Practical Assignments – 3

Objective: Building a user registration system using PHP and MySQL.

Tools and packages Used:

1. Visual Studio Code as Text Editor.
2. HTML, CSS and JavaScript for Frontend.
3. Node to install tailwind.
4. Tailwind CSS as CSS Framework.
5. PHP for backend scripting.
6. XAMPP as a development environment for MySQL.

User Management System:

In this project, we build a secure user registration system that handles user data (username, email, etc.) with full CRUD support. It features session management, role-based access control, and security measures like password hashing and input validation to protect user information and restrict access to authorized users only.

Database setup:

Database Name: user_management
Table Name: users
Table Structure:

- Column 1: user_id - INT, Primary Key, Auto-increment
- Column 2: username - VARCHAR (50), Unique
- Column 3: email - VARCHAR (255), Unique
- Column 4: password - VARCHAR (255)
- Column 5: created_at – TIMESTAMP, Default CURRENT_TIMESTAMP

Project's Folder Structure:

Folder/File	Description
node_modeules	Dependency folder (auto generated).
Package.json , package-lock.json	Project metadata and dependencies.
src/	Main source folder.
src/server.php	Contains globally used variables and tasks
src/output.css	Auto generated CSS file by tailwind CSS
src/input.css	Custom CSS file
src/pages	Main page contents. Files like index.php, login.php,etc.
src/components	Reuseable components, like headers, models.
src/controllers	Essential functions required by the project

Features:

- Provides permission-based CRUD operations.
- Session-based authentication.
- Secure authentication. Avoids session hijacking, fixation and SQL Injections.
- Personalized Dashboard (dashboard.php).
- Error handling with session-based messaging.
- Mobile-responsive UI using Tailwind CSS.
- Light and dark theme modes.

Security Measure:

- Input sanitization
- Password encryption.
- Authenticate to ensure only the owner of account can edit/delete data.
- Form Validation before processing
- Redirect on unauthorized dashboard access.

How to run the project?

To run the project, simply follow the given steps:

1. Install XAMPP and node if not yet.
2. Open command prompt and type: ``` cd C:\xampp\htdocs ``` for windows, ```cd /Applications/XAMPP/htdocs``` for MAC and ```cd opt/lampp/htdocs``` for linux, then hit enter.
3. Type: ``` git clone https://github.com/prashantStha308/user_reg ``` and press enter.
4. ```cd user_reg``` to enter the root directory of the project.
5. Then, type ```npm i``` and press enter to install all required packages.
6. Open XAMPP and start MySQL server.
7. Head over to ```localhost/phpmyadmin``` in a browser of your choice.
8. Follow the database setup above and create user_management database and users table.
9. Create a user with basic CRUD permissions. Set the host name to 'localhost'.
10. Back to the root folder of our project, edit the constants of server.php.
    Set:
    define("USER" , YOUR_USER');
    define("PASSWORD" , ‘YOUR_PASSWORD');
    leave rest as they are.
11. Finally, visit ```localhost/user_reg/src/pages/index.php``` to access the webpage.

User Registration:

1. How will you implement the user registration process in PHP using the provided database and table structure?
   - Using the provided database and table, I'd create a pdo object to create a connection with the database.

By using authentication and form validation, I can use prepared SQL queries to implement user registration process.

2. What steps will you take to validate and sanitize user input during registration?

   - Following steps were taken to validate and sanitize user inputs:

   a. Sanitize Input: Use trim() to remove extra spaces and htmlspecialchars() to convert special characters to HTML entities to prevent XSS attacks.
   b. Validate User:
      i. Check if the username, email or password are not empty.
      ii. Ensure the usernames and emails are unique in the database.
      iii. Validate the email format using filter_var().
      iv. Use regex to make sure that passwords are o valid length and complexity
   c. Use prepared SQL entries and bindings to prevent SQL injection.
   d. Use password_hash() to securely encrypt password.

3. How will you handle cases where a username or email is already in use?

   - In the case when user’s username or email is already in use, an error message is shown to the user guiding them to use a separate username or email.

4. Once a user is successfully registered, how will you store their information in the "users" table?

   - On a successful registration of user, SQL's INSERT statement is prepared, and the required data are bounded using bindParam() method to insert the data in users table.

5. How will you ensure the security of user passwords during registration?

   - To ensure the security of the user, a PHP in-built function, password_hash() is used. This function hashes the provided password with the provided algorithm. For example

```
$password = “Password1”;
$hashed_password = password_hash( $password, PASSWORD_DEFAULT);
echo $hashed_password;
```

   Output: “$2y$10$ksirlDDr.PSvqwu7cEl7Wu4sPlLCmUqrsnEQAAE./kr.i.xwSGo2e”

<u>Session Management:</u>

1. After a successful registration, how will you initiate a session for the newly registered user?

   - On a successful user registration, the user is redirected to login page, upon successful login, the session id is regenerated by session_regenerate_id(true) to prevent any fixation or hijacking attacks. After a session is restarted, essential data like user_id and username are stored within $_SESSION super global variable.

