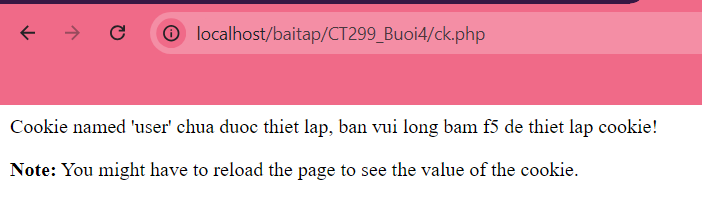
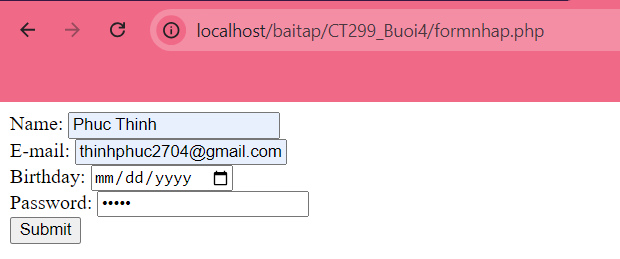
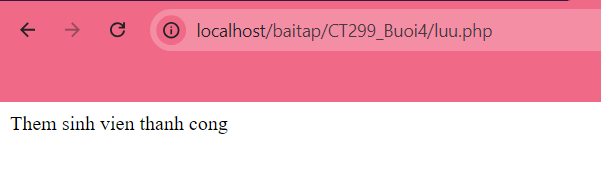
Ck.php



