**MSSV: B2203440**

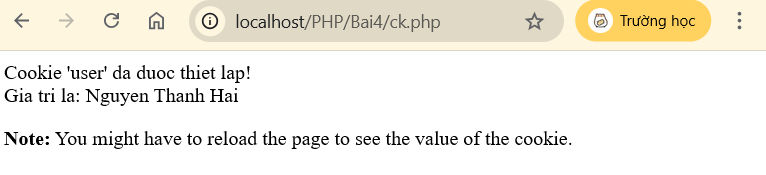
**Họ và tên: Nguyễn Lập Huỳnh Giang**

**Bài tập thực hành 04**

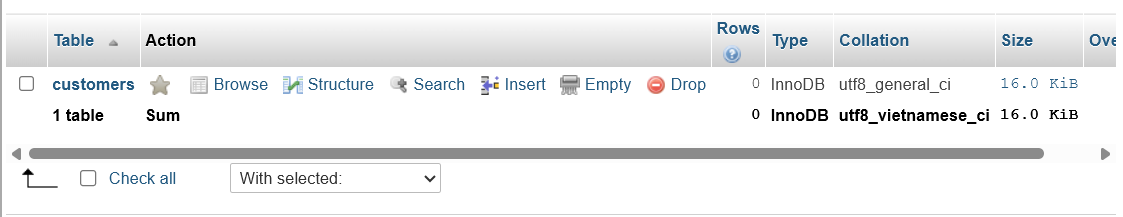
Lập trình PHP nâng cao

# Câu 1: Bạn hãy chạy tất cả các lệnh hướng dẫn ở trên và chụp lại màn hình kết quả.

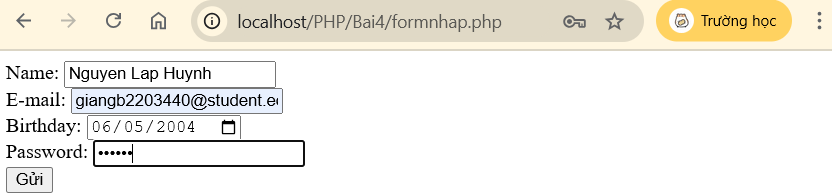
* **Tạo Cookies**



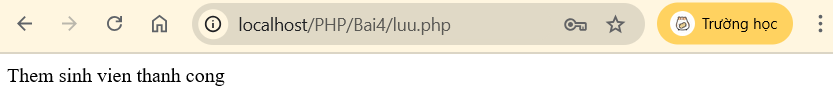
* **Tạo 1 trang đăng ký người dùng với mật khẩu được mã hóa**



Hình 1: Tạo bảng customers

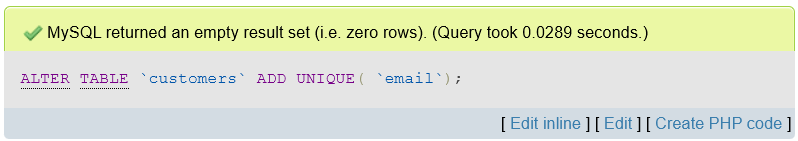


Hình 2: Form nhập



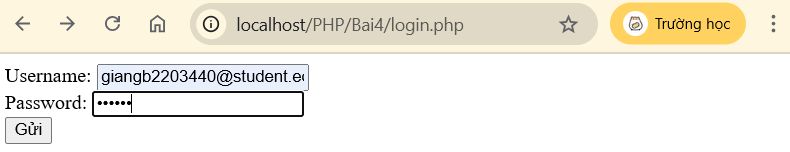


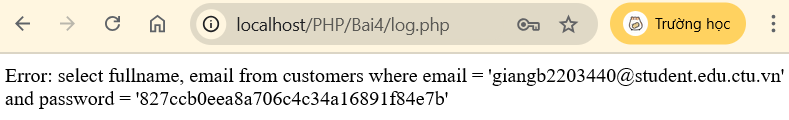
Hình 3: Đã lưu sinh viên vừa nhập vào customers



Hình 4: Chọn email là unique key để không có email trùng nhau khi dùng làm username

