Project 3 – Login System with PHP

Overview

PHP is a programming language commonly used to create dynamic websites. Your task is to create a basic website that requires the user to log in using a username and password. Once logged in, there are various pages with various features. Finally, it should have a way to logout.

Pages

Your project will be a website made of various pages as described below. All of these pages require that the user be logged in. If the user attempts to visit any of the pages and isn't logged in, you should redirect them to the login page.

Your project should contain *at minimum* the following pages. Every page (except for the login and menu page) should have a link back to the main menu page.

index.php
The main page. This page should act as a menu, displaying links to the other pages below as well as a link to the logout page. If the user isn't logged in, this page should redirect them to the login page.

repeater.php
This page should display a sentence of your choosing a random number of times (lets limit to 500 repeats max). The number of times the sentence appears should be different every time they visit this page. Requires login.

date.php
This page should display the current date and time in MM/DD/YYYY HH:MM AM/PM format. So, for example, a properly formatted date would be 10/31/2017 5:37 PM or 12/25/2017 11:32 AM. Requires login.

rps.php
This page should recreate the game of "Rock Paper Scissors". It should have three radio buttons on it: "Rock", "Paper", and "Scissors" and also a "Submit" button. When the user clicks submit, it should submit their choice to playrps.php. Requires login.

playrps.php
This file should randomly pick "Rock", "Paper", or "Scissors" as the CPUs selection. It should then display the results of the game. Example output would be "Player: Rock. CPU: Scissors. Result: Win". If the user didn't submit a choice, an error should be displayed. Requires login.

logout.php
This page logs the user out and then redirects them to login.php

login.php
This is where the user may sign in using a username and password. If they're already logged in, redirect them to index.php. If they type in the incorrect username/password combo, display the login page again with an error message.

Creating a custom class

PHP is an object oriented language. This means that you may write a class once but use it on multiple pages. Your task is to create class named UserAuthenticator in a file named 'UserAuthenticator.php'. This class should contain all of the logic for logging in, logging out, and other features. Each of your pages should then use this UserAuthenticator class for their login related logic.

Below is a UML class diagram which describes the UserAuthenticator class. It MUST have all the methods as described in the UML class diagram. You can add more methods if you'd like.

UserAuthenticator
+isLoggedIn() : Boolean +authenticate(username : String, password : String) : Boolean +logUserIn(username: String) +logout() +redirectToLogin()

isLoggedIn: Returns true or false depending upon if the user is logged in.
authenticate: Accepts a username and password. If they're correct, returns true. False otherwise.
logUserIn: Creates the session record necessary to log the user in.
logout: Logs the current user out.
redirectToLogin: Redirects them to the login page.

Hints

- You can use the $_SESSION to keep track of the currently logged in user.
- PHP has a really useful class, DateTime, for getting and formatting times. You should use this.
- You may want to have constants in your UserAuthenticator for the correct username and password.
- You can redirect the user to another page by sending special headers using the header() method in PHP. You must call this method BEFORE printing any output or else it won't work.
- Ask me for help! Seriously, people have crashed and burned on this project before and then were amazed at how simple it could be once they asked for help.

Suggestions

Below are some suggestions to help keep your project organized

- Separate your logic from the pages. For example, all the logic could be in one PHP file, and then all the HTML and page layout could be in a separate PHP file that the first one includes.
- If you have a common menu on all pages, you could put that in a separate file and include it on all pages.
- Use a single CSS file to style the entire website. If you're wanting a challenge, make your website responsive, meaning it will displays well on small phone screens.

Deploying to Thor

Your project three should be uploaded to the CS web server, Thor, in a 'p3' directory. You should update the landing page of your website with a link to project 3.

It is important to make sure your site works on Thor! This is where I will be testing them, so if your site isn't at the correct path and doesn't work on Thor, you will lose points!

Grading

The assignment is worth 150 points. You will be graded on the following criteria:

Has all required pages	10
UserAuthenticator class meets requirements	20
Log in, log out, and redirect work	25
Repeater page works as described	15
Date page works as described	15
Rock/paper/scissors page works as described	25
Site works on Thor	10
Website is styled with CSS and looks appealing	20
Header comment block on all files	10
Total	**150**

Submitting your work

Your work should be submitted as an archive (.zip, .7z, .rar) on Moodle. This archive should include your all files needed for your project. Late submissions are allowed up to a week after the due date, but have an automatic 20% penalty.

This project is due Thursday, November 14th at 11:59pm.