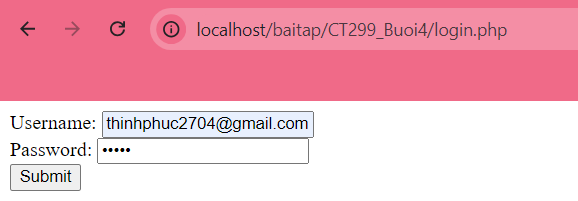
Formnhap.php



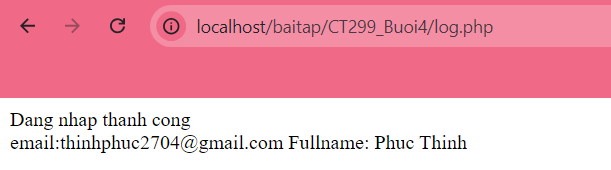
Luu.php



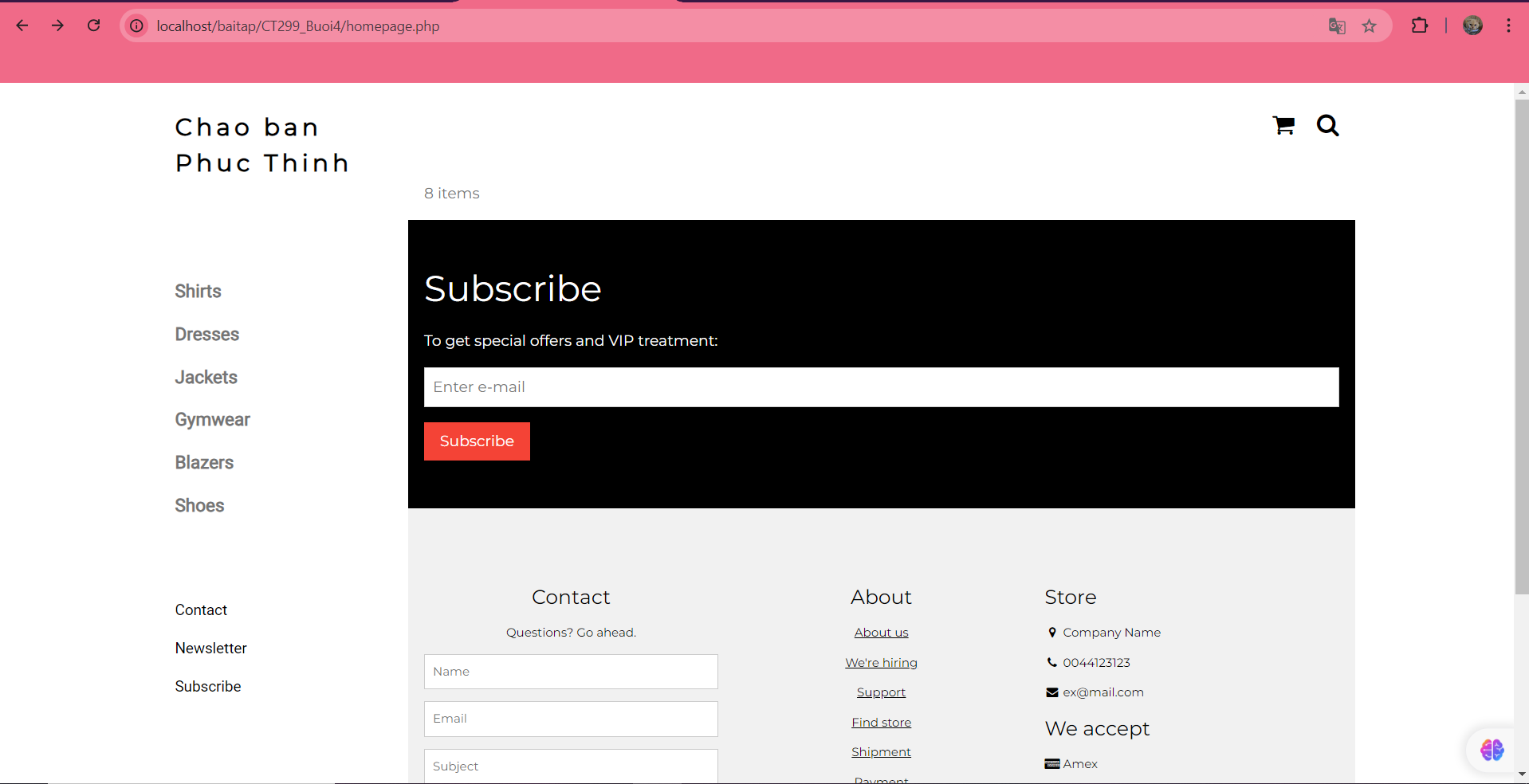
Login.php



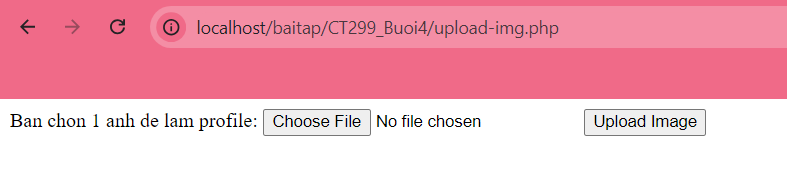
Log.php



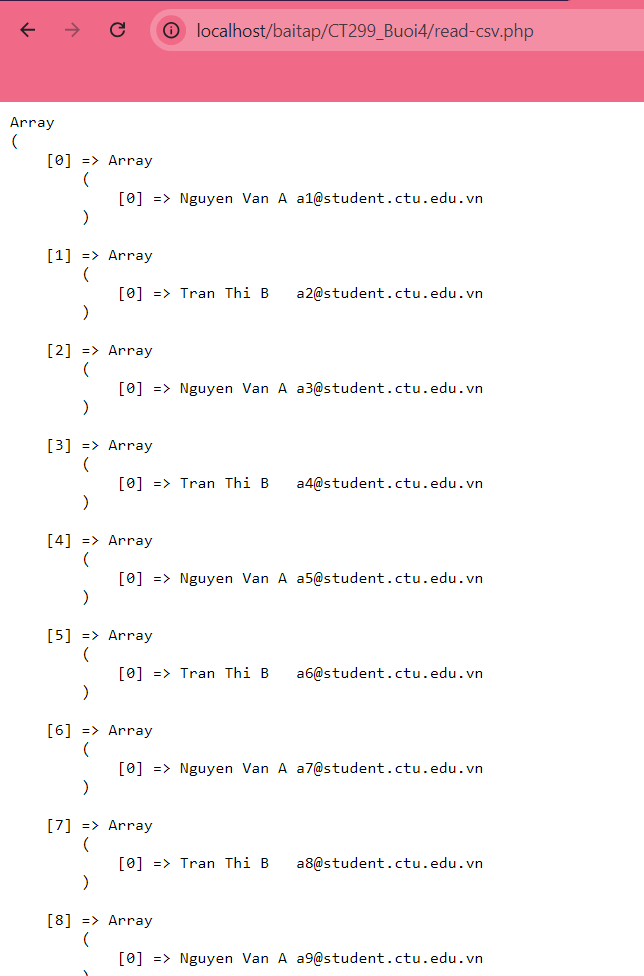
Homepage.php



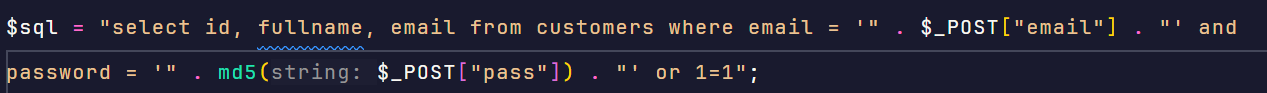
Upload-img.php

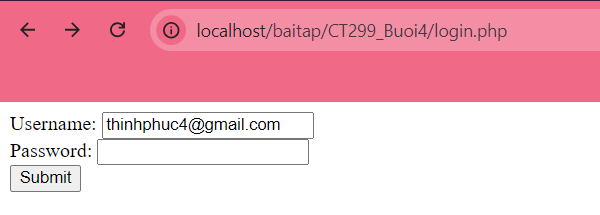


Read-csv.php

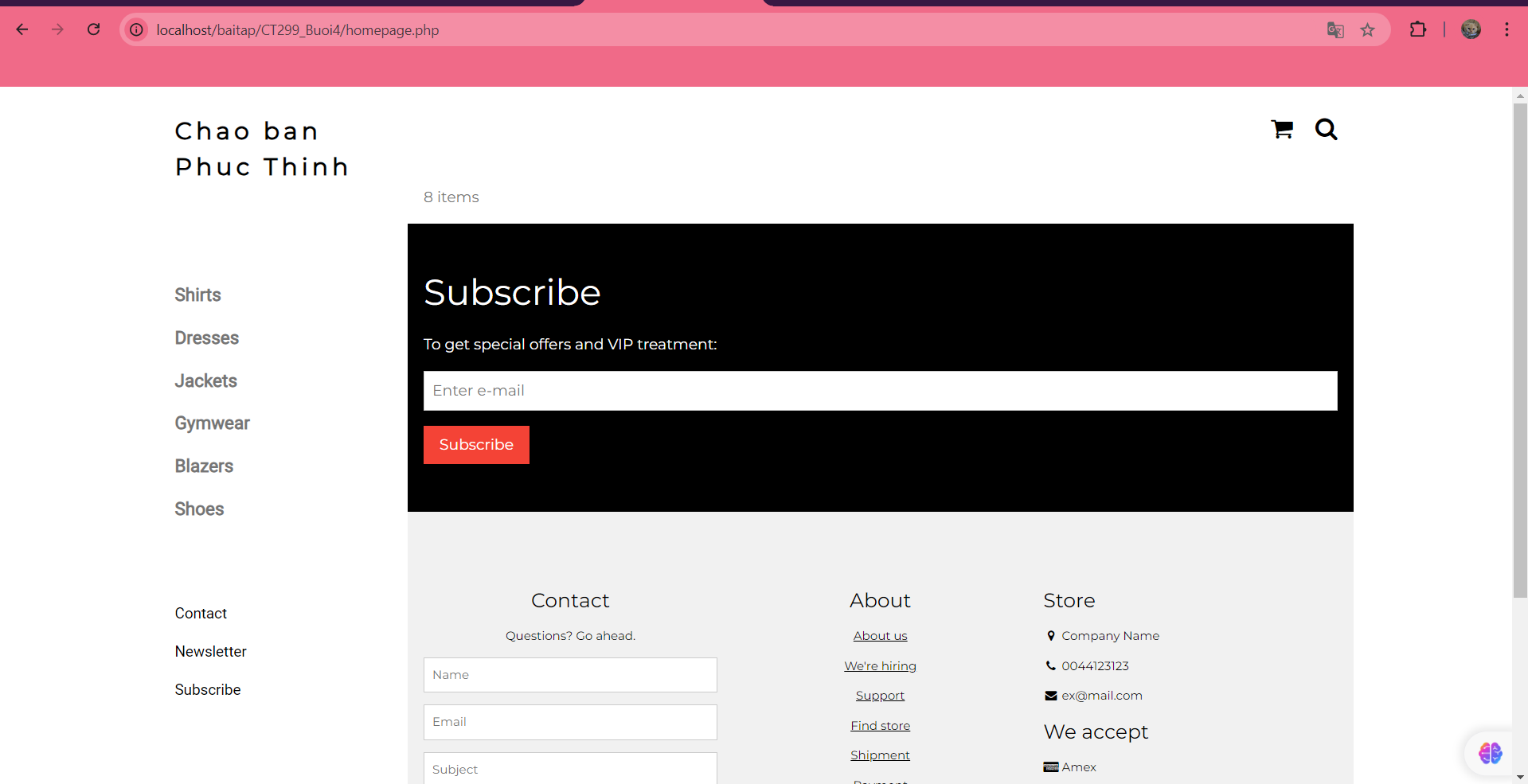


2. **SQL Injection** là một kỹ thuật tấn công mà hacker lợi dụng lỗ hổng bảo mật trong các ứng dụng web bằng cách chèn các câu lệnh SQL độc hại vào các trường nhập liệu (như form đăng nhập, thanh tìm kiếm) để thay đổi câu lệnh SQL ban đầu. Mục tiêu của SQL Injection là truy xuất, thay đổi hoặc phá hủy cơ sở dữ liệu mà không có sự cho phép. SQL Injection lợi dụng việc không có sự kiểm tra hoặc lọc dữ liệu đầu vào trước khi gửi đến cơ sở dữ liệu. Nếu dữ liệu đầu vào này được sử dụng trực tiếp trong các truy vấn SQL mà không được kiểm tra hoặc bảo mật đúng cách, kẻ tấn công có thể chèn mã SQL độc hại vào để đánh cắp thông tin hoặc phá hoại hệ thống.





