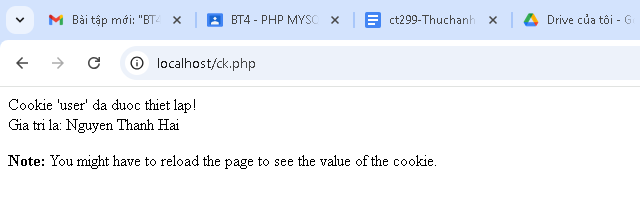
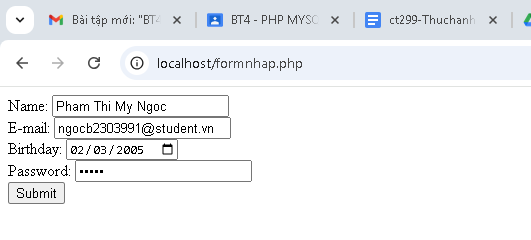
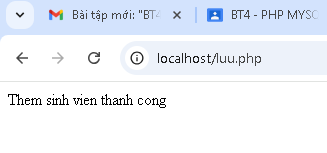
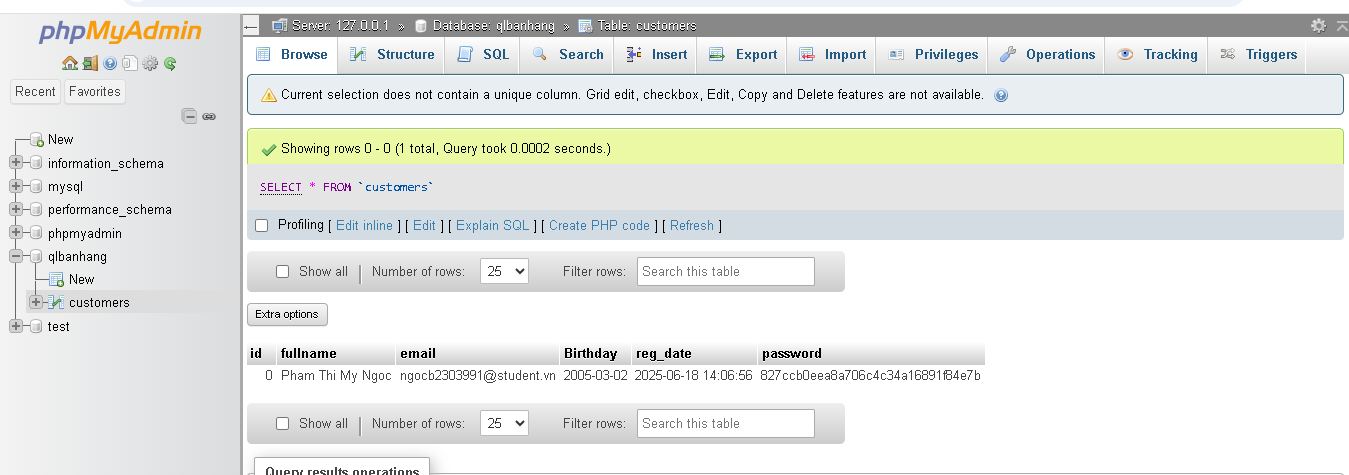
**BÀI THỰC HÀNH 04: PHP MYSQL NÂNG CAO**

