National University of Computer & Emerging Sciences,
Karachi Computer Science Department

| Course Code: SE-3003 | Course : Web Engineering Lab |
|---|---|

**Spring 2024, Lab Manual – 09**

**LLO 02: Web application using PHP and MySQL**

## Contents:

- Intro to Web Engineering
- Technologies
- Tools
- Introduction to MySQL
- PHP and MySQL Connection
- PHP CRUD
- PHP Signup & Login
- PHP Session and Cookies

## Introduction to Web Engineering

Web Engineering is the application of systematic and quantifiable approaches (concepts methods, techniques tools) to cost - effective requirements analysis, design, implementation, testing, operation, and maintenance of **high quality Web applications.**

## Technologies to be studied

- HTML
- CSS
- JavaScript
- Bootstrap
- JQuery
- PHP
- MySQL [Database]
- Laravel [PHP FRAMEWORK]

## Tools – IDEs

- Visual Studio Code
- Adobe Dreamweaver
- Visual Studio

- XAMPP

## 9.1 PHP Signup & Login:

User authentication is very common in modern web application. It is a security mechanism that is used to restrict unauthorized access to member-only areas and tools on a site.

Following are the steps to create login and signup pages, using PHP and MySql.

### Building the Registration System

In this section we'll build a registration system that allows users to create a new account by filling out a web form. But, first we need to create a table that will hold all the user data.

### Step 1: Creating the Database Table

Execute the following SQL query to create the *users* table inside your MySQL database.

```
CREATE TABLE users (
    id INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
    username VARCHAR(50) NOT NULL UNIQUE,
    password VARCHAR(255) NOT NULL,
    created_at DATETIME DEFAULT CURRENT_TIMESTAMP
);
```

### Step 2: Creating the Config File

After creating the table, we need create a PHP script in order to connect to the MySQL database server. Let's create a file named "config.php" and put the following code inside it.

```
<?php
/* Database credentials. Assuming you are running MySQL
server with default setting (user 'root' with no password) */
define('DB_SERVER', 'localhost');
define('DB_USERNAME', 'root');
define('DB_PASSWORD', '');
define('DB_NAME', 'demo');

/* Attempt to connect to MySQL database */
$link = mysqli_connect(DB_SERVER, DB_USERNAME, DB_PASSWORD, DB_NAME);

// Check connection
if($link === false){
    die("ERROR: Could not connect. " . mysqli_connect_error());
}
?>
```

**Step 3: Creating the Registration Form For Signup:**

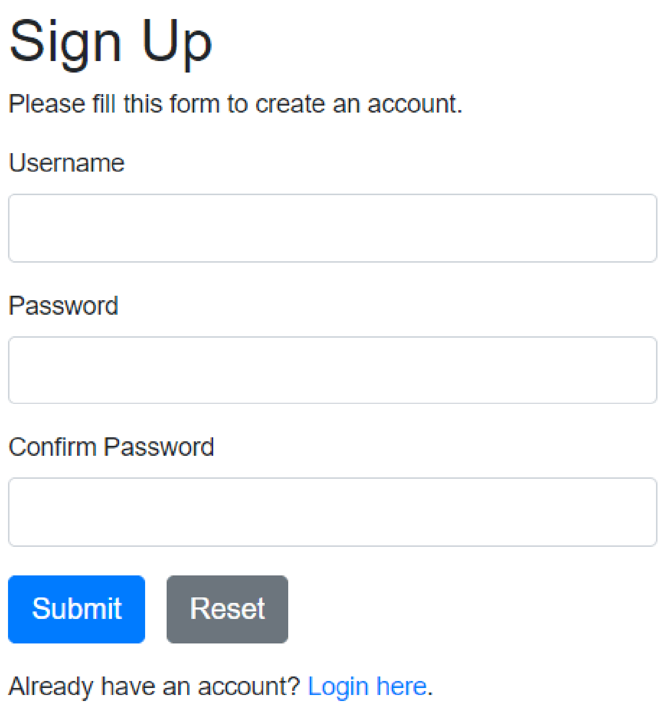
Let's create another PHP file "register.php" and put the following example code in it. This example code will create a web form that allows user to register themselves.