Không nhập password vẫn login vào người dùng được



3. Cookie là các tệp nhỏ được lưu trữ trên máy tính của người dùng thông qua trình duyệt, chứa dữ liệu liên quan đến người dùng và các phiên làm việc. Khi người dùng truy cập trang web, trang web có thể lưu một số dữ liệu vào cookie và sau đó có thể lấy lại dữ liệu này trong các lần truy cập tiếp theo. Cookie thường được sử dụng để theo dõi thông tin người dùng như thông tin đăng nhập, tùy chọn cá nhân, giỏ hàng, hoặc trạng thái phiên làm việc.

Hàm setcookie() trong PHP được sử dụng để tạo và gửi cookie đến trình duyệt. Cú pháp của hàm setcookie(): setcookie(name, value, expire, path, domain, secure, httponly);

 **name**: (Bắt buộc) Tên của cookie. Đây là tên duy nhất được sử dụng để nhận diện cookie.

 **value**: (Tùy chọn) Giá trị của cookie. Giá trị này sẽ được lưu trữ trong cookie và có thể lấy lại sau đó. Giá trị này cần được mã hóa URL nếu chứa các ký tự đặc biệt.

 **expire**: (Tùy chọn) Thời gian cookie hết hạn, tính bằng timestamp (giây). Nếu không đặt giá trị này, cookie sẽ hết hạn khi phiên làm việc của trình duyệt kết thúc (khi người dùng đóng trình duyệt). Ví dụ: time() + 3600 tạo cookie có thời hạn 1 giờ.

 **path**: (Tùy chọn) Đường dẫn trên máy chủ mà cookie có hiệu lực. Mặc định là /, nghĩa là cookie có sẵn trên toàn bộ trang web.

 **domain**: (Tùy chọn) Tên miền mà cookie có hiệu lực. Mặc định là tên miền của trang web hiện tại. Ví dụ, bạn có thể đặt cookie có hiệu lực trên toàn bộ tên miền phụ.

 **secure**: (Tùy chọn) Nếu được đặt là true, cookie chỉ được gửi qua kết nối HTTPS an toàn. Mặc định là false.

 **httponly**: (Tùy chọn) Nếu được đặt là true, cookie chỉ có thể được truy cập thông qua HTTP và không thể truy cập thông qua JavaScript. Điều này giúp bảo vệ cookie khỏi các cuộc tấn công XSS (Cross-site scripting).

Lưu giá trị vào Cookie: Sử dụng hàm setcookie() để lưu cookie. Ví dụ, tạo một cookie tên là user với giá trị Phuc Thinh, và nó sẽ hết hạn sau 1 giờ:

<?php

$cookie\_name = "user";

$cookie\_value = "Phuc Thinh";

setcookie($cookie\_name, $cookie\_value, time() + 3600, "/");

?>

Lấy giá trị từ Cookie: có thể lấy giá trị của một cookie bằng cách truy cập mảng toàn cục $\_COOKIE. Ví dụ:

<?php

if(isset($\_COOKIE[$cookie\_name])) {

echo "Cookie '" . $cookie\_name . "' có giá trị: " . $\_COOKIE[$cookie\_name];

} else {

echo "Cookie '" . $cookie\_name . "' chưa được tạo.";

}

?>

Kết quả sẽ hiển thị giá trị của cookie user nếu nó đã được đặt.

Xóa Cookie: Để xóa một cookie cần đặt thời gian hết hạn của cookie về quá khứ. Ví dụ:

<?php

setcookie("user", "", time() - 3600, "/"); // Đặt thời gian hết hạn về quá khứ để xóa cookie

?>

Ví dụ đầy đủ về lưu, lấy và xóa cookie:

<?php

// Lưu giá trị vào cookie

$cookie\_name = "user";

$cookie\_value = "Phuc Thinh";

setcookie($cookie\_name, $cookie\_value, time() + (86400 \* 30), "/"); // Cookie hết hạn sau 30 ngày

// Lấy giá trị từ cookie

if(isset($\_COOKIE[$cookie\_name])) {

echo "Cookie '" . $cookie\_name . "' có giá trị: " . $\_COOKIE[$cookie\_name];

} else {

echo "Cookie '" . $cookie\_name . "' chưa được tạo.";

}

// Xóa cookie

setcookie($cookie\_name, "", time() - 3600, "/"); // Kích hoạt để xóa cookie

?>

4. **Session** là một cơ chế trong PHP giúp lưu trữ thông tin của người dùng trong một phiên làm việc, tồn tại trên toàn bộ trang web. Không giống như cookie, dữ liệu của session không được lưu trữ trên trình duyệt của người dùng mà được lưu trữ trên máy chủ. Mỗi phiên làm việc của người dùng được liên kết với một **Session ID** duy nhất, thường được lưu trong cookie hoặc URL để máy chủ nhận diện.

Session thường được sử dụng để lưu trữ các thông tin như:

* Thông tin đăng nhập của người dùng.
* Trạng thái giỏ hàng trong quá trình mua sắm.
* Các cài đặt tùy chỉnh của người dùng.