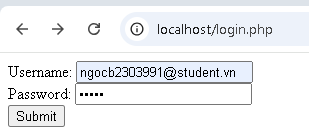
**Câu 1:**

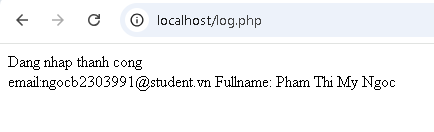


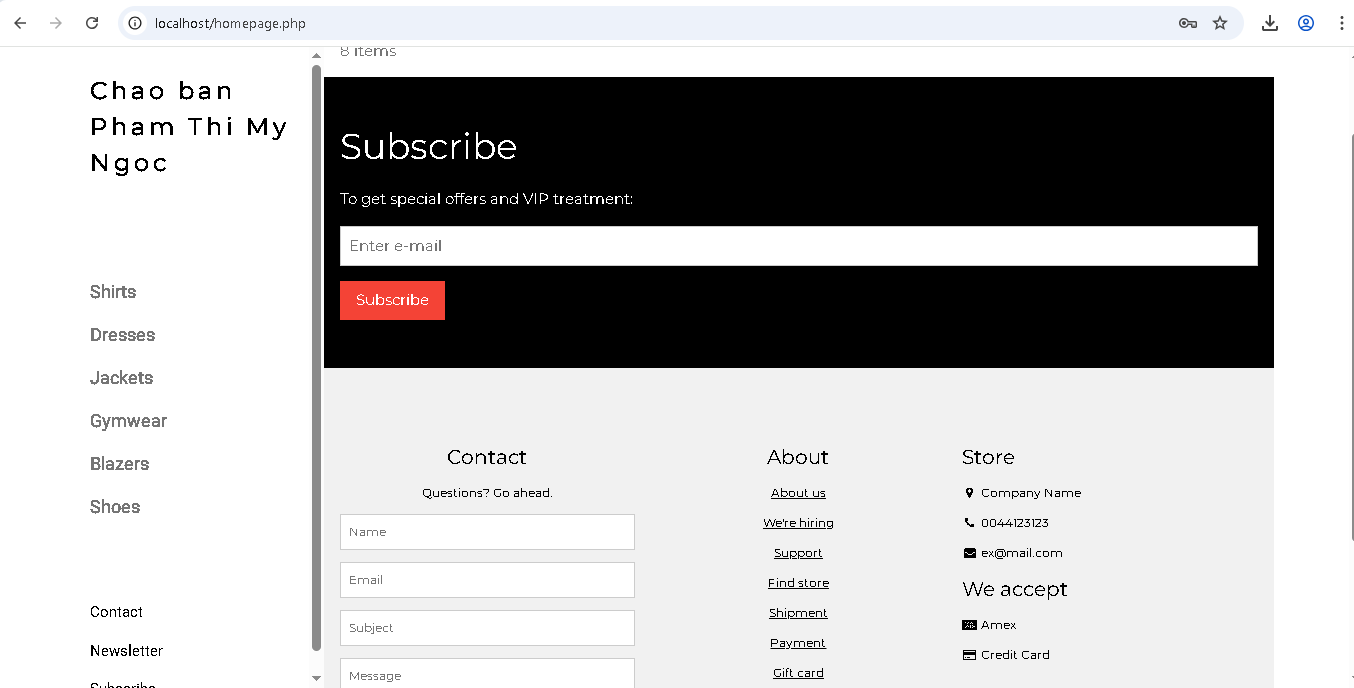


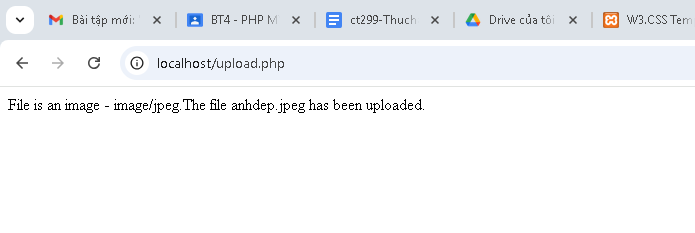