* Tạo form đăng nhập



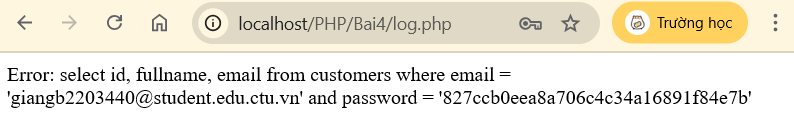


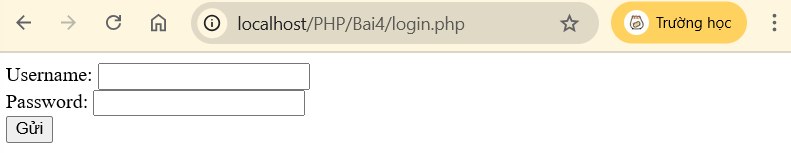
Hình 5: Nếu nhập sai password



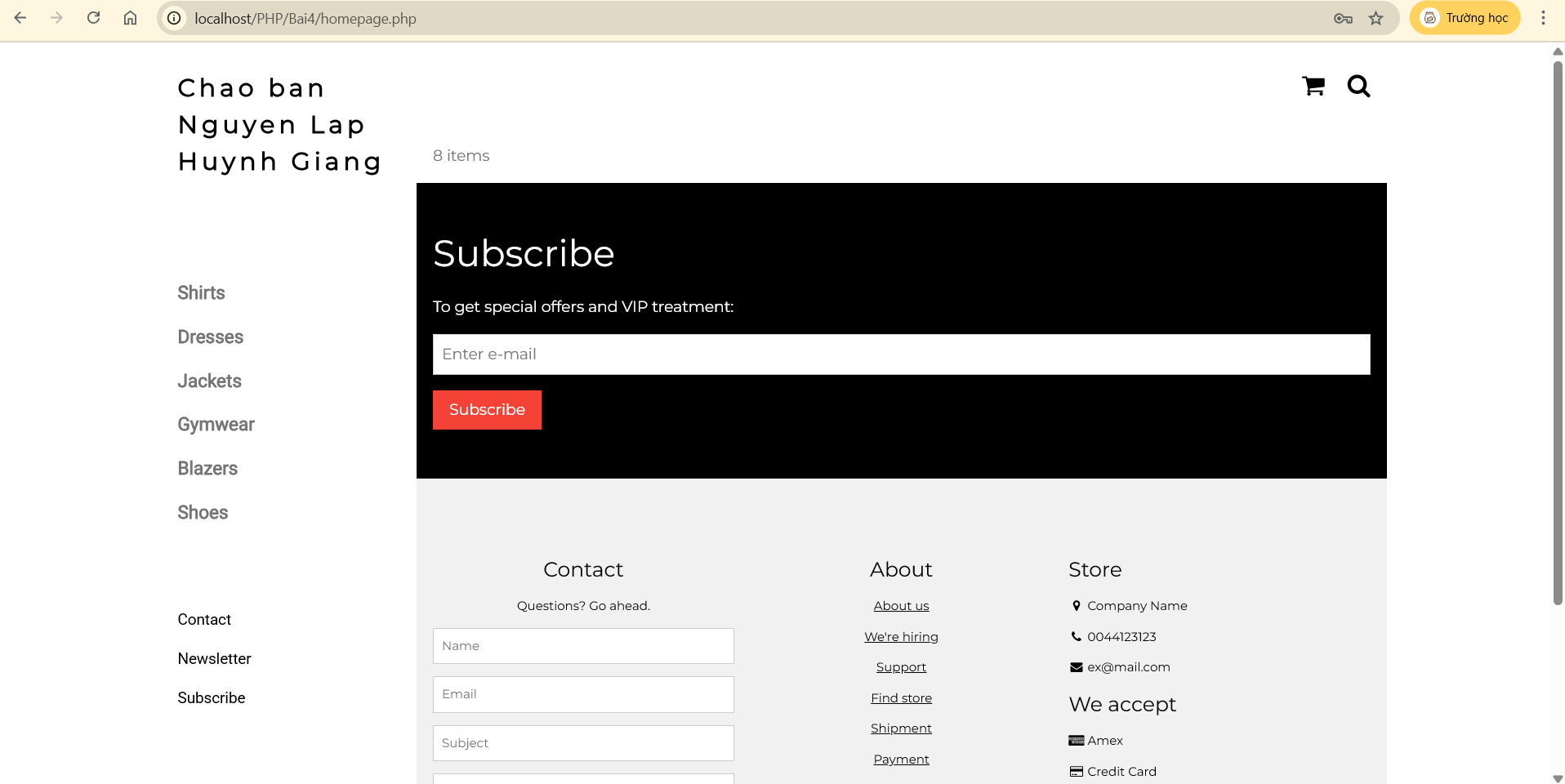
Hình 6: Khi nhập đúng

* Dùng Cookies lưu lại tên đăng nhập, email và điều hướng



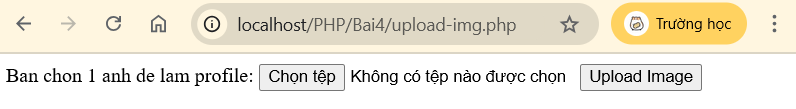


Hình 7: Nếu nhập sai password thì vẫn sẽ hiện thông báo rồi quay về trang đăng nhập