Lưu giá trị trong Session: có thể lưu giá trị trong session bằng cách sử dụng biến toàn cục $\_SESSION. Dữ liệu được lưu sẽ tồn tại trong toàn bộ phiên làm việc của người dùng và có thể truy cập từ bất kỳ trang PHP nào trên cùng một phiên.

<?php

session\_start();

$\_SESSION["username"] = "Phuc Thinh";

echo "Giá trị đã được lưu trong session.";

?>

Ở đây, giá trị "John Doe" được lưu vào session với khóa là "username".

**Lấy giá trị từ Session:** Để lấy giá trị đã lưu trong session, chỉ cần truy cập đến biến $\_SESSION với khóa tương ứng. VD: Lấy giá trị của username đã lưu trong session:

**<?php**

**session\_start();**

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

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

**} else {**

**echo "Tên người dùng chưa được lưu trong session.";**

**}**

**?>**

**Xóa giá trị trong Session: Để xóa một giá trị cụ thể trong session, bạn có thể sử dụng hàm unset() trên phần tử $\_SESSION. Ví dụ: Xóa giá trị của username trong session.**

**<?php**

**session\_start();**

**unset($\_SESSION["username"]);**

**echo "Đã xóa tên người dùng khỏi session.";**

**?>**

**Xóa toàn bộ Session**

**Để xóa toàn bộ dữ liệu trong session, bạn có thể sử dụng hàm session\_unset() để hủy tất cả biến session và session\_destroy() để hủy toàn bộ session, bao gồm cả Session ID.**

