UNIVERSITI TUNKU ABDUL RAHMAN

FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY

UCCD3243 SERVER-SIDE WEB APPLICATIONS DEVELOPMENT

PRACTICAL 6

File Upload and Processing

Objectives:

1. Create file upload form
2. Handle file uploads with PHP
3. Testing and troubleshooting

Instructions: Continue working with the same project file you set up in the previous practical class, use the same PHP files saved in “product” folder. Make sure you are logged in to access CRUD functionalities.

Create file upload form

Step 1: Create file for upload form

- Create a new PHP page named “file_manager.php”

```
<?php
include("auth.php");
require('database.php');

//File Upload Section

//File Delete Section
?>
<!DOCTYPE html>
<html>
<head>
   <meta charset="utf-8">
   <title>File Manager</title>
</head>
<body>
 <p><a href="dashboard.php">User Dashboard</a> |
 <a href="logout.php">Logout</a></p>
   <h1>File Manager</h1>
    <!-- Form for File Upload section -->
   <form enctype="multipart/form-data" method="post" action="">
     <input type="text" name="user_input" placeholder="Add comment or note" required
/>
     <input type="file" name="file" required /><br><br>
     <input type="submit" name="upload" value="Upload File" />
   </form>
</body>
</html>
```

Why use multipart/form-data?

Table 1

Step 2: Add “file_manager.php” link in “dashboard.php”

- In “dashboard.php”, add the following code:

```
<p><a href="file_manager.php">Upload File</a></p>
```

Table 2

- User should be able to click Upload File in the user dashboard.

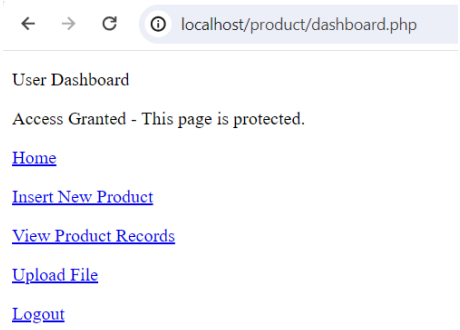


Figure 1

Step 3: Add new table in “product_db” database

- Go to phpMyAdmin, create a new table and name it as “files”.

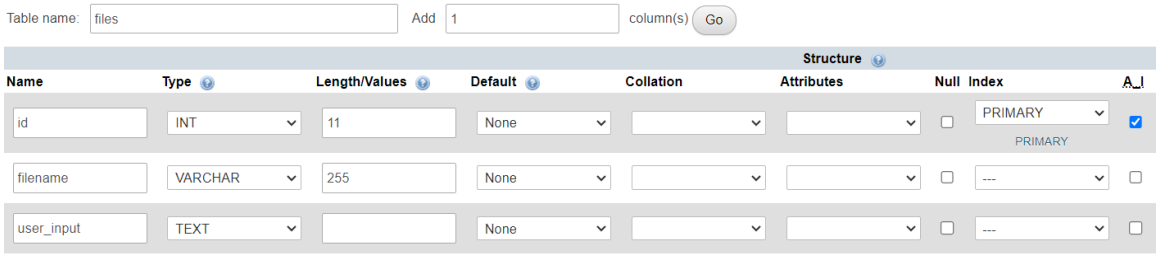


Figure 2

- Click save after adding all the attributes.

Handle file uploads with PHP

Step 4: Upload file using PHP

- Create an “upload” folder in your project directory where the uploaded files will be stored.

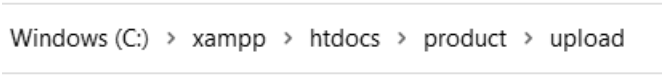


Figure 3

- In “file_manager.php”, add the following code to save file information to the database, right after the comment “//File Upload Section”.

```
if (isset($_POST['upload'])) {
    $uploadedFileName = $_FILES['file']['name'];
    $targetDirectory = "upload/";
    $targetFilePath = $targetDirectory . $uploadedFileName;

    if (move_uploaded_file($_FILES['file']['tmp_name'], $targetFilePath)) {
        $userInput = $_POST['user_input'];

        $insertQuery = "INSERT INTO files (filename, user_input) VALUES ('$uploadedFileName', '$userInput')";
        mysqli_query($con, $insertQuery) or die(mysqli_error($con));

        $status = "File uploaded successfully.";
    } else {
        $status = "File upload failed.";
    }
}
```

What are the key parameters of PHP $_FILES array?

