University of Anbar

College of Computer Sciences and

Information Technology

Web Development

Dr. Alaa Abdalqahar Jihad

Fourth Stage – Second Course

Lecture 9: File Uploads and Image Handling

Objectives:

- Understand how file uploads work in web applications.
- Build HTML forms that allow file selection and upload.
- Use PHP's $_FILES superglobal to handle file uploads.
- Store uploaded files on the server using move_uploaded_file().

Introduction to File Uploads

- **Purpose and Use Cases:**
    - Enable users to upload profile pictures, documents, and other media.
    - How file uploads integrate with user-generated content.
- **Overview of the Process:**
    - User selects a file via an HTML form.
    - PHP processes and validates the file.
    - File is moved to a designated server directory.
- **Real-World Examples:** Social media platforms, content management systems, e-commerce sites.

Creating an HTML Form for File Uploads

- **Form Setup:** Use the enctype="multipart/form-data" attribute in the <form> tag.

Example Form Code:

```
<form action="upload.php" method="POST" enctype="multipart/form-data">
    <label for="file">Choose an image:</label>
    <input type="file" name="file" id="file">
    <input type="submit" value="Upload">
</form>
```

- **Discussion:** Importance of method="POST" and enctype.

Handling File Uploads in PHP

- **Understanding the $_FILES Array:**
    - Structure: $_FILES['file']['name'], $_FILES['file']['type'], $_FILES['file']['tmp_name'], $_FILES['file']['error'], and $_FILES['file']['size'].

Basic File Processing Code: (upload.php)

```
<?php
if ($_SERVER['REQUEST_METHOD'] == 'POST') {
    // Check for errors
    if ($_FILES['file']['error'] === UPLOAD_ERR_OK) {
        $tmpName = $_FILES['file']['tmp_name'];
        $fileName = basename($_FILES['file']['name']);
        $uploadDir = "uploads/";
        $uploadFile = $uploadDir . $fileName;

        // Move the file from temporary location to target directory
        if (move_uploaded_file($tmpName, $uploadFile)) {
            echo "File uploaded successfully!";
        } else {
```

```
            echo "Failed to upload file.";
        }
    } else {
        echo "Error during file upload.";
    }
}
?>
```

Summary

- How file uploads work via HTML forms and PHP's $_FILES array.

Homework Assignment:

Create a PHP script that: Implements a file upload form.

Lecture 10: Building a Complete PHP + MySQL Project

Complete Project Requirements:

- **Project Concept:** Build a simple web application (e.g., user management system or basic blog) that supports user registration, login/logout, profile management, and data display.
- **Functional Requirements:**
    - Implement user registration and login.
    - Display a list of items (users, posts, products, etc.).
    - Enable CRUD operations on the data.
- **Non-Functional Requirements:**
    - Ensure security through input validation, data sanitization, and prepared statements.
    - Maintain a modular and organized code structure.
- **Project Architecture:**
    - **Database:** Create a MySQL database (e.g., project_db) with a Users table (id, username, email, hashed password, created_at) and so on.
    - **Application Structure:** Use HTML/CSS/JavaScript for the front-end and PHP for the back-end (e.g., register.php, login.php, dashboard.php, logout.php) with a dedicated configuration file for the database connection.
    - **File Organization:** Organize files into folders like includes/ (for reusable components) and assets/ (for CSS/JS).

- **Development Environment:**
    - Install and configure XAMPP.
    - Create the database and tables using phpMyAdmin or command-line tools.
    - Establish a central database connection file (db.php).
- **Core Features Implementation:**
    - **User Registration:** Create a registration page with input validation, prepared statements, and password hashing using password_hash().
    - **User Login:** Develop a login page to verify credentials with password_verify() and set session variables.
    - **Dashboard & CRUD:** Build a dashboard to display data and implement secure update/delete operations using prepared statements.
    - **Session Management:** Start sessions on protected pages, validate session data, and implement a logout script to destroy sessions.
- **Security Best Practices:**
    - Validate and sanitize all inputs using functions like htmlspecialchars() and filter_var().
    - Prevent SQL injection with prepared statements.
- **Integration & Testing:** Build and test the application to ensure all features (registration, login, CRUD, logout) work as expected and debug any issues.

- **Documentation:** Prepare documentation outlining the project structure, setup procedures, and key features.