**<?php**

**session\_start();**

**session\_unset(); // Xóa tất cả các biến session**

**session\_destroy(); // Hủy session**

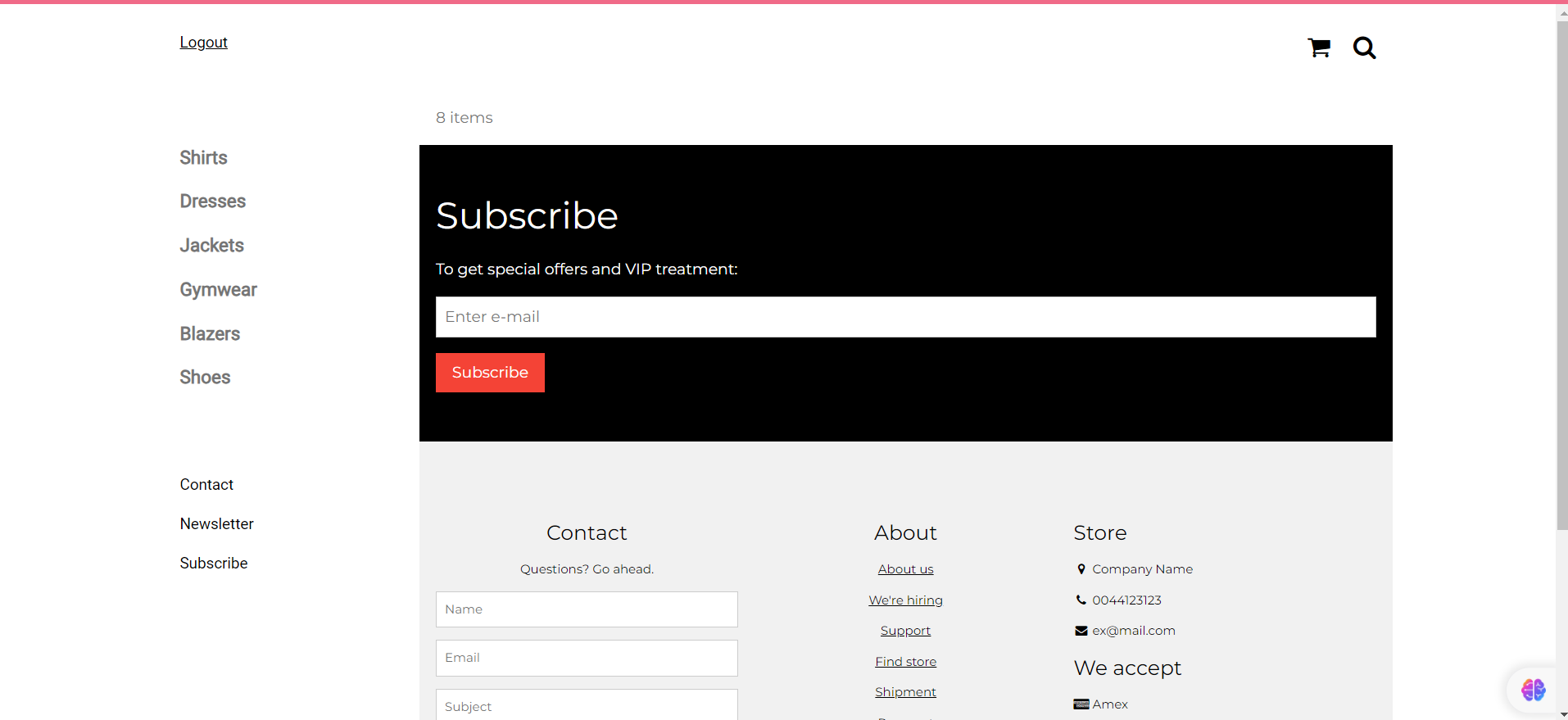
**echo "Session đã bị xóa.";**

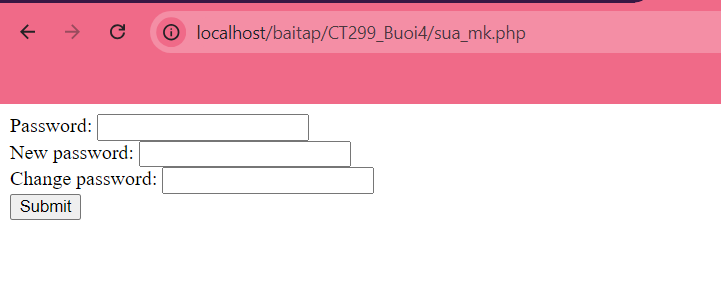
**?>**

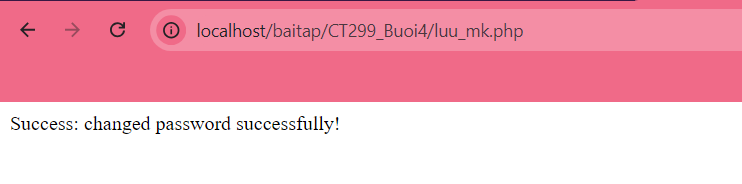
**5. So sánh cookie và session:**

| **Tiêu chí** | **Cookie** | **Session** |
| --- | --- | --- |
| **Lưu trữ dữ liệu** | **Lưu trữ trên trình duyệt của người dùng (client-side).** | **Lưu trữ trên máy chủ (server-side).** |
| **Dung lượng lưu trữ** | **Có giới hạn dung lượng (thường khoảng 4KB).** | **Không có giới hạn cố định, dung lượng phụ thuộc vào máy chủ.** |
| **Thời gian tồn tại** | **Có thể đặt thời gian tồn tại cụ thể (ngắn hạn hoặc dài hạn). Cookie có thể tồn tại ngay cả sau khi người dùng đóng trình duyệt.** | **Thường chỉ tồn tại trong phiên làm việc (session) và sẽ bị xóa khi người dùng đóng trình duyệt hoặc khi phiên hết hạn.** |
| **Bảo mật** | **Dễ bị tấn công hơn do lưu trữ trên máy của người dùng, có thể bị đánh cắp hoặc chỉnh sửa nếu không được bảo mật đúng cách.** | **Bảo mật hơn vì dữ liệu được lưu trên máy chủ, không thể bị sửa đổi trực tiếp từ phía người dùng.** |
| **Cách truyền dữ liệu** | **Cookie được gửi kèm theo mỗi yêu cầu HTTP giữa trình duyệt và máy chủ.** | **Session sử dụng Session ID, chỉ gửi Session ID qua trình duyệt, dữ liệu thực tế lưu trên máy chủ.** |
| **Sử dụng trong các trường hợp** | **Phù hợp với lưu trữ thông tin người dùng lâu dài như tùy chọn cá nhân, giỏ hàng, thông tin đăng nhập (với hạn sử dụng dài).** | **Phù hợp với các thông tin nhạy cảm, tạm thời, hoặc phiên làm việc như đăng nhập, giỏ hàng trong thời gian phiên.** |
| **Tốc độ truy cập** | **Truy cập nhanh vì dữ liệu lưu trên trình duyệt. Tuy nhiên, kích thước quá lớn có thể ảnh hưởng đến tốc độ trang.** | **Truy cập chậm hơn do dữ liệu được lưu trên máy chủ, nhưng thích hợp cho các ứng dụng bảo mật và quản lý dữ liệu lớn.** |
| **Cách lưu trữ** | **Cookie được lưu dưới dạng các cặp khóa-giá trị (key-value) trong trình duyệt.** | **Session lưu trữ dữ liệu trong các tệp hoặc cơ sở dữ liệu trên máy chủ.** |
| **Ví dụ sử dụng** | **Lưu thông tin đăng nhập tự động, ngôn ngữ ưa thích của người dùng, dữ liệu theo dõi người dùng (tracking).** | **Lưu thông tin đăng nhập của người dùng trong thời gian đăng nhập, giỏ hàng trong phiên làm việc.** |

6.



7. 



8. **Cơ chế Upload File**

* **Form HTML**: Tạo một form HTML với enctype="multipart/form-data" và phương thức POST. Đây là cách để gửi dữ liệu file từ trình duyệt đến máy chủ.
* **Hàm PHP**: Sử dụng các biến toàn cục của PHP như $\_FILES để xử lý file được gửi từ form. PHP cung cấp một loạt các biến và hàm để quản lý file upload.

