

Republic of the Philippines

**CAVITE STATE UNIVERSITY**

**Don Severino delas Alas Campus**

Indang, Cavite

**Names: John Robin Estrella, Julia Carla C. Abutin Course, Yr. & Section: BSIT 3-1 Date: June 30, 2024**

**ITEC106 – IT Elective 2 – Web Systems and Technologies 2**

Laboratory Manual 8 – CAPTCHA

**Prepared by:** Simeon E. Daez & Jose Andrew A. Bunan

Create a PHP-based form that includes CAPTCHA to prevent automated submissions. The form should verify the CAPTCHA response and provide feedback to the user. Use your website for the final project as the basis for this activity.

**This activity is to be done by pair/trio according to the final project.**

Provide the modified PHP script and the screenshot of your output in the space provided below.

Save your work as a **PDF** file with the filename ***surname\_lab8*** e.g. bunan\_lab8

Upload your work on the respective links:

**BSIT 3-1:**

[**https://drive.google.com/drive/folders/1bokEETfnrwDwnf3SsxU9zxOyDT17KfQ1?usp=sharing**](https://drive.google.com/drive/folders/1bokEETfnrwDwnf3SsxU9zxOyDT17KfQ1?usp=sharing)

**BSIT 3-2:**

[**https://drive.google.com/drive/folders/1bokEETfnrwDwnf3SsxU9zxOyDT17KfQ1?usp=sharing**](https://drive.google.com/drive/folders/1bokEETfnrwDwnf3SsxU9zxOyDT17KfQ1?usp=sharing)

**BSIT 3-4:**

[**https://drive.google.com/drive/folders/18yS5fz4BWI3ANih2rgwb4aiNxT5wra3t?usp=drive\_link**](https://drive.google.com/drive/folders/18yS5fz4BWI3ANih2rgwb4aiNxT5wra3t?usp=drive_link)

**Grading Rubrics:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Laboratory Activity Grading Rubric** | | | |
| **Criteria** | **Definition** | **Points Possible** | **Earned Points** |
| **Discussion** | The student could follow the directions given in the activity and answer the questions properly. | 30 |  |
| **Coding Skills** | The student could create what was being asked in the activity. | 30 |  |
| **Scientific Understanding** | The student demonstrated a clear understanding of PHP Cookies and User Sessions. | 30 |  |
| **Runtime** | The website did not have any bugs during runtime. | 10 |  |
| **Total** | | 100 |  |

**Checked by:**

**JOSE ANDREW A. BUNAN**

**Instructor I**

**MODIFIED SCRIPT:**

**db\_connection.php**<?php

// Database connection details

$host = 'localhost';

$db = 'mywebsite';

$user = 'root';

$pass = '';

$charset = 'utf8mb4';

// Connection string (DSN)

$dsn = "mysql:host=$host;dbname=$db;charset=$charset";

// Options for the PDO connection

$options = [

PDO::ATTR\_ERRMODE => PDO::ERRMODE\_EXCEPTION, // Show errors as exceptions

PDO::ATTR\_DEFAULT\_FETCH\_MODE => PDO::FETCH\_ASSOC, // Fetch results as an associative array

PDO::ATTR\_EMULATE\_PREPARES => false, // Use real prepared statements

];

try {

// Create a new PDO instance

$pdo = new PDO($dsn, $user, $pass, $options);

} catch (\PDOException $e) {

// If there's an error, show the error message and code

throw new \PDOException($e->getMessage(), (int)$e->getCode());

}

**auth.php**

<?php

session\_start(); // Start the session

// Include the database connection

include 'includes/db\_connection.php';

// Function to log captcha attempts

function logCaptchaAttempt($pdo, $captcha) {

$ip\_address = $\_SERVER['REMOTE\_ADDR'];

$sql = "INSERT INTO captcha\_log (ip\_address, captcha) VALUES (:ip\_address, :captcha)";

$stmt = $pdo->prepare($sql);

$stmt->bindValue(':ip\_address', $ip\_address);

$stmt->bindValue(':captcha', $captcha ? $captcha : null, PDO::PARAM\_STR);

if (!$stmt->execute()) {

error\_log(print\_r($stmt->errorInfo(), true));

}

}

if ($\_SERVER['REQUEST\_METHOD'] == 'POST') { // Check if the request is a POST request

if (isset($\_POST['login'])) { // Check if the login form is submitted

$email = $\_POST['email'];

$password = $\_POST['password'];