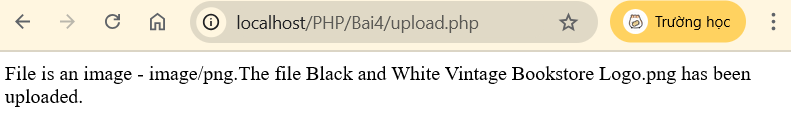


Hình 8: Nếu nhập đúng sẽ chuyển đến trang chủ như thế này

* Upload tập tin

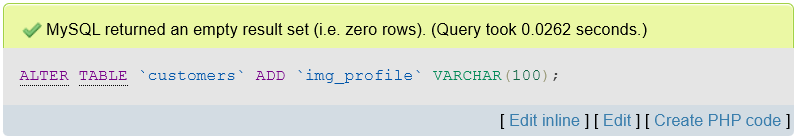


Hình 9: Trang upload ảnh

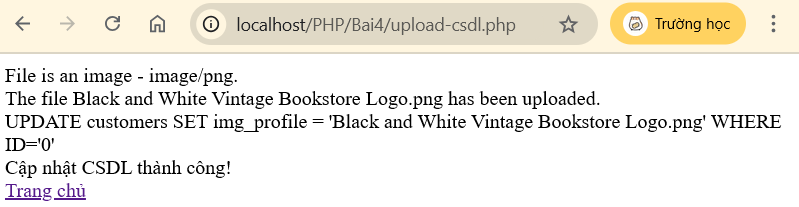


Hình 10: Upload thành công

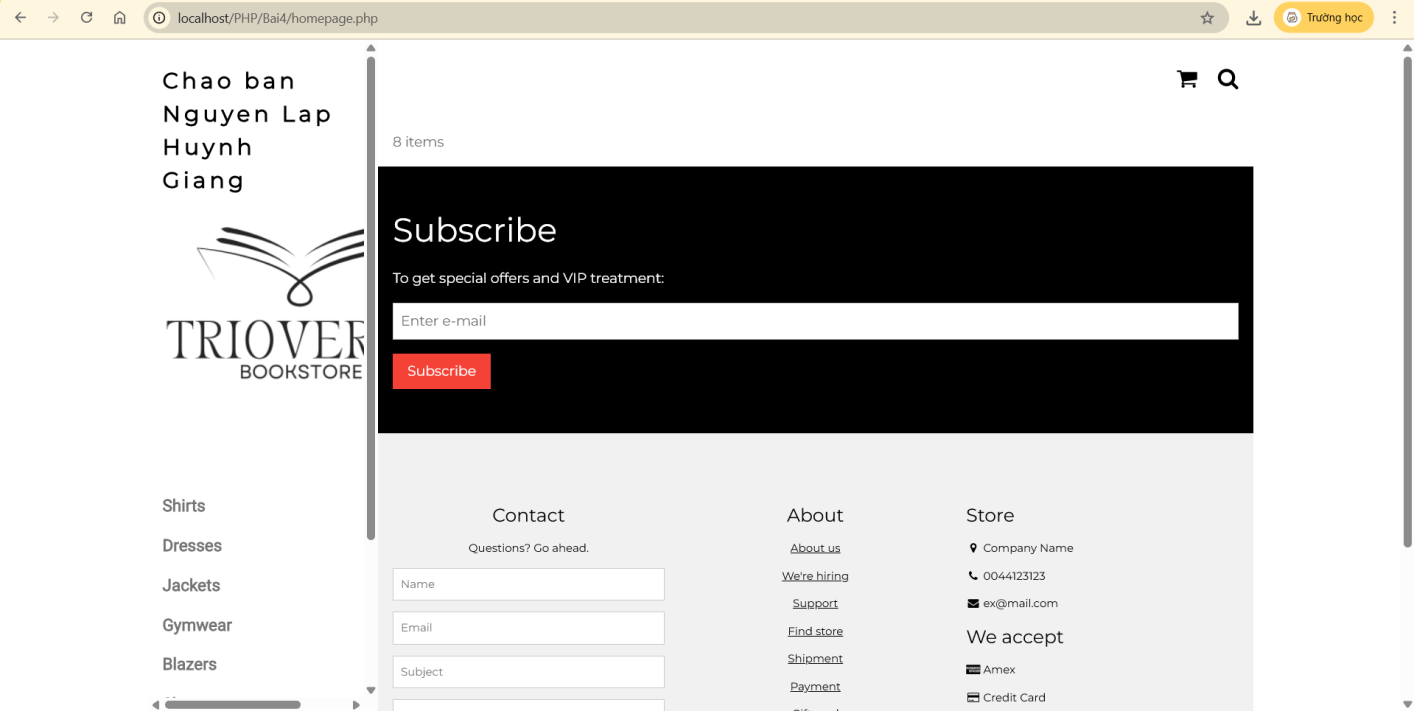
* Upload ảnh và lưu dữ liệu về ảnh trong CSDL