Khi người dùng gửi form, dữ liệu file sẽ được gửi đến trang xử lý PHP, chẳng hạn như upload.php. Trong file PHP này, bạn sẽ sử dụng biến toàn cục $\_FILES để xử lý file.

**Các Thành Phần của $\_FILES**

* **$\_FILES['fileToUpload']['name']**: Tên gốc của file trên máy của người dùng.
* **$\_FILES['fileToUpload']['tmp\_name']**: Đường dẫn tạm thời của file trên máy chủ.
* **$\_FILES['fileToUpload']['error']**: Mã lỗi nếu có lỗi trong quá trình upload.
* **$\_FILES['fileToUpload']['size']**: Kích thước của file.

**VD:**

<?php

$target\_dir = "uploads/";

$target\_file = $target\_dir . basename($\_FILES["fileToUpload"]["name"]);

$uploadOk = 1;

$imageFileType = strtolower(pathinfo($target\_file, PATHINFO\_EXTENSION));

if (isset($\_POST["submit"])) {

$check = getimagesize($\_FILES["fileToUpload"]["tmp\_name"]);

if ($check !== false) {

echo "File is an image - " . $check["mime"] . ".";

$uploadOk = 1;

} else {

echo "File is not an image.";

$uploadOk = 0;

}

}

if (file\_exists($target\_file)) {

echo "Sorry, file already exists.";

$uploadOk = 0;

}

if ($\_FILES["fileToUpload"]["size"] > 500000) {

echo "Sorry, your file is too large.";

$uploadOk = 0;

}

if ($imageFileType != "jpg" && $imageFileType != "png" && $imageFileType != "jpeg" && $imageFileType != "gif") {

echo "Sorry, only JPG, JPEG, PNG & GIF files are allowed.";

$uploadOk = 0;

}

if ($uploadOk == 0) {

echo "Sorry, your file was not uploaded.";

} else {

if (move\_uploaded\_file($\_FILES["fileToUpload"]["tmp\_name"], $target\_file)) {

echo "The file " . htmlspecialchars(basename($\_FILES["fileToUpload"]["name"])) . " has been uploaded.";

} else {

echo "Sorry, there was an error uploading your file.";

}

}

?>

**Giải thích Mã PHP**

1. **Tạo Đường dẫn Đích**: $target\_dir và $target\_file xác định thư mục lưu trữ và tên file đích.
2. **Kiểm Tra Định Dạng File**: Sử dụng getimagesize() để kiểm tra xem file có phải là hình ảnh không.
3. **Kiểm Tra Đã Tồn Tại**: Kiểm tra xem file có đã tồn tại trong thư mục đích chưa.
4. **Kiểm Tra Kích Thước File**: Đảm bảo kích thước file không vượt quá giới hạn (500KB trong ví dụ này).
5. **Cho Phép Định Dạng File**: Chỉ cho phép các định dạng hình ảnh nhất định (JPG, JPEG, PNG, GIF).
6. **Di chuyển File**: Nếu mọi kiểm tra đều thành công, sử dụng move\_uploaded\_file() để di chuyển file từ thư mục tạm thời đến thư mục đích.

**Các Hàm/Thủ Tục Liên Quan**

* **move\_uploaded\_file()**: Di chuyển file từ thư mục tạm thời đến thư mục đích. Cú pháp: move\_uploaded\_file($tmp\_name, $target\_file).
* **getimagesize()**: Xác định xem file có phải là hình ảnh không. Cú pháp: getimagesize($filename).
* **basename()**: Trả về tên file từ đường dẫn đầy đủ. Cú pháp: basename($path).
* **pathinfo()**: Lấy thông tin về đường dẫn file. Cú pháp: pathinfo($filename, $options).

9.

**1. file()**

**Chức năng:**

* Đọc toàn bộ nội dung của một file vào một mảng, với mỗi phần tử của mảng đại diện cho một dòng trong file.

**Cú pháp:**

file(string $filename, int $flags = 0, resource $context = null)

**Tham số:**

* **$filename**: Tên của file cần đọc.
* **$flags**: Các tùy chọn điều khiển cách đọc file. Ví dụ:
  + FILE\_USE\_INCLUDE\_PATH – Tìm kiếm file trong thư mục include\_path.
  + FILE\_IGNORE\_NEW\_LINES – Không bao gồm ký tự newline vào từng dòng.
  + FILE\_SKIP\_EMPTY\_LINES – Bỏ qua các dòng trống.
* **$context**: Ngữ cảnh để điều chỉnh hành vi của file (như tùy chọn mở rộng hoặc proxy).

**Ví dụ:** $lines = file('example.txt');

foreach ($lines as $line) {

echo $line . "<br>";

}

**2. file\_get\_contents()**

**Chức năng:**

* Đọc toàn bộ nội dung của file vào một chuỗi.

