Working on Project towards completion of understanding PHP by Ullman

Project Brief: Task 1 – Login and Display User in a Table based on owner

You were asked by an old age home care centre for Alzheimer patients to write an application to help them with their administration. You need to create a web application to allow carers within the facility to select certain products that are needed for the elderly.

These products can be added by staff, as well as family can add products/requests which they bring into the facility. The details of that patient will be recorded with a photo for identification in the patient table. The details for the users will be stored in the users table.

Process:

The application must be done as an RWD or Responsive Web Design application, as you have have done for the login page already. A subset of bootstrap CSS is also found within Ullman textbook.

Details for Project

The Project is broken down into FOUR tasks that must function as one web application; for you to demonstrate your PHP scripting abilities and to demonstrate your learning from the previous tasks. You are required to study Chapters 1–13 in the Ullman textbook. In other words, ensure that you demonstrate your understanding of:

- PHP scripting;
- Functions and control structures;
- Manipulation of strings;
- Handling of user input;
- Manipulating arrays;
- Working with databases and MySQL;
- Manipulating MySQL Databases with PHP
- Managing State Information (If time permits dictated by learning)
- Object Oriented PHP (If time permits) Optional

Please note that when completing your four tasks, you are required to use good coding standards. Please refer to the marking rubric which indicates how your coding will be assessed.

You are required to document your development by reflecting on:

1. **Explanations of shortfalls (lessons learnt) for:**

   a. Variable naming;
   b. Comments/ code readability throughout PHP code;
   c. PHP file naming (modular coding);

2. **User friendly design in:**
   a. Error handling and data validation using HTML5.
   b. Integration of member functions and objects
   c. Shopping cart data structure user experience
   d. Correct structuring of database scheme with tables

3. **Pros and cons you experienced in working with PHP**

Task 1: Start during Session 14 (22 Oct) and must be completed after Session 16 (29 Oct)

1.1 Complete the exercises set out in Ullman and **as needed**. The goal is to understand and implement a sticky form, All-in-One form using the $_SERVER["SCRIPTNAME"] autoglobal or the scriptname. Use that in conjunction with an assosiative array, read from the database and ensure PHPMyAdmin is used to create tables and export the database as an SQL file. Please consult Ullman on creating a template using HTML5 folder structure (images, css, js) to abide by the rules and use HTML5 for validation purposes as set out in https://www.wufoo.com/html5/ when creating your form.

1.2 Style your login page, using a CSS framework. Reference Ullman Quickstart on Chapters 3, 4, and 8 for responsive web design (RWD). You may also use Bootstrap CSS instead of Ullman's Concise CSS (http://concisecss.com), but Chapter 8 (Ullman) covers example code in detail.

1.3 Create a login form achieving the last exercise when a form calls itself with MENU options at top.

1.4 Do exercise in reading a text file with proverbs into an array, creating a table within a database and writing those proverbs to the table using PHP and MySQL. Ensure the connection script is embedded in an include file, e.g. dbConn.php.

1.5 **SUGGESTED APPROACH**

1. Create a simple login form
2. Create a Login form calling another script (Two Part Form)
3. Create a login form calling itself
4. Create a Login form calling itself and incorporating a Sticky form.

Task 1 — Login Validation and Table creation

Your task must meet the following specifications:

1. Create a table ***tbl_User*** and a table ***tbl_Patient*** in MySQL using the console or phpMyAdmin, consisting of the following column names:

a. The table structure for USER is as follows for ***tbl_User***:
   ID (autoincrement - PK), FName (text), LName (text), adress1, address2, Postal Code, Email (text), CellNum (Text), Password (Text), userImage (Text or BLOB), Status (user/admin/matron)

b. The table structure for PATIENT is as follows for ***tbl_Patient:***
   ID (autoincrement - PK), FName (text), LName (text), Roomno (text), Password (Text), NextOfKinID (in User Table), adress1, address2, Postal Code, Grade-Classification (A, B, C), prescript, patientImage (Text or BLOB)
   The NextOfKinID for staff will be NULL for they can access every patient

c. Create text files ***userData.txt*** and ***patientData.tx***t and populate the textfiles with at least **30 fictitious** entries, e.g. John Doe j.doe@abc.co.za 29ef52e7563626a96cea7f4b4085c124

d. Use the console or phpMyAdmin and load the text files manually into the table.

2. Use the existing ***test*** database that is pre-installed with wamp or xamp.

   a. Create a connection within the ***test*** database using PHP and store the code which creates a connection in a file called ***DBConn.php***.
   b. Create a script called ***createTable.php*** that will check if the table ***tbl_Patient*** *and the table* ***tbl_User*** exists and if it does, delete the table and (re)create the table and load the data into the table using the ***patientData.txt*** file as source file and **userData.txt** as source file.
   c. Embed the ***DBConn.php*** as an include file within the ***createTable.php*** script. Hint: You may include the connection code in the createTable.php directly and later refactor the code into an include file to modularize your code.
   d. Each time the script is run the table will be deleted if it exists and reloaded afresh with the data stored in the text file.

3. Create a login page for your project. The login page must:

   a. Accept a users's name, surname, email address (of next of kin) and password(of next of kin).
   b. The password must be compared to a e.g. hash "29ef52e7563626a96cea7f4b4085c124" in the ***tbl_User table***. The correct password is: "P@55w0rd!" Note: You may NOT hard code the hashed value in your PHP code.
   c. When clicking the submit button, use HTML5 to validate. Textboxes and the password from the login details must be compared to the stored hashed password value in the MySQL database.
   d. If the validation satisfies the password, then display the users's data using an associative read approach regarding the column names in a table. However if the

password is incorrect then use a sticky form and redisplay the details entered allowing the user/data administrator to edit the fields instead of re-typing all the fields. Display a string at the top of the page that identifies the user and reads: "User John Doe is logged in"

e. If the user does not exist, the administrator or next of kin can register him/herself create the hash and login.

Note that the administrator/matron will have a super password that will allow him/her to edit any patient data in patient table. The designated sisters will have access to add items but not edit user data.

The next of kin may also log in with viewing options only. To see levels of items like diper levels, maybe sheets, shoes etc.

4. The pictures of the users and staff must be stored in a folder ***images***. Take care of naming these jpg files for the ID must pick up the picture when these users are displayed in a table. You may use another way of storing the name of the image as part of the table as a BLOB.

5. When the user starts the index.php script, the user is prompted to log into the system. He/she must be verified and then be greeted as Welcome John Doe. You are a visitor/admin/Matron staff

SUGGESTED APPROACH

1. Create a simple login form
2. Create a Login form calling another script (Two Part Form)
3. Create a login form calling itself
4. Create a Login form calling itself and incorporating a Sticky form.