$captcha = isset($\_POST['captcha']) ? $\_POST['captcha'] : null;

$remember\_me = isset($\_POST['remember\_me']);

// Validate email

if (empty($email)) {

$\_SESSION['errors']['email'] = 'Email is required.';

header("Location: login.php");

exit();

}

// Validate password

if (empty($password)) {

$\_SESSION['errors']['password'] = 'Password is required.';

header("Location: login.php");

exit();

}

// Validate captcha

if (!isset($\_SESSION['captcha']) || empty($\_SESSION['captcha']) || $captcha !== $\_SESSION['captcha']) {

$\_SESSION['errors']['captcha'] = 'Incorrect CAPTCHA.';

logCaptchaAttempt($pdo, $captcha);

header("Location: login.php");

exit();

}

// Admin login

$sql = "SELECT id, first\_name, last\_name, email, password FROM admins WHERE email = :email";

$stmt = $pdo->prepare($sql);

$stmt->execute(['email' => $email]);

$admin = $stmt->fetch();

if ($admin && password\_verify($password, $admin['password'])) {

// Set session variables for admin

$\_SESSION['admin\_id'] = $admin['id'];

$\_SESSION['admin\_email'] = $admin['email'];

$\_SESSION['admin\_first\_name'] = $admin['first\_name'];

$\_SESSION['admin\_last\_name'] = $admin['last\_name'];

if ($remember\_me) {

setcookie("admin\_email", $email, time() + (86400 \* 30), "/"); // 30 days

setcookie("admin\_password", $password, time() + (86400 \* 30), "/"); // 30 days

} else {

// Expire cookies if Remember Me is not checked

setcookie("admin\_email", "", time() - 3600, "/");

setcookie("admin\_password", "", time() - 3600, "/");

}

header("Location: indexes.php");

exit();

}

// User login

$sql = "SELECT id, fname, lname, email, password, image FROM users WHERE email = :email";

$stmt = $pdo->prepare($sql);

$stmt->execute(['email' => $email]);

$user = $stmt->fetch();

if ($user && password\_verify($password, $user['password'])) {

// Set session variables for regular user

$\_SESSION['user\_id'] = $user['id'];

$\_SESSION['name'] = $user['fname'] . ' ' . $user['lname'];

$\_SESSION['email'] = $user['email'];

$\_SESSION['image'] = $user['image'];

if ($remember\_me) {

setcookie("email", $email, time() + (86400 \* 30), "/"); // 30 days

setcookie("password", $password, time() + (86400 \* 30), "/"); // 30 days

} else {

// Expire cookies if Remember Me is not checked

setcookie("email", "", time() - 3600, "/");

setcookie("password", "", time() - 3600, "/");

}

header("Location: index.php");

exit();

} else {

// Set login error message if email or password is incorrect

$\_SESSION['errors']['login'] = 'Invalid email or password.';

header("Location: login.php");

exit();

}

}

}

**login.php**

<?php

// Include the authentication script

include("auth.php");

// Include the error message script

include("error-message.php");

?>

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<!-- Link to the favicon -->

<link rel="icon" type="image/png" sizes="16x16" href="http://localhost/Filipiniana%20E-commerce/assets/images/favicon-16x16.png">

<!-- Page title -->

<title>Flpn.e-Co | Sign In</title>

<!-- Link to the custom fonts -->

<link rel="stylesheet" href="assets/css/fonts.css">

<!-- Link to the Font Awesome library -->

<link rel="stylesheet" href="assets/font-awesome-6-pro-main/css/all.min.css">

<!-- Link to the custom styles -->

<link rel="stylesheet" href="assets/css/styles.css">

</head>

<body>

<!-- Container for displaying error messages -->

<div class="message-container">

<?php showError(); // Call the function to display errors ?>

</div>

<!-- Main container -->

<div class="container">

<!-- Login form container -->

<div class="login-form">

<!-- Login form -->

<form class="form" method="post" action="auth.php">

<!-- Form title -->

<p class="title">Login</p>

<!-- Display login error if it exists -->

<?php if (isset($\_SESSION['errors']['login'])): ?>

<div class="error"><?php echo $\_SESSION['errors']['login']; ?></div>

<?php endif; ?>

<!-- Email input field -->

<label>

<input required="" placeholder="" type="email" class="input" name="email"

value="<?php echo isset($\_COOKIE['email']) ? htmlspecialchars($\_COOKIE['email']) : (isset($\_SESSION['post\_data']['email']) ? htmlspecialchars($\_SESSION['post\_data']['email']) : ''); ?>">

