**Lab Assignment 04 - PHP User Authentication/ Registration and Web Activity Logging**

**Objectives**

The objective of this assignment is to build a multi-page responsive web application using a Bootstrap template created/modified in Lab 3, using PHP and PostgreSQL to register new and authenticate existing students on our website, while capturing web activity in flat files.

NOTE: for this assignment students ***ARE NOT*** allowed to use any Generative AI (including but not limited to ChatGPT) to create assignment pages or PHP scripts except for the privacy policy, acceptable use and cookie policy pages (this is stated explicitly below). If students submit any other content that they did not create individually they will be written up for academic dishonesty.

**Requirements**

1. **PHP Style Guide Adherence**

Ensure that the PHP code adheres to the style guidelines outlined in INFT2100\_PHP\_CourseStyleGuide\_v2024.pdf, following conventions for naming, indentation, and commenting.

NOTE: at a minimum each \*.php file created/submitted for each and every file is to have file-level /\* PHP comments \*/

1. **Changes to header.php/footer.php/functions.php**

* All pages created for this lab should implement the bootstrap layout (using header.php and footer.php) and function the same as when they were setup in lab 3, including dynamic <!-- HTML Comments --> page <title>.
* The header.php should include code the starts a session (i.e. calls session\_start(); and ob\_start(); )
* The nav bar in the header.php should have a static link to the home page (whether someone is logged in or not)
* There should be dynamic links in the nav bar that shows the following if someone IS NOT logged in/authenticated:
  + Login (goes to login.php)
  + Register (goes to register.php)
* These links should be replaced with the following if some IS logged in/authenticated
  + Dashboard (goes to grades.php created in lab 3)
  + Log out (goes to logout.php)

NOTE: to be able to determine if a student is logged in, check the $\_SESSION to see if user data has been placed there by the login.php page

1. **Changes to functions.php**

* Creates a new, appropriately named pg\_prepare()’ed statement that takes a single parameter for the user/student id used to SELECT user and student information from the users and students table
* Creates a new, appropriately named pg\_prepare()’ed statement that takes a single parameter for the user/student id used to UPDATE the users table so that the last\_access field is set to the current time
* Creates two (2) new, appropriately named pg\_prepare()’ed statement that takes the inputted data from the register.php form and INSERT records into both the users and students tables.
* NOTES:
  + the id should be auto-generated in the users table, that id should be then retrieved and used to insert the student information into the students table
  + The enrol\_date and last\_access field in the users table should be set to the current timestamp (when the record is inserted). This can be achieved either on the database or programmatically
  + all pg\_prepare() function must exist in the functions.php file described below
  + There SHOULD BE NO pg\_query() function calls anywhere on your website

1. **Login/Authentication Functionality (**login.php**)**

* Checks to see if a cookie exists for a user id for the person that last logged in, and if it does pre-load it into the login form (make it stick in the form)
* Has a "self-referring" form that submits back to itself in POST mode with a text input for the student id and a password input for the student’s password.
* If the page loads in POST mode (i.e. someone has submitted the form), the input for the user id and password should be retrieved from the form (removing any leading or trailing whitespace using the trim() function).
* The entered values should be passed to a prepared statement that takes a user id (see functions.php requirements above).
  + If no record is returned (i.e. there is not the entered id), the user should remain on the page with a message stating they were not successfully logged in should be displayed.
  + If a record is found (i.e. the id exists in the users/students tables):
    - The entered password should be verified as the bcrypt hash of the value stored in the database using the password\_verify() command
    - If the password is correct the user record should be retrieved from the result set and placed onto the session as a user
    - the users database table last\_access field should be set to the current timestamp for the user that logged in using the prepared statement created above in the functions.php requirements
    - a cookie name LOGIN\_COOKIE should be created with its expiry date to be thirty (30) days in the future (i.e. time() + 60\*60\*24\*30; )
    - the user should be redirected to the grades.php page where their personal information is displayed. i.e. first\_name, last\_name, email\_address, enrol\_date and last\_access (the one retrieved as part of the authentication, NOT the latest one after the UPDATE).