This script will also generate errors if a user tries to submit the form without entering any value, or if username entered by the user is already taken by another user.

```
<?php
// Include config file
require_once "config.php";

// Define variables and initialize with empty values
$username = $password = $confirm_password = "";
$username_err = $password_err = $confirm_password_err = "";

// Processing form data when form is submitted
if($_SERVER["REQUEST_METHOD"] == "POST"){

    // Validate username
    if(empty(trim($_POST["username"]))){
        $username_err = "Please enter a username.";
    } elseif(!preg_match('/^[a-zA-Z0-9_]+$/',
trim($_POST["username"]))){
        $username_err = "Username can only contain letters, numbers,
and underscores.";
    } else{
        // Prepare a select statement
        $sql = "SELECT id FROM users WHERE username = ?";

        if($stmt = mysqli_prepare($link, $sql)){
            // Bind variables to the prepared statement as parameters
            mysqli_stmt_bind_param($stmt, "s", $param_username);

            // Set parameters
            $param_username = trim($_POST["username"]);

            // Attempt to execute the prepared statement
            if(mysqli_stmt_execute($stmt)){
                /* store result */
                mysqli_stmt_store_result($stmt);

                if(mysqli_stmt_num_rows($stmt) == 1){
                    $username_err = "This username is already taken.";
                } else{
                    $username = trim($_POST["username"]);
                }
            } else{
                echo "Oops! Something went wrong. Please try again 
later.";
            }

            // Close statement
            mysqli_stmt_close($stmt);
        }
    }

    // Validate password
    if(empty(trim($_POST["password"]))){
        $password_err = "Please enter a password.";
    } elseif(strlen(trim($_POST["password"])) < 6){
        $password_err = "Password must have atleast 6 characters.";
    } else{
        $password = trim($_POST["password"]);
    }

    // Validate confirm password
    if(empty(trim($_POST["confirm_password"]))){
        $confirm_password_err = "Please confirm password.";
    } else{
        $confirm_password = trim($_POST["confirm_password"]);
        if(empty($password_err) && ($password != $confirm_password)){
            $confirm_password_err = "Password did not match.";
        }
    }

    // Check input errors before inserting in database
    if(empty($username_err) && empty($password_err) &&
empty($confirm_password_err)){

        // Prepare an insert statement
        $sql = "INSERT INTO users (username, password) VALUES (?, ?)";

        if($stmt = mysqli_prepare($link, $sql)){
            // Bind variables to the prepared statement as parameters
            mysqli_stmt_bind_param($stmt, "ss", $param_username,
$param_password);

            // Set parameters
            $param_username = $username;
            $param_password = password_hash($password,
PASSWORD_DEFAULT); // Creates a password hash

            // Attempt to execute the prepared statement
            if(mysqli_stmt_execute($stmt)){
                // Redirect to login page
                header("location: login.php");
            } else{
                echo "Oops! Something went wrong. Please try again
later.";
            }

            // Close statement
            mysqli_stmt_close($stmt);
        }
    }

    // Close connection
    mysqli_close($link);
}
?>

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Sign Up</title>
    <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap
.min.css">
    <style>
        body{ font: 14px sans-serif; }
        .wrapper{ width: 360px; padding: 20px; }
    </style>
</head>
<body>
    <div class="wrapper">
        <h2>Sign Up</h2>
        <p>Please fill this form to create an account.</p>
        <form action="<?php echo
htmlspecialchars($_SERVER["PHP_SELF"]); ?>" method="post">
            <div class="form-group">
                <label>Username</label>
                <input type="text" name="username" class="form-control
<?php echo (!empty($username_err)) ? 'is-invalid' : ''; ?>"
value="<?php echo $username; ?>">
                <span class="invalid-feedback"><?php echo
$username_err; ?></span>
            </div>
            <div class="form-group">
                <label>Password</label>
                <input type="password" name="password" class="form-
control <?php echo (!empty($password_err)) ? 'is-invalid' : ''; ?>"
value="<?php echo $password; ?>">
                <span class="invalid-feedback"><?php echo
$password_err; ?></span>
            </div>
            <div class="form-group">
                <label>Confirm Password</label>
                <input type="password" name="confirm_password"
class="form-control <?php echo (!empty($confirm_password_err)) ? 'is-
invalid' : ''; ?>" value="<?php echo $confirm_password; ?>">
                <span class="invalid-feedback"><?php echo
$confirm_password_err; ?></span>
            </div>
            <div class="form-group">
                <input type="submit" class="btn btn-primary"
value="Submit">
                <input type="reset" class="btn btn-secondary ml-2"
value="Reset">
            </div>
            <p>Already have an account? <a href="login.php">Login
here</a>.</p>
```

```
        </form>
    </div>
</body>
</html>
```

— The output of the above example (i.e. signup form) will look something like this:


Sign Up
Please fill this form to create an account.
Username
Password
Confirm Password
Submit
Reset
Already have an account? Login here.