<span>Email</span>

</label>

<!-- Display email error if it exists -->

<?php if (isset($\_SESSION['errors']['email'])): ?>

<div class="error"><?php echo $\_SESSION['errors']['email']; ?></div>

<?php endif; ?>

<!-- Password input field -->

<label>

<input required="" placeholder="" type="password" class="input" name="password"

value="<?php echo isset($\_COOKIE['password']) ? htmlspecialchars($\_COOKIE['password']) : ''; ?>">

<span>Password</span>

</label>

<!-- Display password error if it exists -->

<?php if (isset($\_SESSION['errors']['password'])): ?>

<div class="error"><?php echo $\_SESSION['errors']['password']; ?></div>

<?php endif; ?>

<!-- Remember Me checkbox and Forgot Password link -->

<div class="forgot-pass">

<label class="chk">

<input type="checkbox" id="rememberMe" name="remember\_me" <?php echo isset($\_COOKIE['email']) ? 'checked' : ''; ?>>

<div class="checkmark"></div>

<span>Remember Me</span>

</label>

<a href="#">Forgot password?</a>

</div>

<!-- Captcha input field -->

<div class="captcha">

<label for="captcha-input">Enter Captcha</label>

<div class="preview"></div>

<div class="captcha-form">

<input type="text" id="captcha-form" name="captcha" placeholder="Enter captcha text">

<button class="captcha-refresh">

<i class="fa-solid fa-arrows-rotate"></i>

</button>

</div>

<!-- Display captcha error if it exists -->

<?php if (isset($\_SESSION['errors']['captcha'])): ?>

<div class="error"><?php echo $\_SESSION['errors']['captcha']; ?></div>

<?php endif; ?>

</div>

<!-- Submit button -->

<button type="submit" class="submit" id="login-btn" name="login">Login</button>

<!-- Signup link -->

<p class="signin">Don't have an account? <a href="register.php">Signup</a> </p>

</form>

</div>

</div>

<!-- Include the captcha script -->

<script src="assets/js/captcha.js"></script>

<!-- Script to remove error messages after 5 seconds -->

<script>

setTimeout(function() {

var messages = document.querySelectorAll('.message.show');

messages.forEach(function(message) {

message.classList.remove('show');

});

}, 5000);

</script>

</body>

</html>

**captcha.js**

(function() {

// Array of font styles for the captcha characters

const fonts = ["cursive", "sans-serif", "serif", "monospace"];

// Variable to store the generated captcha value

let captchaValue = "";

// Function to generate a new captcha value

function generateCaptcha() {

// Generate a random value and encode it in base64

let value = btoa(Math.random() \* 1000000000);

// Trim the value to a random length between 5 and 10 characters

value = value.substr(0, 5 + Math.random() \* 5);

// Store the generated value in the captchaValue variable

captchaValue = value;

// Send the captcha value to the server via POST request

fetch('store\_captcha.php', {

method: 'POST',

headers: {

'Content-Type': 'application/json'

},

// Convert the captcha value to JSON format

body: JSON.stringify({ captcha: captchaValue })

})

.then(response => {

// Check if the response is not ok

if (!response.ok) {

throw new Error('Network response was not ok');

}

// Parse the response as JSON

return response.json();

})

.then(data => {

// Log the response data

console.log('Response:', data);

// Check if the response indicates a failure

if (!data.success) {

// Log an error message

console.error('Error:', data.message);

// Handle error, e.g., display an error message to the user

} else {

// Call setCaptcha to display the captcha after generating it successfully

setCaptcha();

}

})

.catch(error => {

// Log any errors that occur during the fetch request

console.error('Error:', error);

});

}

// Function to display the generated captcha value

function setCaptcha() {

// Map each character in the captcha value to a styled HTML element

let html = captchaValue.split("").map((char) => {

// Generate a random rotation angle between -20 and 20 degrees

const rotate = -20 + Math.trunc(Math.random() \* 40);

// Select a random font style from the fonts array

const font = fonts[Math.trunc(Math.random() \* fonts.length)];

// Return the character wrapped in a span element with the generated styles

return `<span style="transform: rotate(${rotate}deg); font-family:${font}">${char}</span>`;

}).join("");

// Set the inner HTML of the captcha preview element to the generated HTML

document.querySelector(".captcha .preview").innerHTML = html;

}

// Function to initialize the captcha

function initCaptcha() {

// Add an event listener to the captcha refresh button

document.querySelector(".captcha .captcha-refresh").addEventListener("click", function(event) {

