



## RAMON MAGSAYSAY MEMORIAL COLLEGES

Information Technology Education Program

1<sup>st</sup> SEMESTER: AY: 2025 - 2026



NAME: Marjovic P. Alejado

Schedule: 7:00AM-10:00AM

Score: \_\_\_\_\_

SUBJECT: **WEB SYSTEMS AND TECHNOLOGIES** INSTRUCTOR: Jim S. Jamero

DATE: 10-7-2025

### LABORATORY EXERCISE 6 COURSE ENROLLMENT SYSTEM

#### Learning Objectives

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

- Design and create a new database table to manage relationships between users and courses.
- Implement server-side logic for handling course enrollments.
- Display user-specific data (enrolled courses) in a dashboard.
- Utilize jQuery and AJAX to create a dynamic, seamless user experience without page reloads.
- Understand and implement basic foreign key relationships in a web application.

#### Prerequisite student experiences and knowledge

Before starting this exercise, students should have:

- ❖ Completed Laboratory Exercise 5 (Admin and Student Dashboards).
- ❖ A solid understanding of the MVC architecture in CodeIgniter.
- ❖ Proficiency in writing database queries using CodeIgniter's Query Builder.
- ❖ Basic knowledge of SQL relationships (one-to-many).
- ❖ Familiarity with jQuery syntax and the concept of AJAX.
- ❖ Ability to create and style front-end components with Bootstrap.

#### Background

A core feature of any Learning Management System (LMS) is the ability for students to enroll in available courses. This involves creating a relationship between the **users** table (students) and the **courses** table. This relationship is typically stored in a pivot table. To enhance user experience, the enrollment process should be dynamic, allowing students to join courses without refreshing the page. This is achieved using jQuery AJAX to send a request to the server in the background, providing immediate feedback to the user.

#### 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 the Enrollments Table

1. Create a new migration file for the **enrollments** table.  
Run: php spark make:migration CreateEnrollmentsTable



## RAMON MAGSAYSAY MEMORIAL COLLEGES

Information Technology Education Program

1<sup>st</sup> SEMESTER: AY: 2025 - 2026



NAME: Marjovic P. Alejado

Schedule: 7:00AM-10:00AM

Score: \_\_\_\_\_

SUBJECT: **WEB SYSTEMS AND TECHNOLOGIES** INSTRUCTOR: Jim S. Jamero

DATE: 10-7-2025

---

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)
  - ✓ user\_id (int, foreign key to **users** table)
  - ✓ course\_id (int, foreign key to **courses** table)
  - ✓ enrollment\_date (datetime)
4. In the down() method, define how to drop the table.
5. Run the migration: php spark migrate.

### Step 2: Create the Enrollment Model

1. Navigate to app/Models/ and create a file named EnrollmentModel.php.
2. Create a model class with methods to:
  - ✓ enrollUser(\$data): Insert a new enrollment record.
  - ✓ getUserEnrollments(\$user\_id): Fetch all courses a user is enrolled in.
  - ✓ isAlreadyEnrolled(\$user\_id, \$course\_id): Check if a user is already enrolled in a specific course to prevent duplicates.

### Step 3: Modify the Course Controller

1. Open your Course.php controller (or create it if it doesn't exist).
2. Add a new method, enroll(), to handle the AJAX request.
  - ✓ This method should:
  - ✓ Check if the user is logged in.
  - ✓ Receive the **course\_id** from the POST request.
  - ✓ Check if the user is already enrolled.
  - ✓ If not, insert the new enrollment record with the current timestamp.
  - ✓ Return a JSON response indicating success or failure.

### Step 4: Update Student Dashboard View

1. Open/Check the student dashboard view file.
2. Create a section to **Display Enrolled Courses**. Use a Bootstrap list group or cards to iterate over and display the courses returned by **EnrollmentModel::getUserEnrollments()**.
3. Create another section for **Available Courses**. Display a list of courses with an **Enroll** button next to each.

### Step 5: Implement AJAX Enrollment

1. In the **Available Courses** section of the dashboard, add a **data\_course\_id** attribute to each **Enroll** button containing the specific course ID.
2. Include the jQuery library in your view if it's not already included.
3. Write a jQuery script that:
  - ✓ Listens for a click on the **Enroll** button.
  - ✓ Prevents the default form submission behavior.



## RAMON MAGSAYSAY MEMORIAL COLLEGES

Information Technology Education Program

1<sup>st</sup> SEMESTER: AY: 2025 - 2026



NAME: Marjovic P. Alejado

Schedule: 7:00AM-10:00AM

Score: \_\_\_\_\_

SUBJECT: **WEB SYSTEMS AND TECHNOLOGIES** INSTRUCTOR: Jim S. Jamero

DATE: 10-7-2025

- ✓ Uses `$.post()` to send the `course_id` to the `/course/enroll` URL.
- ✓ On a successful response from the server:
- ✓ Displays a Bootstrap alert message.
- ✓ Hides or disables the **Enroll** button for that course.
- ✓ Updates the **Enrolled Courses** list dynamically without reloading the page.