In the above example, we have used the PHP's inbuilt password_hash() function to create a password hash from the password string entered by the user (*line no-78*). This function creates a password hash using a strong one-way hashing algorithm. It also generates and applies a random salt automatically when hashing the password; this basically means that even if two users have the same passwords, their password hashes will be different.

At the time of login we'll verify the given password with the password hash stored in the database using the PHP password_verify() function, as demonstrated in the next example.

NOTE: Password salting is a technique which is widely used to secure passwords by randomizing password hashes, so that if two users have the same password, they will not have the same password hashes. This is done by appending or prepending a random string, called a salt, to the password before hashing.

**Step 4: Creating the Login page:**

In this section we'll create a login form where user can enter their username and password. When user submit the form these inputs will be verified against the credentials stored in the database, if the username and password match, the user is authorized and granted access to the site, otherwise the login attempt will be rejected.

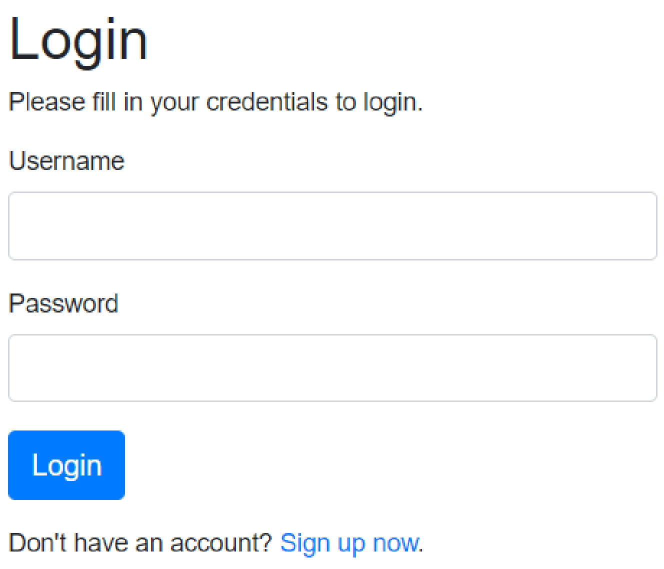
Let's create a file named "login.php" and place the following code inside it.

```
<?php
// Initialize the session
session_start();

// Check if the user is already logged in, if yes then redirect him to
welcome page
if(isset($_SESSION["loggedin"]) && $_SESSION["loggedin"] === true){
    header("location: welcome.php");
    exit;
}

// Include config file
require_once "config.php";

// Define variables and initialize with empty values
$username = $password = "";
$username_err = $password_err = $login_err = "";

// Processing form data when form is submitted
if($_SERVER["REQUEST_METHOD"] == "POST"){

    // Check if username is empty
    if(empty(trim($_POST["username"]))){
        $username_err = "Please enter username.";
    } else{
        $username = trim($_POST["username"]);
    }

    // Check if password is empty
    if(empty(trim($_POST["password"]))){
        $password_err = "Please enter your password.";
    } else{
        $password = trim($_POST["password"]);
    }

    // Validate credentials
    if(empty($username_err) && empty($password_err)){
        // Prepare a select statement
        $sql = "SELECT id, username, password FROM users WHERE
username = ?";
```

```
        if($stmt = mysqli_prepare($link, $sql)){
            // Bind variables to the prepared statement as parameters
            mysqli_stmt_bind_param($stmt, "s", $param_username);

            // Set parameters
            $param_username = $username;

            // Attempt to execute the prepared statement
            if(mysqli_stmt_execute($stmt)){
                // Store result
                mysqli_stmt_store_result($stmt);

                // Check if username exists, if yes then verify
password
                if(mysqli_stmt_num_rows($stmt) == 1){
                    // Bind result variables
                    mysqli_stmt_bind_result($stmt, $id, $username,
$hashed_password);
                    if(mysqli_stmt_fetch($stmt)){
                        if(password_verify($password,
$hashed_password)){
                            // Password is correct, so start a new
session
                            session_start();

                            // Store data in session variables
                            $_SESSION["loggedin"] = true;
                            $_SESSION["id"] = $id;
                            $_SESSION["username"] = $username;

                            // Redirect user to welcome page
                            header("location: welcome.php");
                        } else{
                            // Password is not valid, display a
generic error message
                            $login_err = "Invalid username or
password.";
                        }
                    }
                } else{
                    // Username doesn't exist, display a generic error
message
                    $login_err = "Invalid username or password.";
                }
            } else{
                echo "Oops! Something went wrong. Please try again
later.";
            }

            // Close statement
            mysqli_stmt_close($stmt);
        }
    }

    // Close connection
    mysqli_close($link);
}
?>

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Login</title>
    <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap
.min.css">
    <style>
        body{ font: 14px sans-serif; }
        .wrapper{ width: 360px; padding: 20px; }
    </style>
</head>
<body>
    <div class="wrapper">
        <h2>Login</h2>
        <p>Please fill in your credentials to login.</p>

        <?php
        if(!empty($login_err)){
            echo '<div class="alert alert-danger">' . $login_err .
'</div>';
        }
        ?>

        <form action="<?php echo
htmlspecialchars($_SERVER["PHP_SELF"]); ?>" method="post">
            <div class="form-group">
                <label>Username</label>
                <input type="text" name="username" class="form-control
<?php echo (!empty($username_err)) ? 'is-invalid' : ''; ?>"
value="<?php echo $username; ?>">
```

```
                <span class="invalid-feedback"><?php echo
$username_err; ?></span>
            </div>
            <div class="form-group">
                <label>Password</label>
                <input type="password" name="password" class="form-
control <?php echo (!empty($password_err)) ? 'is-invalid' : ''; ?>">
                <span class="invalid-feedback"><?php echo
$password_err; ?></span>
            </div>
            <div class="form-group">
                <input type="submit" class="btn btn-primary"
value="Login">
            </div>
            <p>Don't have an account? <a href="register.php">Sign up
now</a>.</p>
        </form>
    </div>
</body>
</html>
```