How does it handle file upload process?

Table 3

- User should be able to upload file of any format, including images, documents, audio files, and more, along with a text input field.

Testing and troubleshooting

Step 5: Display uploaded file

- In “file_manager.php”, add the following code to display the uploaded files right after the “Form for File Upload” section.

```
<?php
    $filesQuery = "SELECT * FROM files";
    $filesResult = mysqli_query($con, $filesQuery);

    if ($filesResult) {
    while ($fileRow = mysqli_fetch_assoc($filesResult)) {
        echo "<li>";
        echo $fileRow['filename'];
        echo " - Comment or note: " . $fileRow['user_input'];
        echo " <a href='upload/" . $fileRow['filename'] . "' target='_blank'>View</a>";
        echo "</li>";
    }
  }
?>
```

How to retrieve the file from database and display the file?

Table 4

- User can click the “View” link to open the uploaded file in a new browser tab or window.
- Under the “View” link, add “Delete” link to allow user to delete the uploaded file in “upload” folder and database.

```
echo " | <a href='file_manager.php?delete=" . $fileRow['id'] . "' onclick=\"return
confirm('Are you sure you want to delete this file?')\">Delete</a>";
```

Table 5

Step 6: Delete uploaded file

- In “file_manager.php”, add the following code to handle file deletion, right after the comment “//File Delete Section”.

```
if (isset($_GET['delete'])) {
    $fileId = $_GET['delete'];

    $selectQuery = "SELECT filename FROM files WHERE id = $fileId";
    $result = mysqli_query($con, $selectQuery);
    $row = mysqli_fetch_assoc($result);
    $filename = $row['filename'];

    $filePath = "upload/" . $filename;
    if (file_exists($filePath) && !is_dir($filePath)) {
        unlink($filePath);
    }
```

What happens to the file in database and upload folder once it is deleted?

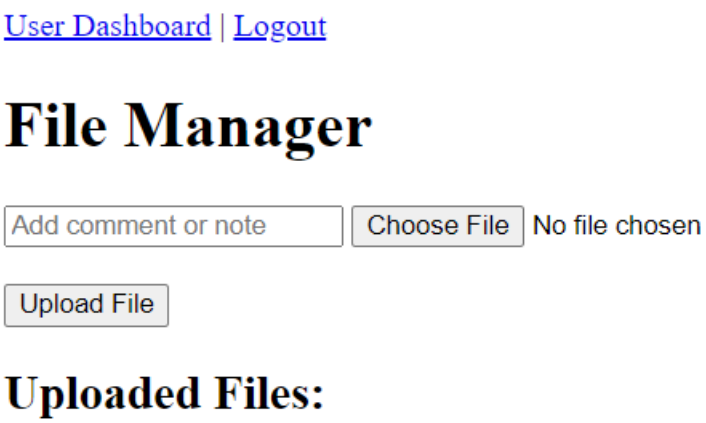
```
        $deleteQuery = "DELETE FROM files WHERE id = $fileId";
        mysqli_query($con, $deleteQuery);

        $status = "File deleted successfully.";
    }
```

Table 6

- Now, the “file_manager.php” should display the uploaded files and handle file deletion as described in the instructions.
- After completing the tasks, you should be able to upload, view, and delete the file.