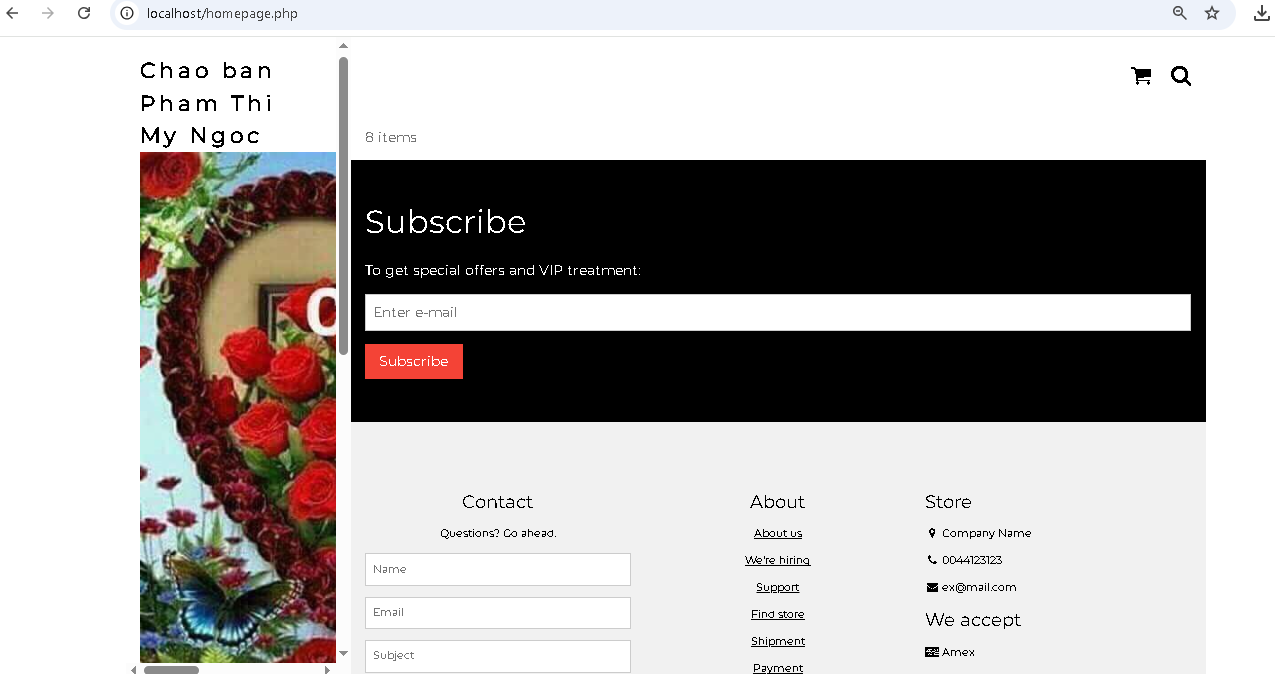






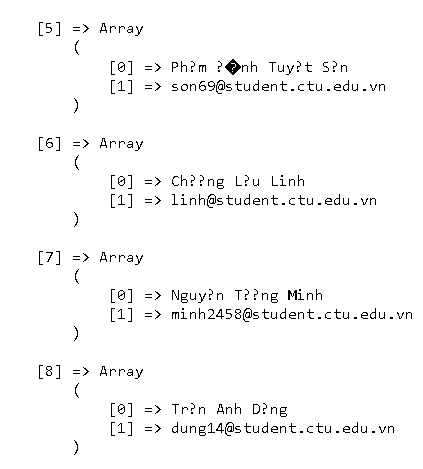








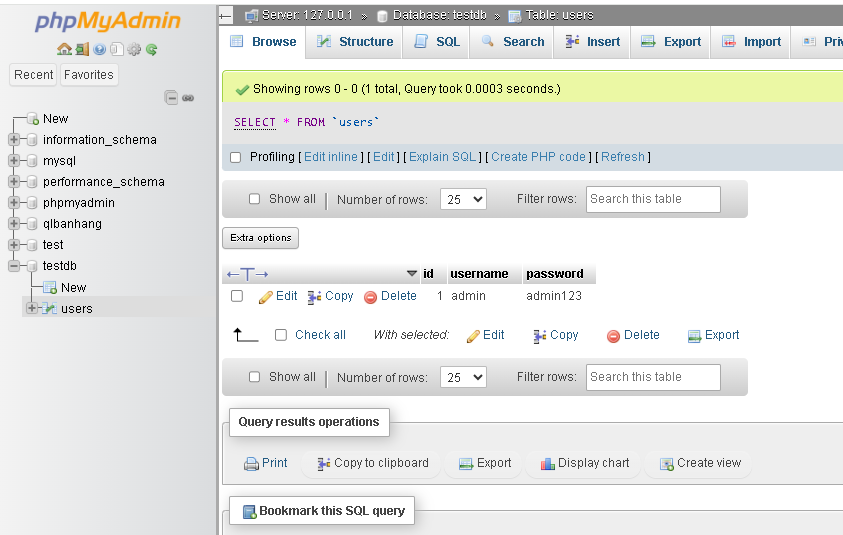


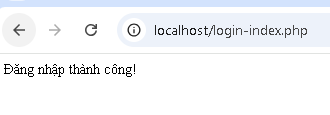
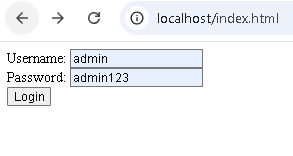