— The output of the above example (i.e. login form) will look something like this:


Login
Please fill in your credentials to login.
Username
Password
Login
Don't have an account? Sign up now.

## 9.2 PHP Cookies & Session:

**What is a Cookie**

A cookie is a small text file that lets you store a small amount of data (nearly 4KB) on the user's computer. They are typically used to keeping track of information such as username that the site can retrieve to personalize the page when user visit the website next time.

Note: Each time the browser requests a page to the server, all the data in the cookie is automatically sent to the server within the request.

## **Setting a Cookie in PHP**

The setcookie() function is used to set a cookie in PHP. Make sure you call the setcookie() function before any output generated by your script otherwise cookie will not set. The basic syntax of this function can be given with:

```
setcookie(name, value, expire, path, domain, secure);
```

The parameters of the setcookie() function have the following meanings:

| Parameter | Description |
|---|---|
| `name` | The name of the cookie. |
| `value` | The value of the cookie. Do not store sensitive information since this value is stored on the user's computer. |
| `expires` | The expiry date in UNIX timestamp format. After this time cookie will become inaccessible. The default value is 0. |
| `path` | Specify the path on the server for which the cookie will be available. If set to `/`, the cookie will be available within the entire domain. |
| `domain` | Specify the domain for which the cookie is available to e.g www.example.com. |
| secure | This field, if present, indicates that the cookie should be sent only if a secure HTTPS connection exists. |

NOTE: If the expiration time of the cookie is set to 0, or omitted, the cookie will expire at the end of the session i.e. when the browser closes.

Here's an example that uses setcookie() function to create a cookie named username and assign the value value John Carter to it. It also specify that the cookie will expire after 30 days (30 days * 24 hours * 60 min * 60 sec).

```
<?php
// Setting a cookie
setcookie("username", "John Carter", time()+30*24*60*60);
?>
```

### Accessing Cookies Values

The PHP $_COOKIE superglobal variable is used to retrieve a cookie value. It typically an associative array that contains a list of all the cookies values sent by the browser in the current request, keyed by cookie name. The individual cookie value can be accessed using standard array notation, for example to display the username cookie set in the previous example, you could use the following code.

```
<?php
// Accessing an individual cookie value
echo $_COOKIE["username"];
?>
```

The PHP code in the above example produce the following output.