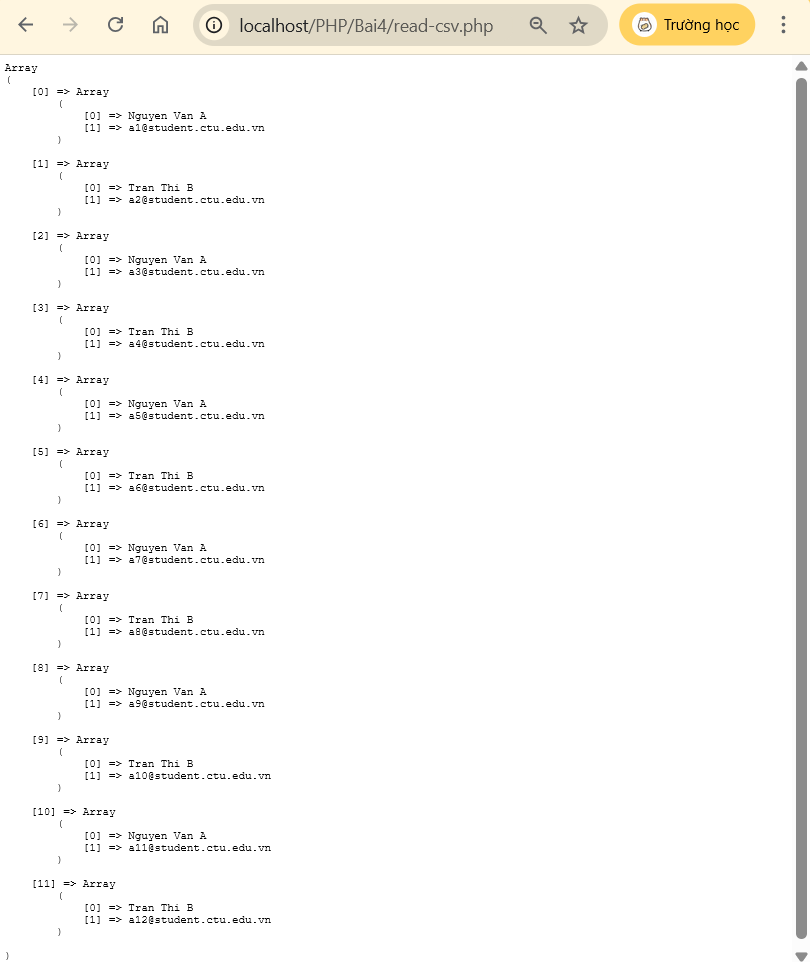
Hình 11: Thêm cột để lưu đường dẫn ảnh



Hình 12: Upload thành công vào csdl



* Upload file csv và đọc dữ liệu vào mảng

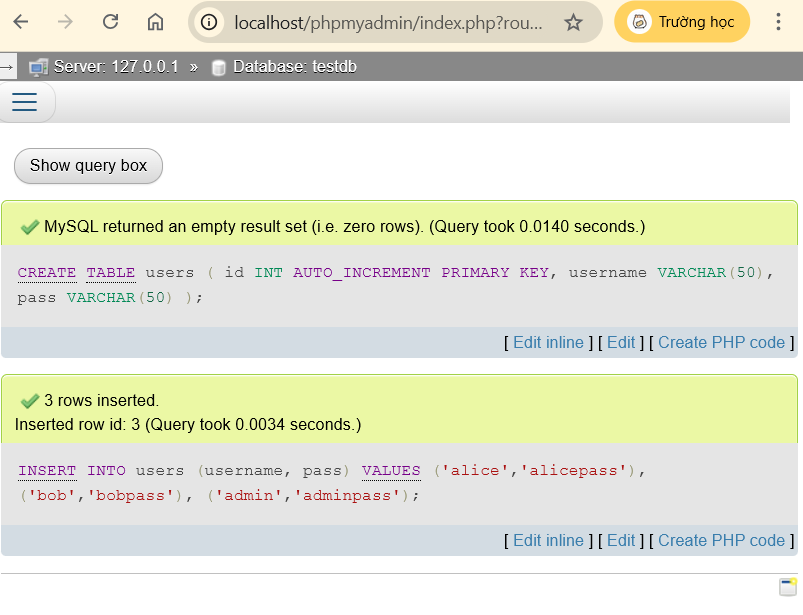


# Câu 2: Trong vài trường hợp, hacker có thể sử dụng các kỹ thuật SQL Injection để hack hệ thống của bạn. Bạn hãy trình bày SQL Injection là gì và thử các ví dụ trình bày trong 1.Ứng dụng kỹ thuật đó vào trang đăng nhập mà bạn đã tạo, chụp lại các kết quả.

* SQL injection

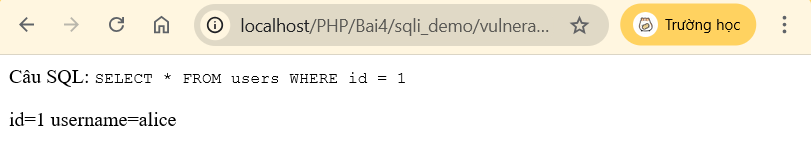
SQL injection (SQLi) là kỹ thuật lợi dụng việc ứng dụng chèn trực tiếp dữ liệu người dùng vào câu SQL mà không xử lý/kiểm tra, khiến kẻ tấn công có thể thay đổi cấu trúc truy vấn SQL. Hậu quả: đọc/ghi/xóa dữ liệu trái phép, đăng nhập bypass, lộ dữ liệu nhạy cảm.