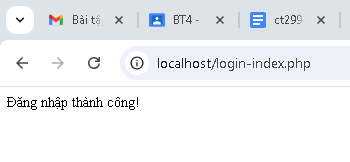
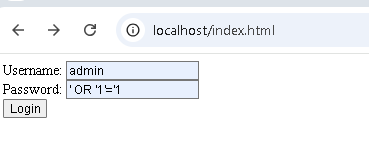
**Câu 2:**

**SQL Injection** là một kỹ thuật tấn công bảo mật, nơi kẻ tấn công "tiêm" (inject) mã SQL độc hại vào truy vấn của ứng dụng để:

* **Xem dữ liệu nhạy cảm**
* **Xóa hoặc thay đổi dữ liệu**
* **Đăng nhập trái phép vào hệ thống**
* **Thực thi lệnh SQL không mong muốn**







**Cách phòng tránh SQL Injection**Dùng Prepared Statement:

php

$stmt = $conn->prepare("SELECT \* FROM users WHERE username = ? AND password = ?");

$stmt->bind\_param("ss", $username, $password);

$stmt->execute();

$result = $stmt->get\_result();

Đoạn này **tách dữ liệu người dùng khỏi câu lệnh SQL**, khiến hacker không thể inject được.

**Câu 3:** Cookie là một đoạn dữ liệu nhỏ mà máy chủ gửi tới trình duyệt của người dùng và được lưu trữ trên máy tính của người dùng. Mỗi khi người dùng truy cập lại vào trang web đó, cookie sẽ được gửi lại cho máy chủ, giúp lưu trữ thông tin giữa các lần truy cập, ví dụ như lưu trạng thái đăng nhập hoặc tùy chọn của người dùng.

**Diễn giải các tham số trong setcookie():**

Hàm setcookie() trong PHP dùng để tạo và gửi một cookie từ máy chủ đến trình duyệt của người dùng. Cú pháp cơ bản của setcookie() như sau:

php

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

Dưới đây là ý nghĩa của các tham số:

1. **name**: Tên của cookie (chuỗi văn bản). Đây là tham số bắt buộc.
   * Ví dụ: "user", "session\_id"
2. **value**: Giá trị của cookie (chuỗi văn bản). Đây là tham số bắt buộc.
   * Ví dụ: "John Doe", "12345"
3. **expire**: Thời gian hết hạn của cookie, tính bằng giây kể từ thời điểm hiện tại. Nếu không xác định, cookie sẽ chỉ tồn tại trong phiên làm việc của trình duyệt (cookie phiên).
   * Ví dụ: time() + 3600 (hết hạn sau 1 giờ).
4. **path**: Đường dẫn trên máy chủ mà cookie sẽ có hiệu lực. Mặc định là /, tức là cookie sẽ có hiệu lực trên toàn bộ trang web.
   * Ví dụ: "/admin" (cookie chỉ có hiệu lực trên các trang con của /admin).
5. **domain**: Miền mà cookie sẽ có hiệu lực. Nếu không đặt, cookie sẽ chỉ có hiệu lực trong cùng một miền mà nó được tạo ra.
   * Ví dụ: "example.com" (cookie có hiệu lực trên toàn bộ miền example.com và các subdomain).
6. **secure**: Nếu tham số này được đặt là true, cookie chỉ được gửi qua kết nối HTTPS (an toàn). Nếu không đặt, cookie có thể được gửi qua HTTP.
   * Ví dụ: true (chỉ gửi qua HTTPS).
7. **httponly**: Nếu tham số này được đặt là true, cookie chỉ có thể được truy cập qua giao thức HTTP, không thể truy cập từ JavaScript. Điều này giúp tăng cường bảo mật, tránh tấn công XSS (Cross-Site Scripting).
   * Ví dụ: true (không thể truy cập cookie từ JavaScript).