**Cú pháp:**

file\_get\_contents(string $filename, bool $use\_include\_path = false, resource $context = null, int $offset = -1, int $max\_length = null)

**Tham số:**

* **$filename**: Tên của file cần đọc.
* **$use\_include\_path**: Nếu là true, tìm kiếm file trong thư mục include\_path.
* **$context**: Ngữ cảnh điều chỉnh hành vi của file.
* **$offset**: Vị trí bắt đầu đọc trong file.
* **$max\_length**: Số lượng byte tối đa cần đọc.

**Ví dụ:** $content = file\_get\_contents('example.txt');

echo $content;

**3. file\_put\_contents()**

**Chức năng:**

* Ghi dữ liệu vào file. Nếu file đã tồn tại, nó sẽ được ghi đè.

**Cú pháp:** file\_put\_contents(string $filename, mixed $data, int $flags = 0, resource $context = null)

**Tham số:**

* **$filename**: Tên của file cần ghi dữ liệu vào.
* **$data**: Dữ liệu cần ghi vào file (có thể là chuỗi, mảng, v.v.).
* **$flags**: Các tùy chọn điều khiển cách ghi dữ liệu. Ví dụ:
  + FILE\_APPEND – Thêm dữ liệu vào cuối file thay vì ghi đè.
  + LOCK\_EX – Khóa file để tránh xung đột trong khi ghi.
* **$context**: Ngữ cảnh điều chỉnh hành vi của file.

**Ví dụ:**

file\_put\_contents('example.txt', 'This is a new line of text.', FILE\_APPEND);

**4. fopen()**

**Chức năng:**

* Mở một file hoặc URL.

**Cú pháp:**

fopen(string $filename, string $mode, bool $use\_include\_path = false, resource $context = null)

**Tham số:**

* **$filename**: Tên của file hoặc URL cần mở.
* **$mode**: Chế độ mở file. Ví dụ:
  + 'r' – Đọc.
  + 'w' – Ghi (ghi đè).
  + 'a' – Ghi (thêm vào cuối).
* **$use\_include\_path**: Nếu là true, tìm kiếm file trong thư mục include\_path.
* **$context**: Ngữ cảnh điều chỉnh hành vi của file.

**Ví dụ:**

$handle = fopen('example.txt', 'r');

$content = fread($handle, filesize('example.txt'));

fclose($handle);

echo $content;

**5. fwrite()**

**Chức năng:**

* Ghi dữ liệu vào file đã mở.

**Cú pháp:**

fwrite(resource $handle, string $string, int $length = null);

**Tham số:**

* **$handle**: Tài nguyên file đã mở.
* **$string**: Dữ liệu cần ghi vào file.
* **$length**: Số byte tối đa cần ghi (nếu không chỉ định, ghi toàn bộ dữ liệu).

**Ví dụ:**

$handle = fopen('example.txt', 'w');

fwrite($handle, 'Hello World!');

fclose($handle);

**6. fgetcsv()**

Hàm fgetcsv() đọc một dòng từ file, phân tách dòng đó thành các phần tử mảng dựa trên ký tự phân cách CSV (Comma-Separated Values). Hàm này thường được sử dụng để đọc dữ liệu từ file CSV. Trả về mảng chứa các giá trị phân tách từ dòng CSV đọc được. Nếu không còn dòng nào để đọc hoặc có lỗi xảy ra, hàm trả về false.

Cú pháp:

fgetcsv(resource $handle, int $length = 0, string $delimiter = ",", string $enclosure = '"', string $escape = "\\")

**Tham số:**

* **$handle**: Tài nguyên file đã mở bằng hàm fopen(). Đây là file cần đọc dữ liệu CSV từ đó.
* **$length**: (Tùy chọn) Kích thước tối đa của dòng đọc. Nếu giá trị bằng 0, không giới hạn độ dài của dòng.
* **$delimiter**: (Tùy chọn) Ký tự phân cách các giá trị trong file CSV. Mặc định là dấu phẩy (,).
* **$enclosure**: (Tùy chọn) Ký tự được sử dụng để bao quanh các giá trị trong file CSV. Mặc định là dấu nháy kép (").
* **$escape**: (Tùy chọn) Ký tự để thoát ký tự phân cách và ký tự bao quanh trong các giá trị. Mặc định là dấu gạch chéo ngược (\).

Ví dụ về đọc tệp tin CSV (read-csv.php) sử dụng các hàm trên:

<?php

$file = fopen("read-csv.csv", "r"); // Mở tập tin csv để đọc

if ($file) {

while (($data = fgetcsv($file)) !== false) {

// Xử lý từng dòng dữ liệu trong tập tin

$name = $data[0];

$age = $data[1];

$email = $data[2];

echo "Name: " . $name . ", Age: " . $age . ", Email: " . $email .

"<br>";

}

fclose($file); // Đóng tập tin sau khi đọc xong

} else {

echo "Error opening file.";

}

?>