RAMON MAGSAYSAY MEMORIAL COLLEGES
Information Technology Education Program
1st SEMESTER: AY: 2025 - 2026

NAME: Kent Edrick B. Felicia SCHEDULE: Fri - Sat SCORE: _____
SUBJECT: **WEB SYSTEMS AND TECHNOLOGIES** INSTRUCTOR: JIm Jamero DATE: 10/21/25

LABORATORY EXERCISE 7
FILE UPLOADS (COURSE MATERIALS)

Learning Objectives

By the end of this laboratory exercise, students should be able to:

- Design and implement a database schema for managing file uploads related to courses.
- Utilize CodeIgniter's File Uploading Library to handle file uploads securely.
- Create an administrative interface for uploading and managing course materials.
- Implement access control to ensure only enrolled students can download materials.
- Enhance the user interface with Bootstrap for a clean and functional file management system.

Prerequisite student experiences and knowledge

Before starting this exercise, students should have:

- Completed Laboratory Exercise 6 (Course Enrollment System).
- A solid understanding of CodeIgniter's MVC structure and database operations.
- Experience with HTML forms and Bootstrap styling.
- Basic knowledge of handling file inputs in HTML.
- Familiarity with user authentication and session management in CodeIgniter.

Background

A core feature of any Learning Management System (LMS) is the distribution of learning resources. This involves allowing instructors to upload various file types (such as PDFs, PowerPoint presentations, and documents) and making them available to enrolled students. CodeIgniter provides a robust File Uploading Library that simplifies the process of validating, uploading, and securing files. This exercise will integrate this functionality into the LMS, ensuring that file management is seamless and secure.

Materials/Resources

- Personal Computer with Internet Access
- XAMPP/WAMP/LAMP server installed
- CodeIgniter Framework (latest version)
- Visual Studio Code or any code editor
- Git and GitHub Account
- Web Browser (Chrome, Firefox, etc.)

Laboratory Activity

Step 1: Create a Database Migration for Materials Table

1. Create a new migration file for the materials table.
   Run: **php spark make:migration CreateMaterialsTable**
2. Open the newly created file in **app/Database/Migrations/**.
3. In the **up() method**, define the table with the following fields:
   - id (Primary Key, Auto-Increment)

- course_id (INT, Foreign Key referencing the `courses` table)
- file_name (VARCHAR(255)) to store the original filename
- file_path (VARCHAR(255)) to store the path to the uploaded file
- created_at (DATETIME)

4. In the **down() method**, define how to drop the table.
   Run the migration: php spark migrate

Step 2: Create a Model for Materials

1. Navigate to app/Models/ and create a file named **MaterialModel.php**.
2. Create a model class with methods to:
   - **insertMaterial($data)** Insert a new material record.
   - **getMaterialsByCourse($course_id)** Get all materials for a specific course.

Step 3: Create or Modify a Controller for Materials

1. You can create a new Materials.php controller or add methods to an existing admin controller.
2. Add the following methods:
   - **upload($course_id)** Displays the file upload form and handles the file upload process.
   - **delete($material_id)** Handles the deletion of a material record and the associated file.
   - **download($material_id)** Handles the file download for enrolled students.

Step 4: Implement File Upload Functionality

1. In the **upload($course_id**) method, check for a POST request.
2. Load CodeIgniter's File Uploading Library and Validation Library.
3. Configure the upload preferences (upload path, allowed file types, maximum file size).
4. Perform the file upload. If successful, prepare data (course_id, file_name, file_path) and save it to the database using the **MaterialModel**.
5. Set a flash message indicating success or failure and redirect back to the course management page.

Step 5: Create the File Upload View

1. Create a view file.
2. The view should contain a form with the **enctype="multipart/form-data"** attribute and a file input field.
3. Style the form using Bootstrap classes.

Step 6: Display Downloadable Materials for Students

1. In the student dashboard or a dedicated course view, use the **MaterialModel** to fetch materials for the courses the student is enrolled in.
2. Create a view that lists these materials, displaying the file name and a download button/link for each.
3. The download link should point to the **download($material_id)** method in your controller.

NAME: Kent Edrick B. Felicia SCHEDULE: Fri - Sat SCORE: _____
SUBJECT: **WEB SYSTEMS AND TECHNOLOGIES** INSTRUCTOR: JIm Jamero DATE: 10/21/25

Step 7: Implement the Download Method

1. In the **download($material_id)** method:
   - ✓ Check if the current user is logged in and enrolled in the course associated with the material.
   - ✓ Retrieve the file path from the database using the `$material_id`.
   - ✓ Use CodeIgniter's download() helper function or the Response class to force the file download securely.

Step 8: Update Routes

1. Update app/Config/Routes.php to include routes for the new functionalities.
   $routes->get('/admin/course/(:num)/upload', 'Materials::upload/$1');
   $routes->post('/admin/course/(:num)/upload', 'Materials::upload/$1');
   $routes->get('/materials/delete/(:num)', 'Materials::delete/$1');
   $routes->get('/materials/download/(:num)', 'Materials::download/$1');

Step 9: Test the Application

1. Run the application and test the complete flow:
   - ✓ Log in as an admin/instructor.
   - ✓ Navigate to a course and upload a file (PDF, PPT).
   - ✓ Verify the file is saved in the designated folder and a record is added to the `materials` table.
   - ✓ Log in as a student enrolled in the course.
   - ✓ Navigate to the course page and verify the material is listed.
   - ✓ Test the download functionality.
   - ✓ Test accessing the download link while not enrolled (it should be restricted).

Step 9: Push to GitHub

1. Add, commit, and push your changes to your GitHub repository.

Output / Results

- ✓ Screenshot of the **materials** table schema from your database (phpMyAdmin or equivalent).
- ✓ Screenshots of the file upload form (admin side) and the student view showing the list of downloadable materials.
- ✓ Screenshot of the server's file system (upload directory) showing the uploaded file.
- ✓ A screenshot of the GitHub repository with the latest commit for this exercise.

RAMON MAGSAYSAY MEMORIAL COLLEGES
Information Technology Education Program
1st SEMESTER: AY: 2025 - 2026

NAME: Kent Edrick B. Felicia SCHEDULE: Fri - Sat SCORE: _____
SUBJECT: **WEB SYSTEMS AND TECHNOLOGIES** INSTRUCTOR: JIm Jamero DATE: 10/21/25

QUESTIONS:

1. What are the security risks associated with file uploads, and how did you mitigate them using CodeIgniter's File Uploading Library?
2. Explain the purpose of the **enctype="multipart/form-data"** attribute in the form tag for file uploads.
3. Why is it important to check if a student is enrolled in a course before allowing them to download a material? How does this enforce application security?.

RAMON MAGSAYSAY MEMORIAL COLLEGES
Information Technology Education Program
1st SEMESTER: AY: 2025 - 2026

NAME: Kent Edrick B. Felicia SCHEDULE: Fri - Sat SCORE: _____
SUBJECT: **WEB SYSTEMS AND TECHNOLOGIES** INSTRUCTOR: JIm Jamero DATE: 10/21/25

Output / Results

RAMON MAGSAYSAY MEMORIAL COLLEGES
Information Technology Education Program
1st SEMESTER: AY: 2025 - 2026

NAME: Kent Edrick B. Felicia SCHEDULE: Fri - Sat SCORE: _____
SUBJECT: **WEB SYSTEMS AND TECHNOLOGIES** INSTRUCTOR: JIm Jamero DATE: 10/21/25

Conclusion