Output:



Fugure 4

Example: File Update - User Input and File Re-Upload

1. Use the existing “file_manager.php” script.
2. Insert the new code block for file update/re-upload.
3. Locate the section after the file upload and before the file deletion handling code in the "file_manager.php" script.
4. Type the following code below after the file upload section and before the file deletion section:

How does it handle file re-upload process?

```
    if (isset($_POST['update'])) {
        $fileId = $_POST['file_id'];
        $userInput = $_POST['user_input'];

        if ($_FILES['new_file']['size'] > 0) {
            $newUploadedFileName = $_FILES['new_file']['name'];
            $targetDirectory = "upload/";
            $targetFilePath = $targetDirectory . $newUploadedFileName;

            if (move_uploaded_file($_FILES['new_file']['tmp_name'], $targetFilePath)) {
```

```
                $selectQuery = "SELECT filename FROM files WHERE id = $fileId";
                $result = mysqli_query($con, $selectQuery);
                $row = mysqli_fetch_assoc($result);
                $oldFilename = $row['filename'];

                $oldFilePath = "upload/" . $oldFilename;
                if (file_exists($oldFilePath) && !is_dir($oldFilePath)) {
                    unlink($oldFilePath);
                }

                $updateFileQuery = "UPDATE files SET filename = '$newUploadedFileName',
user_input = '$userInput' WHERE id = $fileId";
                mysqli_query($con, $updateFileQuery) or die(mysqli_error($con));

                $status = "File re-uploaded successfully.";
            } else {
                $status = "File re-upload failed.";
            }
        } else {
            $updateQuery = "UPDATE files SET user_input = '$userInput' WHERE id = $fileId";
            mysqli_query($con, $updateQuery) or die(mysqli_error($con));
            $status = "File details updated successfully.";
        }
    }
```

What happens to the file in database and upload folder once it is re-uploaded?

Table 7

5. Search for the section in the script where uploaded files are being displayed to the user.
6. Find the part in the script that displays the uploaded files (where a while loop iterates through the query results to show each file).
7. Replace the previous code block within this loop:

```
    if ($filesResult) {
        while ($fileRow = mysqli_fetch_assoc($filesResult)) {
        echo "<li>";
        echo "<form method='post' enctype='multipart/form-data'>";
        echo "<input type='hidden' name='file_id' value='" . $fileRow['id'] . "' />";
        echo "<input type='text' name='user_input' value='" . $fileRow['user_input'] . "' />";
        echo "<label for='reupload_file'>Re-upload File:</label>";
        echo "<input type='file' name='new_file' id='reupload_file' />";
        echo "<input type='submit' name='update' value='Update' />";
        echo "</form>";
        echo " <a href='upload/" . $fileRow['filename'] . "' target='_blank'>View</a>";
        echo " | <a href='file_manager.php?delete=" . $fileRow['id'] . "' onclick=\"return
confirm('Are you sure you want to delete this file?')\">Delete</a>";
        echo "</li>";
        }
    }
```

Table 8

8. It creates a form with:
   - Hidden input for file ID.
   - Text input for user input (with the existing value displayed).
   - Option to re-upload a file using a file input field.
   - An 'Update' button to submit changes.
   - Links to ‘View’ the file and ‘Delete’ the file.

9. Add the following code at the end of the page (before </html>) to display the status message:

```
<?php
    if (isset($status)) {
        echo "<p>Status: $status</p>";
    }
?>
```

Table 9

Output:

Figure 5

Additional Exercise: Implement File Validation for File Upload, File Update and File Deletion

Issue: Users may not be aware of the accepted file types or may mistakenly select the wrong files during the upload process. Common scenario: Students might upload files with PDF or DOC file types because they are commonly used for document/assignment submissions, but there is a risk of accidental or intentional uploading of incorrect or harmful files.

Solution: Implement a file type validation system that checks the uploaded files by specifying the lists of approved file types.

Overview Process:

- To define/specify the file types are allowed for upload, update, and deletion (in this example: pdf, doc, docx, txt).
- To implement method for file extensions validation, ensuring only allowed file types: pdf, doc, docx, txt can be uploaded.
- Implement update function: To verify new file extensions and replace existing files if needed.
- Implement deletion function: To check file existence before removal to prevent errors.

Remark:

- Now that you have successfully implemented file upload, re-upload and file deletion. You may take your learning further by exploring and modifying the code on your own (eg: to use other method for file validation (example: accept only image formats such as jpg, jpeg, png, and gif.).
- Remember to document any changes you make and understand how the changes impact your system.