* Các ví dụ

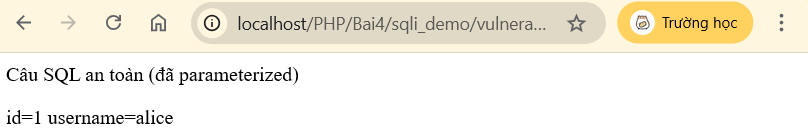


Hình 13: Tạo db test và bảng ví dụ

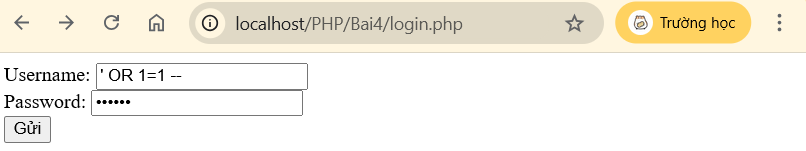
|  |
| --- |
| <?php  // vulnerable\_select.php  $conn = new mysqli('localhost','root','','testdb');  if ($conn->connect\_error) die("conn fail");  $txtUserId = $\_GET['UserId'] ?? '';  $sql = "SELECT \* FROM users WHERE id = " . $txtUserId;  echo "<p>Câu SQL: <code>{$sql}</code></p>";  $res = $conn->query($sql);  if ($res) {  while($r = $res->fetch\_assoc()) {  echo "id={$r['id']} username={$r['username']}<br>";  }  } else {  echo "Lỗi SQL: " . $conn->error;  }  $conn->close();  ?> |

