Assignment 3 - Managing Server State

Due Dec 1, 2023 by 11:59p.m. **Points** 100 **Submitting** a file upload **File Types** txt
Available until Dec 4, 2023 at 11:59p.m.

This assignment was locked Dec 4, 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.

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 expand on your knowledge of web applications and use new techniques we have learned to maintain state within a web application. In this assignment you will need to utilize many of the programming methods we have learned so far this course in addition to using SESSION superglobal to maintain the state of a user interaction. Specifically, you will be sanitizing inputs, performing file I/O, using modular programming techniques and maintaining the state of a user interaction.

Starter Code

Please download the **starter code for assignment 3 (https://mycanvas.mohawkcollege.ca/courses/92989/files/17726915?wrap=1)** **(https://mycanvas.mohawkcollege.ca/courses/92989/files/17726915/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. 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/a3 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 implementing the game of **NIM (https://en.wikipedia.org/wiki/Nim)**. NIM is a two-player strategy game where users remove objects from distinct piles. For this assignment, we are implementing a simple version of the game with only a single pile, and during a turn a player/computer can take 1, 2, or 3 stones from the pile. The player alternates turns with the computer (PHP script) with the player making the first turn. Historically, NIM is played as a misère game where the player/computer that takes the last item (stones in our case) from the pile is the loser of the game.

Design, implement and test a PHP script named **nim.php** that takes the following input (GET) parameters: mode, difficulty, count, player_move.

- mode(optional) – if set to 0 the game resets and accepts the initial parameters of difficulty, and count, otherwise the game is being played and difficulty and count are ignored;
- difficulty – 0 means random guess, 1 means optimal play (more on that later)
- count – the initial count of stones used in the game, default to 20;
- player_move – the number of stones the player is removing – only considered on the player's turn, ignored otherwise.

To determine the optimal move, take the **number of stones remaining modulo 4**, and pick a move that makes the result after you pick to be 1. For example: If there are 15 stones, 15 mod 4 = 3. You want to take 2 stones (13 mod 4 = 1). Since we are only working with 1 pile you can use the following logic:

- if the remainder is 3, take 2 stones
- if the remainder is 2, take 1 stone
- if the remainder is 1 – its impossible to make the remainder 1 while taking between 1 and 3 stones, so take a random number of stones

- if the remainder is 0, take 3 stones

Since the script is being called through an AJAX request, your PHP script will need to provide a JSON encoded array with the following properties: move, stones, player, winner.

The properties are defined as follows:

- move – integer – describes the number of stones the player/computer is taking during the turn
- stones – integer - describes the number of stones remaining at the end of the turn
- player – string – the entity who is making the current move
- winner – string – a sentence describing if a winner has been determined or if the game is in progress.

An example solution is available at: **https://csunix.mohawkcollege.ca/~lovie/10260/a3/a3-q1.html (https://csunix.mohawkcollege.ca/~lovie/10260/a3/a3-q1.html)**

As always, in the event of a difference between this document and the sample solution, this document is the "correct" version. If you have any questions about this question, I would encourage you to post a reply to the discussion topic on mycanvas.

Question 2

You will be implementing the game of **Hangman (https://en.wikipedia.org/wiki/Hangman_(game))** . Hangman is a single player game where the player competes against the computer to guess a hidden, random word,

Design, implement, and test a script named **hangman.php** that accepts an AJAX GET request with the following parameters: mode, letter.

The parameters have the following meanings:

- mode – string – optional – if mode is “reset” the game resets and a new word is chosen
- letter – string – the letter that is being guessed, ignored if the mode parameter is provided, treat the letters as case insensitive (upper and lower case are accepted the same, e.g. A==a)

The script provides a JSON encoded array with the following values to the client side: guesses, alphabet, secret, strikes, status.

The parameters have the following meanings:

- guesses – string – a sorted list of the previous guesses made
- alphabet – string – a list of the remaining letters available to guess - for this assignment, standard English letters are valid
- secret – string – the secret word with all unguessed letters blanked out (_)
- strikes – integer – the number of incorrect guesses so far

- status – string – a sentence describing the status of the game

You must use the wordlist.txt file to provide the PHP script with available random words, one of which is chosen as the secret. No word should be repeated for a given game session (playing multiple games with the reset button between games - no long term tracking required). Do not modify wordlist.txt in any way. You will need to keep track of the incorrect guesses the user makes, the game should end once 7 incorrect guesses are made, and the script should reveal the word and tell the user that they have lost the game. If the user manages to guess all the letters of the word before running out of strikes, then the user should be told they won the game.

The sample solution is located here: **https://csunix.mohawkcollege.ca/~lovie/10260/a3/a3-q2.html (https://csunix.mohawkcollege.ca/~lovie/10260/a3/a3-q2.html)**

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/a3** 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 .html starter code):

- Copy the file and add .txt to the end of the filename. For example: question1.php gets copied to question1.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**, **nim.php.txt**, and **hangman.php.txt** to Canvas and uploading the HTML starter code and your 3 .php files to CSUNIX for credit.

10260 Assignment 3 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.	**7.5 pts** **Excellent** Solution is well documented and has not modified the starter code. The solution is named as required.	**4.5 pts** **Good** The solution's documentation is incomplete or missing. Solution is named as required and the starter code is not modified.	**3 pts** **Satisfactory** Solution is well documented but has modified the starter code.	**1.5 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.	7.5 pts
question 1: technical merits Code is written in a manner that is consistent with course instruction and includes appropriate levels of commenting, modularity, and error	**27.5 pts** **Excellent** Code contains appropriate internal documentation, modularity, and error handling. Course concepts such as validation and sanitization are appropriately	**16.5 pts** **Good** Code contains internal documentation, modularity, and error handling but could be improved. Course concepts such as validation and sanitization are	**11 pts** **Satisfactory** Code may not contain appropriate internal documentation, modularity, and/or error handling. Course concepts such as validation and	**5.5 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.	27.5 pts

Criteria	Ratings					Pts
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.		
question 2: 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.	**15 pts** **Excellent** Solution is well documented and has not modified the starter code. The solution is named as required.	**9 pts** **Good** The solution's documentation is incomplete or missing. Solution is named as required and the starter code is not modified.	**6 pts** **Satisfactory** Solution is well documented but has modified the starter code.	**3 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.	15 pts
question 2: technical merits Code is written in a manner that is consistent with course instruction and includes appropriate levels of commenting, modularity, and error	**45 pts** **Excellent** Code contains appropriate internal documentation, modularity, and error handling. Course concepts such as validation and sanitization are appropriately integrated.	**27 pts** **Good** Code contains internal documentation, modularity, and error handling but could be improved. Course concepts such as validation and sanitization are appropriately	**18 pts** **Satisfactory** Code may not contain appropriate internal documentation, modularity, and/or error handling. Course concepts such as validation and sanitization	**9 pts** **Poor** Code may not contain appropriate internal documentation, modularity, or error handling. Course concepts such as validation and sanitization may not be	**0 pts** **No Marks** Solution is missing, inconsequential, or not hosted on CSUNIX.	45 pts

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