**Cách lưu, lấy và xóa giá trị trong cookie:**

1. **Lưu giá trị vào cookie**:  
   Để lưu một giá trị vào cookie, bạn sử dụng hàm setcookie().

php

setcookie("username", "John Doe", time() + 3600, "/"); // Lưu cookie với tên "username" và giá trị "John Doe"

Trong ví dụ trên, cookie "username" sẽ hết hạn sau 1 giờ (3600 giây) và có hiệu lực trên toàn bộ trang web ("/").

1. **Lấy giá trị từ cookie**:  
   Để lấy giá trị của cookie, bạn có thể truy cập thông qua mảng $\_COOKIE trong PHP.

php

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

echo "Hello " . $\_COOKIE["username"];

} else {

echo "Cookie not set!";

}

Ở đây, nếu cookie "username" đã được lưu, nó sẽ hiển thị "Hello John Doe". Nếu không, nó sẽ in ra thông báo "Cookie not set!".

1. **Xóa cookie**:  
   Để xóa cookie, bạn có thể đặt thời gian hết hạn của cookie trong quá khứ (ví dụ: time() - 3600). Điều này sẽ yêu cầu trình duyệt xóa cookie đó.

php

setcookie("username", "", time() - 3600, "/"); // Xóa cookie "username"

Ví dụ trên sẽ xóa cookie "username" bằng cách thiết lập giá trị hết hạn là thời gian đã qua (trong quá khứ).

**Câu 4: Session** trong PHP là một cơ chế lưu trữ thông tin tạm thời cho người dùng trong suốt phiên làm việc với website. Mục đích chính của session là lưu trữ thông tin mà bạn muốn duy trì giữa các lần tải lại trang web mà không cần phải sử dụng cookie.

### **Mục đích sử dụng Session:**

* **Lưu trữ thông tin tạm thời**: Các dữ liệu như thông tin đăng nhập (username, userID) hoặc dữ liệu giỏ hàng có thể được lưu trữ trong session để sử dụng trong suốt phiên duyệt web mà không phải lưu vào cơ sở dữ liệu hay cookie.
* **Duy trì trạng thái người dùng**: Session giúp bạn duy trì trạng thái giữa các trang mà không cần phải truyền tải thông tin qua URL hoặc form (như khi dùng cookie).
* **Bảo mật hơn so với cookie**: Dữ liệu trong session được lưu trên máy chủ, không phải trên máy của người dùng (như cookie), giúp bảo mật hơn.

### **Cách sử dụng Session trong PHP:**

#### **1. Khởi tạo và lưu giá trị vào Session**:

Trước khi sử dụng session, bạn cần gọi session\_start() ở đầu mỗi trang PHP (nếu chưa khởi tạo session). Điều này giúp PHP nhận diện phiên làm việc hiện tại của người dùng.

php

<?php

session\_start(); // Khởi tạo session

// Lưu giá trị vào session

$\_SESSION["username"] = "JohnDoe";

$\_SESSION["user\_id"] = 12345;

?>

* **session\_start()**: Bắt buộc phải gọi ở đầu trang để bắt đầu hoặc tiếp tục một session.
* **$\_SESSION[]**: Mảng siêu toàn cục này dùng để lưu trữ các giá trị trong session. Các giá trị trong $\_SESSION sẽ tồn tại suốt phiên làm việc của người dùng.

#### **2. Lấy giá trị từ Session**:

Để lấy giá trị từ session, bạn truy cập vào mảng $\_SESSION như khi bạn truy cập một mảng bình thường.

php

<?php

session\_start(); // Khởi tạo session

// Lấy giá trị từ session

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

echo "Hello, " . $\_SESSION["username"];

} else {

echo "User is not logged in.";

}

?>

Trong ví dụ này, nếu giá trị "username" tồn tại trong session, nó sẽ được in ra. Nếu không, sẽ hiển thị thông báo "User is not logged in."