2. What user data will you store in the session variables, and how will you use this data throughout the user's session?

   - In the $_SESSION super global variable, username and user_id are stored to authenticate user and validate forms. We also store last_activity which will help us in implementation of session expiration.

3. How will you handle session expiration and implement a secure logout functionality?

   - To handle a session expiration, we can calculate the time user has been inactive by subtracting $_SESSION[‘last_activity] and time (). When the inactivity time is greater than SESSION_TIMEOUT constant which is set at config.php file, the user is logged out by first unsetting the Session variable and then destroying the session with, session_unset() and session_destroy(). Unsetting the session is extremely important as it removes all the session data.

4. What steps will you take to protect against session hijacking or fixation?

   - Session Hijacking and Fixation are common methods attackers use to get access to victim's control. Session hijacking is when the attacker obtains the session id of the user and uses it to take over victim's control. And session fixation is an attack where a known session ID is forced upon user. Both attacks can be prevented by setting the session_regenerate_id to true at important instances, like in logins or updates.

CRUD Operations:

1. How will you implement the functionality for users to view, edit, and delete their profiles?

   - Creating functionality for users to perform certain functions will require user identification through user authentication and validation. Below is a breakdown of how each functionality is implemented.

   a. Viewing User

      i. A username is set as a url param, so that $_GET super variable can be used to capture it

      ii. Upon capture, data of user associated to that username is captured, and shown for displayed.

      iii. If user of the captured username doesn't exit, a model is displayed to the user informing on the invalid username.

      iv. Viewing a user's dashboard requires authentication to display additional options if user is the owner of that dashboard.

   b. Edit User

      i. Before the update function is executed, an authentication and a validation check must be completed.

      ii. The authentication check makes sure that the person sending the edits is the owner of the account and the validation check ensures that all the required input fields are filled.

      iii. If any of the check is not fulfilled, an error is dispatched. If not, the process continues.

      iv. The update function checks for if the updated username or email exists in the database, if even one of them exists, an error is dispatched. Else, the update process takes place and completes.

   c. Delete User

      i. After the user clicks the delete button, a confirm message appears. The start of deletion process depends on user's input. If yes, it moves forward, else it ends.

      ii. The user is then authenticated, if and only if the authentication is valid the deletion process is forwarded.

      iii. The delete query statement is executed, and the process ends.

2. What security measures will you put in place to ensure that only authorized users can perform CRUD operations?

   - To ensure data security and integrity, this project uses role base access control mechanism. As mentioned in above answer, each key process requires proper authentication and validation to succeed. Along with that, by comparing the user in session variable and that received from the GET super global variable, we set an isGuest Boolean variable which we use to either show or hide important options in the dashboard.

3. Can you provide an example of how you would retrieve and display a list of user profiles from the "users" table?

   - To display a list of user profiles from users table:

```
<?php
```

```
        // Database connection
                $db = new PDO("mysql:host=localhost;dbname=user_management", "USER",
"USERPASSWORD");
        // SQL query to fetch user profiles
        $query = $db->prepare("SELECT * FROM users");
        $query->execute();
        // Fetch all results
        $results = $query->fetchAll(PDO::FETCH_ASSOC);
        // Display user profiles
        echo "User Data: <br>";
    foreach ($results as $user) {
                echo "Username: " . $user['username'] . "<br>";
                echo "Email: " . $user['email'] . "<br>";
                echo "Account created at: " . $user['created_at'] . "<br><br>";
    }
?>
```

4. How will you handle user profile updates and deletions?

   - Handling profile updates and deletions are done through following manner:

   a. User profile Updates

      i. User authentication then input sanitization and form validation. These three occurs in the exact order.

      ii. User_update() function is triggered, the function check if the changed email or username are present in the database.

      iii. If, any one of them or both are present, an exception is thrown.

      iv. Else, we prepare the "UPDATE users SET username = :username, email = :email, description = :description WHERE user_id = :user_id" statement and then bind parameters to the named values, using bindValue() method:

      ```
      $query->bindValue(":user_id", $_SESSION['user_id']);
      $query->bindValue(":username", $username);
      $query->bindValue(":email", $email);
      $query->bindValue(":description", $description);
      ```

      v. The statement is then executed and user is updated.

   b. User profile Deletion

      i. User authentication is done, if user is not the owner, the process is terminated.

      ii. If user is the owner, a statement is prepared in the delete_user() function:
      "DELETE FROM users WHERE user_id = :user_id"
      Then the user_id is bonded using the session’s user_id.

      ```
      $query->bindValue(‘:user_id’ , $_SESSION[‘user_id’]);
      ```

      iii. The statement is then executed, and user is deleted.

      iv. After user deletion, the user is redirected to index.php(home page) using:

      ```
      header(“Location: index.php”);
      ```