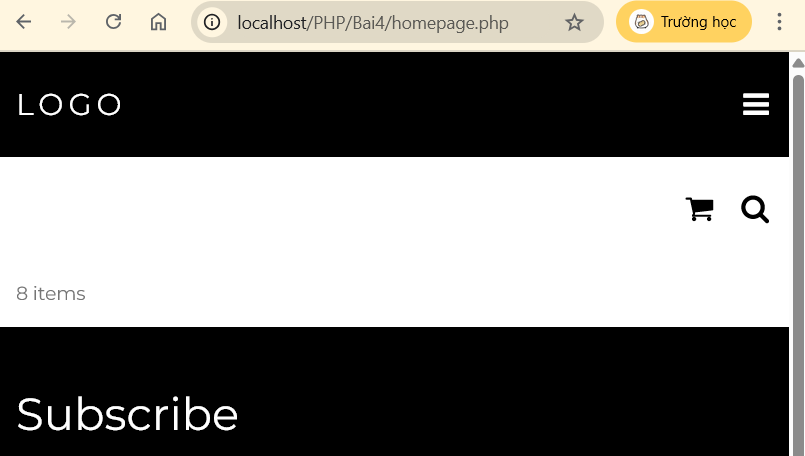


|  |
| --- |
| <?php  // safe\_select.php  $conn = new mysqli('localhost','root','','testdb');  if ($conn->connect\_error) die("conn fail");  // kiểm tra tồn tại và là số nguyên dương  if (!isset($\_GET['UserId']) || $\_GET['UserId'] === '') {  echo "Vui lòng truyền tham số UserId (ví dụ ?UserId=1)";  exit;  }  $txtUserId = $\_GET['UserId'];  // optional: kiểm tra chỉ gồm chữ số  if (!ctype\_digit($txtUserId)) {  echo "UserId không hợp lệ. Phải là số nguyên dương.";  exit;  }  // dùng prepared statement (an toàn)  $stmt = $conn->prepare("SELECT id, username FROM users WHERE id = ?");  $stmt->bind\_param("i", $txtUserId);  $stmt->execute();  $res = $stmt->get\_result();  echo "<p>Câu SQL an toàn (đã parameterized)</p>";  while($r = $res->fetch\_assoc()){  echo "id={$r['id']} username={$r['username']}<br>";  }  $stmt->close();  $conn->close(); |



* Ứng dụng vào trang đăng nhập





Hình 14: Nhập sai vẫn đăng nhập được

# Câu 3: Dựa vào link 2 , bạn hãy cho biết Cookie là gì, diễn giải ý nghĩa các tham số trong setcookie(). Cách lưu, lấy, xóa giá trị trong cookie.

* Cookie là gì?

Cookie là một tệp tin nhỏ mà máy chủ lưu trữ trên máy tính của người dùng. Nó thường được sử dụng để nhận dạng người dùng. Khi người dùng quay lại trang web, trình duyệt sẽ gửi cookie này cùng với yêu cầu, cho phép trang web nhận ra và cung cấp nội dung phù hợp với người dùng đó.

* Các tham số trong hàm setcookie()

Hàm setcookie() được dùng để tạo cookie. Dưới đây là ý nghĩa các tham số của nó:

* setcookie(name, value, expire, path, domain, secure, httponly);
* name (bắt buộc): Tên của cookie.
* value (tùy chọn): Giá trị của cookie.
* expire (tùy chọn): Thời gian hết hạn của cookie (dưới dạng Unix timestamp). Sau thời điểm này, cookie sẽ tự động bị xóa.
* path (tùy chọn): Đường dẫn trên máy chủ mà cookie sẽ có hiệu lực. Nếu đặt là "/", cookie sẽ có hiệu lực trên toàn bộ trang web.
* domain (tùy chọn): Tên miền mà cookie có hiệu lực.
* secure (tùy chọn): Nếu đặt là true, cookie chỉ được gửi qua kết nối HTTPS an toàn.
* httponly (tùy chọn): Nếu đặt là true, cookie sẽ không thể được truy cập thông qua các ngôn ngữ kịch bản phía máy khách như JavaScript.

Lưu ý: Hàm setcookie() phải được gọi trước thẻ <html>.