#### **3. Xóa giá trị khỏi Session**:

Bạn có thể xóa một giá trị cụ thể trong session bằng cách dùng unset().

php

<?php

session\_start(); // Khởi tạo session

// Xóa một giá trị trong session

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

?>

Khi gọi unset(), giá trị tương ứng trong session sẽ bị xóa. Nếu bạn muốn xóa tất cả các giá trị trong session, bạn có thể dùng:

php

<?php

session\_start(); // Khởi tạo session

// Xóa tất cả các giá trị trong session

session\_unset();

?>

#### **4. Xóa toàn bộ Session**:

Nếu bạn muốn hủy toàn bộ session (ví dụ như khi người dùng đăng xuất), bạn sẽ cần gọi session\_destroy() để xóa session hiện tại và tất cả dữ liệu liên quan.

php

<?php

session\_start(); // Khởi tạo session

// Xóa tất cả dữ liệu trong session

session\_unset();

// Hủy session

session\_destroy();

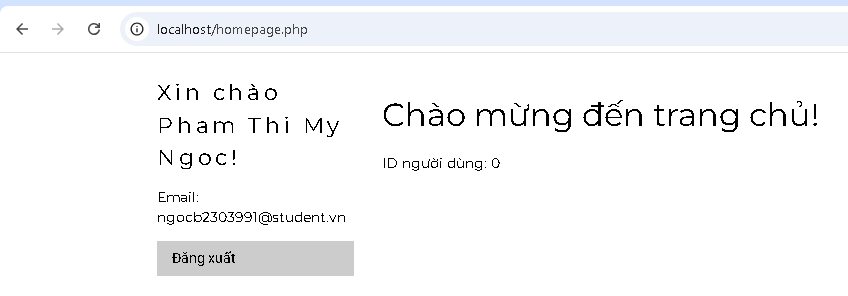
?>

* **session\_unset()**: Xóa tất cả các biến trong session nhưng không hủy session.
* **session\_destroy()**: Hủy session hiện tại, xóa mọi thông tin lưu trữ trong session, và đóng session đó lại.

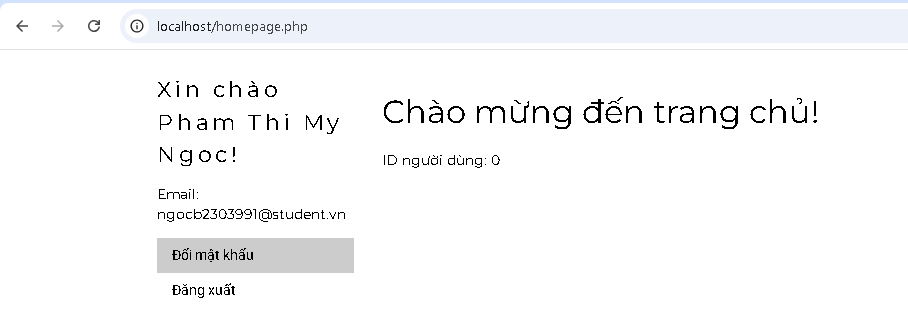
**Câu 5**:

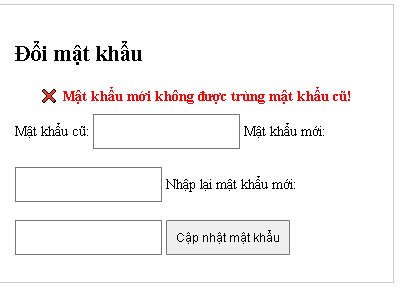
| **Đặc điểm** | **Cookie** | **Session** |
| --- | --- | --- |
| **Vị trí lưu trữ** | Máy của người dùng (client-side) | Máy chủ (server-side) |
| **Thời gian tồn tại** | Tùy vào thời gian hết hạn | Tồn tại đến khi phiên làm việc kết thúc hoặc session hết hạn |
| **Dung lượng** | Giới hạn (khoảng 4KB mỗi cookie) | Không giới hạn, phụ thuộc vào bộ nhớ máy chủ |
| **Bảo mật** | Thấp (dễ bị đánh cắp nếu không bảo vệ) | Cao hơn (dữ liệu lưu trên máy chủ) |
| **Quản lý dữ liệu** | Người dùng có thể sửa/xóa cookie | Dữ liệu không thể truy cập từ client, chỉ có server quản lý |
| **Phạm vi sử dụng** | Lưu trữ dữ liệu lâu dài (ví dụ: ID người dùng, sở thích) | Lưu trữ dữ liệu tạm thời trong suốt phiên làm việc (ví dụ: thông tin đăng nhập) |
| **Sử dụng phổ biến** | Theo dõi hành vi người dùng, lưu trữ thông tin người dùng giữa các lần truy cập | Quản lý phiên người dùng, bảo mật thông tin trong quá trình tương tác với website |