John Carter

It's a good practice to check whether a cookie is set or not before accessing its value. To do this you can use the PHP isset() function, like this:

```
<?php
// Verifying whether a cookie is set or not
if(isset($_COOKIE["username"])){
    echo "Hi " . $_COOKIE["username"];
} else{
    echo "Welcome Guest!";
}
?>
```

### Removing Cookies

You can delete a cookie by calling the same setcookie() function with the cookie name and any value (such as an empty string) however this time you need the set the expiration date in the past, as shown in the example below:

```
<?php
// Deleting a cookie
setcookie("username", "", time()-3600);
?>
```

## <u>What is a Session</u>

Although you can store data using cookies but it has some security issues. Since cookies are stored on user's computer it is possible for an attacker to easily modify a cookie content to insert potentially harmful data in your application that might break your application.

Also every time the browser requests a URL to the server, all the cookie data for a website is automatically sent to the server within the request. It means if you have stored 5 cookies on user's system, each having 4KB in size, the browser needs to upload 20KB of data each time the user views a page, which can affect your site's performance.

You can solve both of these issues by using the PHP session. A PHP session stores data on the server rather than user's computer. In a session based environment, every user is identified through a unique number called session identifier or SID. This unique session ID is used to link each user with their own information on the server like emails, posts, etc.

NOTE: The session IDs are randomly generated by the PHP engine which is almost impossible to guess. Furthermore, because the session data is stored on the server, it doesn't have to be sent with every browser request.

### Starting a PHP Session

Before you can store any information in session variables, you must first start up the session. To begin a new session, simply call the PHP session_start() function. It will create a new session and generate a unique session ID for the user.

The PHP code in the example below simply starts a new session.

```
<?php
// Starting session
session_start();
?>
```

The session_start() function first checks to see if a session already exists by looking for the presence of a session ID. If it finds one, i.e. if the session is already started, it sets up the session variables and if doesn't, it starts a new session by creating a new session ID.

Note: You must call the session_start() function at the beginning of the page i.e. before any output generated by your script in the browser, much like you do while setting the cookies with setcookie() function.

### Storing and Accessing Session Data

You can store all your session data as key-value pairs in the $_SESSION[] superglobal array. The stored data can be accessed during lifetime of a session. Consider the following script, which creates a new session and registers two session variables.

```
<?php
// Starting session
session_start();

// Storing session data
$_SESSION["firstname"] = "Peter";
$_SESSION["lastname"] = "Parker";
?>
```

To access the session data we set on our previous example from any other page on the same web domain — simply recreate the session by calling session_start() and then pass the corresponding key to the $_SESSION associative array.

```
<?php
// Starting session
session_start();

// Accessing session data
echo 'Hi, ' . $_SESSION["firstname"] . ' ' . $_SESSION["lastname"];
?>
```

The PHP code in the example above produce the following output.

```
Hi, Peter Parker
```

**Destroying a Session**

If you want to remove certain session data, simply unset the corresponding key of the $_SESSION associative array, as shown in the following example:

```
<?php
// Starting session
session_start();

// Removing session data
if(isset($_SESSION["lastname"])){
    unset($_SESSION["lastname"]);
}
?>
```

However, to destroy a session completely, simply call the session_destroy() function. This function does not need any argument and a single call destroys all the session data.

```
<?php
// Starting session
session_start();

// Destroying session
session_destroy();
?>
```

Every PHP session has a timeout value — a duration, measured in seconds — which determines how long a session should remain alive in the absence of any user activity. You can adjust this timeout duration by changing the value of session.gc_maxlifetime variable in the PHP configuration file (php.ini).

# *Tasks*

1. Create a simple Crud for Admin Panel of Hospital Management System.

2. Add login functionality to the CRUD system using session management. Create a login form that allows users to enter their username and password. If the credentials match with the database, the user should be redirected to the dashboard. Otherwise, an error message should be displayed.

3. Add cookies functionality to the CRUD system. Use cookies to remember the user's preferences or last visited page.

4. Implement a search functionality to the CRUD system using PHP and MySQL. Create a search form that allows users to search for data in the database. The search results should be displayed in a table format and should include pagination