1. **Logging out Functionality (**logout.php**)**

The logout.php file should perform the following:

* unset the session (session\_unset())
* destroy the session (session\_destroy())
* restart the session (session\_start())
* place a message stating the user has successfully logged out onto the session:

$\_SESSION[‘message’] = “You have successfully logged out”;

* redirect the user to the login.php page:

header(“Location: ./login.php”);

ob\_flush();

* where the message should be displayed and then removed from the session:

$msg = isset($\_SESSION[‘message’])?$\_SESSION[‘message’]:””;

$\_SESSION[‘message’] = “”;

1. **Grades Page Modifications (**grades.php**)**

* The grades.php page should be modified so that only a student that is logged in (i.e. there is user information set on the $\_SESSION when they logged in) are able to access the page.
* If the student is not logged in they should be re-directed to the login.php page with a message that says they must log in to access their grades.
* Otherwise they should stay on the page, with their personal information being on displayed and their grades table being generated. NOTE: for this lab the id should be grabbed from the <form> that was POSTed versus from the URL (i.e. in GET mode) for lab 3.

1. **User Registration (**register.php**)**

* The register page(s) (for registering system users and personal information for both agents and clients) can either be a single page (capturing all of system user's authentication and personal information on one page), or you can have two separate pages: one for user creation and then a separate page for their personal information. The register.php page should perform the following:
  + verify that no one is currently logged in when they access the page (i.e. if there is a user logged on they should be redirected to the grades.php with a message that is placed on the session that informs the user that they are logged in, and cannot login again)
  + contain a self-referring form that is set up to be "sticky" (except for the password/confirm password inputs)
  + Have appropriate and appropriately named <input> tags for email address, first name, last name, password/confirm password, birthdate and a radio button for their Program (CPGA or CPPG)
  + Includes appropriate data validation for each of the fields (i.e. checks that mandatory fields contain data, emails entered are valid, password/confirm password exists AND are not too long or too short, that first/last names exists and are not too long.
* If there are any problems with the inputted data, error message(s) should be generated (including invalid data except for password problems) and the invalid data should be removed from the form. Valid data should remain (i.e. make the form “sticky)
* If the data is is valid, the page should insert a new user record, grab the new id from the Users table and insert a new student record. NOTES: these INSERTs should be prepared statements in functions.php and the plain text password should be bcrypt hashed before inserting (using the PHP password\_hash() function). The activity of enrolling a student should be capture in the web activity log.
* Once a new student in enrolled, they should be redirected to the login.php page and shown a success message and a prompt to log in.

1. **Flat file logging of Web Activity (**activity.log**)**

* Add the functionality that a detailed, time-stamped message is appended to an activity.log file (in a logs sub-folder when):
  + A new user successfully registers
  + A user successfully logs on
  + A user attempts to log on unsuccessfully
  + A user explicitly logs off (using the logout.php link)

1. **Acceptable Use and Privacy Policy Pages (**privacy\_policy.php and aup.php)
   * Create new privacy\_policy.php and aup.php (Acceptable Use Policy) pages
   * These pages should contain appropriate content that describes your privacy and acceptable uses for your fictional website.
   * You should place links in the footer.php file, so they wil be available on each and every page (whether someone is logged in or not.
   * NOTES:
     + This is the only section of the lab that you can use generative AI or a template tool.
     + Be sure to add <!-- HTML Comments --> on each page capturing how the pages were generated
2. **Publishing Web site onto the** opentech **Server**

Publish the completed \*.php files onto the opentech server (pages to be placed in your /var/www/html/inft2100/userid folder), ensuring that the submit pages connect to and retrieve data from students’ individual userid\_db PostgreSQL database (created and populated in Lab Assignment 2).

Your file structure should look like:

userid

├── aup.php

├── grades.php

├── includes

│   ├── footer.php

│   ├── functions.php

│   └── header.php

├── index.php

├── login.php

├── logout.php

├── logs

│   └── activity.log

├── privacy\_policy.php

└── register.php

NOTE: if the opentech database is missing the users, students, courses and grades tables from Lab Assignment 2 and used in Lab Assignment 3, most of the pages created for this lab will not function correctly, and the mark achieved for this assignment will reflect this fact.

1. **Links and Authentication Credentials in DC Connect**

Submit a link to your website in the DC Connect assignment drop box. Be sure to provide credentials (i.e. id/password pairs) for an existing a student with grades and an existing student without grades.

Your dynamic nav bar should provide links for your instructor to register a new user, if this does not work, provide a clickable link to your published register.php page

**Deliverables**

Submit the following items to DC Connect:

1. Zipped file of the website folder structure, including PHP, CSS, and SQL files.

2. Clearly identified and clickable links to your home page and to your grades page demonstrating the various student id scenarios (listed above) in the comment section of the assignment dropbox.

**Submission Guidelines**

Submit the completed assignment as a single zip file to the DC Connect assignment drop box after publishing and testing them on the opentech server. Ensure all files adhere to the PHP style guide provided in DC Connect.