### Step 6: Configure Routes

1. Update app/Config/Routes.php to include a route for the enrollment action.  
`$routes->post('/course/enroll', 'Course::enroll');`

### Step 7: Test the Application Thoroughly

1. Log in as a student.
2. Navigate to the student dashboard.
3. Click the **Enroll** button on an available course and verify:
  - The page does not reload.
  - A success message appears.
  - The button becomes disabled or disappears.
  - The course appears in the **Enrolled Courses** list.

### Step 8: Push to GitHub

1. Commit your changes with a descriptive message.
2. Push your changes to your GitHub repository.

### Step 9: Vulnerable Checking

1. Test for Authorization Bypass
  - ❖ Log out of the application and attempt to directly access the enrollment endpoint via Postman or browser console by sending a POST request to `/course/enroll` with a `course_id` parameter.
  - ❖ Verify that the server returns an unauthorized error instead of processing the enrollment.
2. Test for SQL Injection
  - ❖ While logged in, use browser developer tools to modify the AJAX request and change the `course_id` value to `1 OR 1=1`.
  - ❖ Check if the application properly validates the input and prevents SQL injection attacks.
3. Test for CSRF (Cross-Site Request Forgery)
  - ❖ Check if your enrollment form includes CSRF protection tokens.
  - ❖ Verify that CodeIgniter's CSRF protection is enabled in app/Config/Security.php.
  - ❖ Attempt to make an enrollment request without a valid CSRF token and confirm it is rejected.
4. Test for Data Tampering



## RAMON MAGSAYSAY MEMORIAL COLLEGES

Information Technology Education Program

1<sup>st</sup> SEMESTER: AY: 2025 - 2026



NAME: Marjovic P. Alejado

Schedule: 7:00AM-10:00AM

Score: \_\_\_\_\_

SUBJECT: **WEB SYSTEMS AND TECHNOLOGIES**

INSTRUCTOR: Jim S. Jamero

DATE: 10-7-2025

- ❖ As a student, try to enroll another user in a course by modifying the user ID in the request.
- ❖ Verify that the server-side code uses the logged-in user's session ID rather than trusting client-supplied user IDs.

### 5. Test for Input Validation

- ❖ Attempt to enroll in non-existent courses by sending invalid course\_id values.
- ❖ Verify that the application properly validates that the course exists before creating an enrollment.

### Output / Results

- ✓ Screenshot of your database's **enrollments** table structure (phpMyAdmin or equivalent).
- ✓ A screenshot of the student dashboard showing the **Available** and **Enrolled Courses** sections is attached.
- ✓ A screenshot of the browser's developer tools (Network tab) shows the successful AJAX POST request and response when enrolling in a course.
- ✓ A screenshot of the GitHub repository with the latest commit for this exercise.



## RAMON MAGSAYSAY MEMORIAL COLLEGES

Information Technology Education Program

1<sup>st</sup> SEMESTER: AY: 2025 - 2026



NAME: Marjovic P. Alejado

Schedule: 7:00AM-10:00AM

Score: \_\_\_\_\_

SUBJECT: **WEB SYSTEMS AND TECHNOLOGIES** INSTRUCTOR: Jim S. Jamero

DATE: 10-7-2025

### QUESTIONS:

1. What is the purpose of the **enrollments** table? Why is it necessary, instead of just adding a **course\_id** column to the **users** table?

The purpose of the enrollments table is to connect students with the courses they are taking by storing the relationship between users and courses. It is necessary because one student can take many courses, and one course can have many students enrolled in it.

2. Explain the role of the **isAlreadyEnrolled()** method in the Model. What potential issue does it prevent?

The role of the **isAlreadyEnrolled()** method in the Model is by checking if a student is already enrolled in a specific course before allowing a new enrollment. Potential issue it does prevent is duplicate enrollments because it stops a student from being enrolled in the same course multiple times by mistake.

3. Describe the client-side and server-side steps when students click the **Enroll** button until they receive confirmation.

### Steps in Client Side Scripting

When I click the **Enroll** button, the JavaScript immediately disables the button, shows a loading spinner, and prepares my enrollment request with the **course\_id** and **CSRF token** from the page. The browser then sends a **POST** request to the **/course/enroll** endpoint with headers including **X-Requested-With: XMLHttpRequest** and **X-CSRF-TOKEN** to verify it's AJAX requested by me/the user.

### Steps in Server Side Scripting

The server receives my POST request at the **/course/enroll endpoint** and checks if I'm logged in as a student while validating all the security tokens and headers I sent. If everything is correct and I'm not already enrolled, the server adds my enrollment to the database and sends back a JSON response with my enrollment details.