// Prevent the default form submission behavior

event.preventDefault();

// Generate a new captcha when the refresh button is clicked

generateCaptcha();

});

// Generate a new captcha when the page loads

generateCaptcha();

}

// Call initCaptcha to initialize the captcha when the script is loaded

initCaptcha();

})();

**store\_captcha.php**

<?php

session\_start(); // Start the session

// Check if the request method is POST

if ($\_SERVER['REQUEST\_METHOD'] == 'POST') {

// Decode the JSON data received in the POST request

$data = json\_decode(file\_get\_contents('php://input'), true);

// Check if the 'captcha' key is set in the decoded data

if (isset($data['captcha'])) {

// Store the captcha value in the session

$\_SESSION['captcha'] = $data['captcha'];

// You can also store the captcha value in a database if needed

echo json\_encode(['success' => true]); // Return a success response

exit(); // Exit the script

}

}

// Return an error response if the request is invalid

echo json\_encode(['success' => false, 'message' => 'Invalid request']);

exit(); // Exit the script

?>

**Database for captcha**

CREATE TABLE your\_table\_name (

id INT(11) NOT NULL AUTO\_INCREMENT,

ip\_address VARCHAR(50) NOT NULL,

captcha VARCHAR(255),

attempt\_date TIMESTAMP NOT NULL DEFAULT CURRENT\_TIMESTAMP,

PRIMARY KEY (id)

);

// we need to store the captcha just to have a proper verification and validation, if the CAPTCHA is stored in a session, it ensures that the CAPTCHA value remains consistent for the user session. So, if the user submits a form with a CAPTCHA, the entered value must be compared against the stored value to verify its correctness.

**layout.css**

:root {

/\* color styling \*/

--primary-color: #A78874;

--secondary-color: #422622;

--btn-hover: rgba(104, 65, 44);

--accent-color: #ffca99;

--text-color: #ffffff;

--bg-color: #fff;

--semi-bg-color: #f5e3d3;

--table-text-color: #000;

}

**styles.css**

/\* layout css \*/

@import url('http://localhost/Filipiniana%20E-commerce/assets/css/layout.css');

\* {

margin: 0;

padding: 0;

box-sizing: border-box;

font-family: "Poppins", sans-serif;

text-decoration: none;

}

body {

display: flex;

justify-content: center;

align-items: center;

min-height: 100vh;

background-color: #f0f0f0;

}

.container {

width: 100%;

max-width: 400px;

background: #fff;

padding: 40px;

border-radius: 10px;

box-shadow: 0 10px 15px -3px rgba(0, 0, 0, 0.1), 0 4px 6px -2px rgba(0, 0, 0, 0.05);

}

.form {

display: flex;

flex-direction: column;

gap: 10px;

}

.title {

font-size: 28px;

color: var(--secondary-color);

font-weight: 600;

letter-spacing: -1px;

position: relative;

display: flex;

align-items: center;

padding-left: 30px;

}

.title::before,

.title::after {

position: absolute;

content: "";

height: 16px;

width: 16px;

border-radius: 50%;

left: 0px;

background-color: var(--secondary-color);

}

.title::before {

width: 18px;

height: 18px;

background-color: var(--secondary-color);

}

.title::after {

width: 18px;

height: 18px;

animation: pulse 1s linear infinite;

}

.message, .signin {

color: rgba(88, 87, 87, 0.822);

font-size: 14px;

}

.signin {

text-align: center;

}

.signin a {

color: var(--secondary-color);

}

.signin a:hover {

text-decoration: underline var(--secondary-color);

}

.flex {

display: flex;

width: 100%;

gap: 6px;

}

.form label {

position: relative;

}

.form label .input {

width: 100%;

padding: 10px 10px 20px 10px;

outline: 0;

border: 1px solid rgba(105, 105, 105, 0.397);

border-radius: 10px;

}

.form label .input + span {

position: absolute;

left: 10px;

top: 15px;

color: grey;

font-size: 0.9em;

cursor: text;

transition: 0.3s ease;

}

.form label .input:placeholder-shown + span {

top: 15px;

font-size: 0.9em;

}

.form label .input:focus + span,

.form label .input:valid + span {

top: 25px;

font-size: 0.7em;

font-weight: 600;

}

.form label .input + span {

color: grey;

}

.form label .input:focus + span,

.form label .input:not(:placeholder-shown):invalid + span {

color: rgb(245, 93, 93);