* **Cách lưu, lấy và xóa giá trị trong cookie**
* Cách lưu (Tạo) cookie
  + Sử dụng hàm setcookie() để tạo một cookie mới.
  + Ví dụ: Lưu một cookie tên là "user" với giá trị là "John Doe", hết hạn sau 30 ngày và có hiệu lực trên toàn bộ trang web.

setcookie("user", "John Doe", time() + (86400 \* 30), "/"); // 86400 = 1 ngày

* Cách lấy giá trị cookie
  + Bạn có thể lấy giá trị của một cookie thông qua biến toàn cục $\_COOKIE.
  + Ví dụ: Lấy và hiển thị giá trị của cookie có tên là "user". Nên dùng hàm isset() để kiểm tra xem cookie có tồn tại hay không trước khi sử dụng.

if(isset($\_COOKIE["user"])) {

echo "Chào mừng trở lại, " . $\_COOKIE["user"] . "!";

} else {

echo "Chào bạn, khách!";

}

* Cách xóa cookie
  + Để xóa một cookie, bạn chỉ cần gọi lại hàm setcookie() với thời gian hết hạn đã qua (ví dụ: trừ đi 3600 giây).
  + Ví dụ: Xóa cookie có tên là "user".

setcookie("user", "", time() - 3600);

*(Nguồn tham khảo: <https://www.w3schools.com/php/php_cookies.asp)>*

# Câu 4: Dựa vào link 3, bạn hãy cho biết Session dùng để làm gì. Cách lưu, lấy, xóa giá trị trong Session.

Session được dùng để lưu trữ thông tin của người dùng trên máy chủ (server) và truy cập thông tin đó qua nhiều trang khác nhau trong cùng một phiên làm việc. Điều này rất hữu ích cho các chức năng như đăng nhập, giỏ hàng, hoặc khi cần duy trì dữ liệu của người dùng khi họ di chuyển giữa các trang.

* **Bắt đầu một Session**
* Trước khi thực hiện bất kỳ thao tác nào với biến Session, bạn bắt buộc phải gọi hàm session\_start() ở đầu mỗi trang.

<?php

// Bắt đầu session

session\_start();

?>

<!DOCTYPE html>

<html>

<body>

...

</body>

</html>

* **Lưu giá trị vào Session**
* Để lưu trữ thông tin, chỉ cần gán giá trị cho một phần tử trong mảng $\_SESSION.
* Ví dụ: Lưu tên người dùng và email.

<?php

session\_start();

// Lưu các biến session

$\_SESSION["username"] = "Gemini AI";

$\_SESSION["email"] = "gemini@google.com";

echo "Các biến session đã được lưu.";

?>

* **Lấy giá trị từ Session**
* Để lấy lại thông tin đã lưu, bạn truy cập vào các phần tử của mảng $\_SESSION tương tự như cách đã lưu.
* Ví dụ: Lấy và hiển thị tên người dùng đã lưu ở trang khác.

<?php

session\_start();

// Lấy và hiển thị biến session

if (isset($\_SESSION["username"])) {

echo "Tên người dùng là: " . $\_SESSION["username"];

} else {

echo "Biến 'username' chưa được tạo.";

}

?>

* Có thể xem tất cả các biến đang được lưu trong session bằng cách dùng print\_r($\_SESSION);.
* **Xóa giá trị trong Session**
* Để xóa hoàn toàn một phiên làm việc, bạn cần thực hiện 2 bước:
  + session\_unset(): Xóa tất cả các biến đã được lưu trong session.
  + session\_destroy(): Hủy toàn bộ session.
* Ví dụ: Xóa tất cả thông tin và kết thúc session.

<?php

session\_start();

// Xóa tất cả các biến session

session\_unset();

// Hủy session

session\_destroy();

echo "Toàn bộ session đã được xóa.";

?>

*(Nguồn tham khảo: <https://www.w3schools.com/php/php_sessions.asp)>*

# Câu 5: Bạn hãy so sánh Cookie và Session.

* **Sự khác biệt cơ bản giữa Cookie và Session là nơi lưu trữ dữ liệu:**
* Cookie: Lưu dữ liệu trực tiếp trên máy của người dùng (trình duyệt).
* Session: Lưu dữ liệu trên máy chủ (server).

|  |  |  |
| --- | --- | --- |
|  | **Cookie** | **Session** |
| **Bảo mật** | Kém an toàn (dễ bị sửa đổi). | An toàn hơn. |
| **Dung lượng** | Rất nhỏ (khoảng 4KB). | Lớn hơn nhiều. |
| **Thời gian sống** | Lâu dài (do lập trình viên đặt). | Ngắn (thường hết hạn khi đóng trình duyệt). |