**Câu 6:**



**Câu 7:**









**Câu 8**:

### **1. HTML Form để upload tệp**

Đầu tiên, bạn cần tạo một form HTML với thuộc tính enctype="multipart/form-data" để hỗ trợ việc tải lên tệp.

html

<form action="upload.php" method="post" enctype="multipart/form-data">

<input type="file" name="fileToUpload" id="fileToUpload">

<input type="submit" value="Upload File" name="submit">

</form>

### **2. PHP xử lý tệp upload**

Trong tệp PHP (upload.php), bạn xử lý và kiểm tra tệp tải lên từ $\_FILES.

php

<?php

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

$target\_dir = "uploads/";

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

$uploadOk = 1;

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

// Kiểm tra nếu tệp là ảnh

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

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

if ($check === false) {

echo "File is not an image.";

$uploadOk = 0;

}

}

// Kiểm tra các điều kiện khác (kích thước, loại tệp, tồn tại file)

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

echo "File already exists.";

$uploadOk = 0;

}

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

echo "File is too large.";

$uploadOk = 0;

}

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

echo "Invalid file type.";

$uploadOk = 0;

}

// Nếu tất cả kiểm tra đều ổn, tải lên tệp

if ($uploadOk == 1 && move\_uploaded\_file($\_FILES["fileToUpload"]["tmp\_name"], $target\_file)) {

echo "File has been uploaded.";

} else {

echo "Error uploading file.";

}

}

?>

### **Các bước chính trong PHP:**

1. **Kiểm tra lỗi tệp**: Đảm bảo không có lỗi trong quá trình upload.
2. **Kiểm tra loại tệp**: Kiểm tra xem tệp có phải là ảnh hay không.
3. **Kiểm tra kích thước và phần mở rộng**: Đảm bảo tệp không quá lớn và có phần mở rộng hợp lệ.
4. **Di chuyển tệp**: Dùng move\_uploaded\_file() để chuyển tệp từ thư mục tạm thời đến thư mục đích.

### **Bảo mật:**

* **Kiểm tra loại tệp**: Chỉ cho phép các định dạng tệp hợp lệ (như hình ảnh).
* **Giới hạn kích thước tệp**: Hạn chế kích thước tệp để tránh ảnh hưởng đến server.
* **Đổi tên tệp**: Để tránh trùng tên, bạn có thể đổi tên tệp khi lưu vào server.

**Câu 9:**

**Hàm file() trong PHP** dùng để đọc nội dung tệp và trả về mỗi dòng dưới dạng một phần tử trong mảng.

### **Chức năng:**

* Đọc nội dung tệp và trả về các dòng trong tệp dưới dạng mảng.

### **Cú pháp:**

php

$file = file("filename", $flags, $context);

### **Các tham số:**

1. **filename** (bắt buộc): Đường dẫn tệp cần đọc.
2. **flags** (tuỳ chọn):
   * FILE\_USE\_INCLUDE\_PATH: Tìm tệp trong include\_path.
   * FILE\_IGNORE\_NEW\_LINES: Loại bỏ ký tự newline (\n) ở cuối mỗi dòng.
   * FILE\_SKIP\_EMPTY\_LINES: Bỏ qua các dòng trống.
3. **context** (tuỳ chọn): Ngữ cảnh mở tệp, ít khi sử dụng.

### **Ví dụ:**

* Đọc tệp và in từng dòng:

php

$file = file("data.txt");

foreach ($file as $line) {

echo $line . "<br>";

}

* Đọc tệp mà không có ký tự newline:

php

$file = file("data.txt", FILE\_IGNORE\_NEW\_LINES);

### **Lưu ý**:

* **file()** tải toàn bộ tệp vào bộ nhớ, không phù hợp với tệp lớn. Dùng fopen() hoặc fgets() cho tệp lớn.

**Câu 10:**

