modularity, and error handling. Course concepts such as validation and sanitization are appropriately	**45 pts** **Good** Code contains internal documentation, modularity, and error handling but could be improved. Course concepts such as validation and sanitization are	**30 pts** **Satisfactory** Code may not contain appropriate internal documentation, modularity, and/or error handling. Course concepts such as validation and	**15 pts** **Poor** Code may not contain appropriate internal documentation, modularity, or error handling. Course concepts such as validation and sanitization	**0 pts** **No Marks** Solution is missing, inconsequential, or not hosted on CSUNIX.	75 pts

Criteria	Ratings					Pts
and error handling. Question functionality is correctly implemented and produces correct outputs.	integrated. Functionality is correctly implemented. Starter code is not modified.	appropriately integrated. Functionality is correctly implemented. Starter code is not modified.	sanitization may not be fully integrated. Functionality may contain minor errors but is mostly correctly implemented. Starter code is not modified.	may not be appropriately integrated. Functionality contains significant deviations from the requirements. Starter code may have been modified to accommodate errors.		
					Total Points: 100	