}

.form label .input:valid + span {

color: rgb(81, 198, 81);

}

.submit {

border: none;

outline: none;

background-color: var(--secondary-color);

padding: 10px;

border-radius: 10px;

color: #fff;

font-size: 16px;

transform: .3s ease;

transition: all 200ms ease;

}

.submit:hover {

background-color: rgba(104, 65, 44);

}

.container .captcha {

margin: 5px 0px;

}

.container .captcha .preview {

color: #555;

width: 100%;

text-align: center;

height: 40px;

line-height: 40px;

letter-spacing: 8px;

border: 1px dashed #888;

font-family: 'monospace';

}

.container .captcha .preview span {

display: inline-block;

user-select: none;

}

.container .captcha .captcha-form {

display: flex;

}

.captcha label {

font-size: 14px;

margin-left: 2px;

color: var(--secondary-color);

text-transform: uppercase;

}

.container .captcha .captcha-form input {

width: 100%;

padding: 8px;

border: 1px solid #888;

}

.container .captcha .captcha-form .captcha-refresh {

width: 40px;

border: none;

outline: none;

background: #888;

color: #eee;

cursor: pointer;

}

.forgot-pass {

display: flex;

align-items: center;

justify-content: space-between;

margin-top: 10px;

}

.chk input {

position: absolute;

opacity: 0;

cursor: pointer;

height: 0;

width: 0;

}

.chk {

display: flex;

align-items: center;

position: relative;

cursor: pointer;

font-size: 15px;

user-select: none;

}

.chk span {

font-size: 14px;

color: var(--primary-color);

}

.checkmark {

position: relative;

top: 0;

left: 0;

height: 1.3em;

width: 1.3em;

background-color: #ccc;

border-radius: 50%;

transition: 300ms;

margin-right: 5px;

}

.chk input:checked ~ .checkmark {

background-color: var(--secondary-color);

border-radius: .5rem;

animation: pulse2 500ms ease-in-out;

}

.checkmark:after {

content: "";

position: absolute;

display: none;

}

.chk input:checked ~ .checkmark:after {

display: block;

}

.chk .checkmark:after {

left: 0.45em;

top: 0.25em;

width: 0.25em;

height: 0.5em;

border: solid #E0E0E2;

border-width: 0 0.15em 0.15em 0;

transform: rotate(45deg);

}

.forgot-pass a,

.links a {

text-decoration: none;

color: var(--primary-color);

font-size: 14px;

transition: color 0.3s ease;

}

.forgot-pass a:hover,

.links a:hover {

color: var(--secondary-color);

text-decoration: underline;

}

.links p {

font-size: 0.90rem;

}

.links {

display: flex;

justify-content: space-evenly;

margin-top: 10px;

}

.error{

color: rgb(234, 85, 85);

font-size: 12px;

font-weight: bold;

margin-inline: 5px;

}

.success { color: green; }

.message-container {

position: fixed;

top: 20px;

right: 20px;

z-index: 1000;

}

.message {

padding: 10px 20px;

border-radius: 5px;

margin-bottom: 10px;

width: 250px;

transition: opacity 0.3s ease-in-out, transform 0.3s ease-in-out;

opacity: 0;

transform: translateY(-20px);

}

.success {

background-color: #4CAF50;

color: #fff;

}

.error {

background-color: #f44336;

color: #fff;

}

.con-flu {

width: 100%;

display: flex;

justify-content: center;

}

.back {

margin-top: 10px;

background: var(--secondary-color);

width: 30px;

height: 30px;

border-radius: 50%;

display: flex;

justify-content: center;

align-items: center;

}

.back:hover {

color: rgb(100, 73, 41);

}

.back a {

display: inline-block;

text-decoration: none;

color: white;

font-size: 20px;

line-height: 40px;

}

.message.show {

opacity: 1;

transform: translateY(0);

}

@keyframes pulse2 {

0% {

box-shadow: 0 0 0 #6e460b90;

rotate: 20deg;

}

50% {

rotate: -20deg;

}

75% {

box-shadow: 0 0 0 10px #6e540b60;

}

100% {

box-shadow: 0 0 0 13px #6e4f0b30;

rotate: 0;

}

}

@keyframes pulse {

from {

transform: scale(0.9);

opacity: 1;

}

to {

transform: scale(1.8);

opacity: 0;

}

}

**SCREENSHOT OF OUTPUT:**

**A screenshot of a login screen

Description automatically generated**

A screenshot of a login page

Description automatically generated