## RAMON MAGSAYSAY MEMORIAL COLLEGES

Information Technology Education Program

1<sup>st</sup> SEMESTER: AY: 2025 - 2026



## Output/Results

### phpMyAdmin Enrollments Table

Screenshot of the phpMyAdmin interface showing the 'enrollments' table from the 'lms\_alejado' database.

The table structure is as follows:

	id	user_id	course_id	enrollment_date
<input type="checkbox"/> <a href="#">Edit</a> <a href="#">Copy</a> <a href="#">Delete</a>	1	4	1	2025-10-07 13:48:58
<input type="checkbox"/> <a href="#">Edit</a> <a href="#">Copy</a> <a href="#">Delete</a>	2	4	2	2025-10-07 13:50:08
<input type="checkbox"/> <a href="#">Edit</a> <a href="#">Copy</a> <a href="#">Delete</a>	3	4	3	2025-10-07 13:50:11
<input type="checkbox"/> <a href="#">Edit</a> <a href="#">Copy</a> <a href="#">Delete</a>	4	5	1	2025-10-07 13:53:30
<input type="checkbox"/> <a href="#">Edit</a> <a href="#">Copy</a> <a href="#">Delete</a>	5	5	2	2025-10-07 14:00:13
<input type="checkbox"/> <a href="#">Edit</a> <a href="#">Copy</a> <a href="#">Delete</a>	6	5	3	2025-10-07 14:00:21
<input type="checkbox"/> <a href="#">Edit</a> <a href="#">Copy</a> <a href="#">Delete</a>	7	6	1	2025-10-07 15:02:51
<input type="checkbox"/> <a href="#">Edit</a> <a href="#">Copy</a> <a href="#">Delete</a>	8	6	2	2025-10-07 15:05:59
<input type="checkbox"/> <a href="#">Edit</a> <a href="#">Copy</a> <a href="#">Delete</a>	9	6	3	2025-10-07 15:14:02
<input type="checkbox"/> <a href="#">Edit</a> <a href="#">Copy</a> <a href="#">Delete</a>	10	7	1	2025-10-07 15:19:50
<input type="checkbox"/> <a href="#">Edit</a> <a href="#">Copy</a> <a href="#">Delete</a>	11	7	3	2025-10-07 23:28:51

Query results operations:  Console



## RAMON MAGSAYSAY MEMORIAL COLLEGES

Information Technology Education Program

1<sup>st</sup> SEMESTER: AY: 2025 - 2026



### Student Dashboard

Screenshot of the MGOD LMS Student Dashboard.

The dashboard features a top navigation bar with tabs: MGOD LMS (selected), Student, Dashboard, My Courses, Assignments, Grades, and Schedule. A user profile for Marjovic Alejado is shown on the right.

#### Student Dashboard Summary

Welcome back, Marjovic Alejado! Continue your learning journey and achieve your goals.

**Enrolled Courses:** 2 Active learning paths

**Completed Assignments:** 0

**Pending Assignments:** 0 Awaiting completion

**Average Grade:** 0% Overall performance

#### My Enrolled Courses

Continue your learning journey

Course Name	Status	Progress	Enrollment Date	Action
Database Design	Enrolled	0%	Oct 7, 2025	Continue
Introduction to Programming	Enrolled	0%	Oct 7, 2025	Continue

#### Available Courses

Discover new learning opportunities

Course Name	Status	Credits	Duration	Students	Action
Web Development Basics	Active	4	12w	25	Enroll in Course

#### Upcoming Deadlines

Don't miss these important dates

Assignment deadlines and important dates will appear here.

#### Recent Grades & Feedback

Your latest academic performance

Your grades and teacher feedback will appear here.

System status bar at the bottom shows: 12:03 AM, 10/8/2025.



## RAMON MAGSAYSAY MEMORIAL COLLEGES

Information Technology Education Program

1<sup>st</sup> SEMESTER: AY: 2025 - 2026



### Browser's Developer Tools AJAX POST

The screenshot shows a web browser window with several tabs open. The active tab is 'localhost/ITE311-ALEJADO/dashboard'. A modal dialog box is displayed in the center, titled 'Enrollment Successful!' with a green checkmark icon. The main content area of the page displays course information and a message: 'Welcome to Database Design! You have been successfully enrolled in this course. You can now access course materials and start learning.' Below this, a success message says 'Enrollment Date: Oct 7, 2025 11:28PM' and a 'Refresh Page' button. To the right, the browser's developer tools Network tab is open, showing a list of requests. The first request, 'enroll', is selected and expanded, showing details like Request URL (http://localhost/ITE311-ALEJADO/course/enroll), Request Method (POST), Status Code (201 Created), and Headers. The Headers section includes Cache-Control, Connection, Content-Length, Content-Type, Date, Expires, Keep-Alive, Pragma, Server, Set-Cookie, and X-Powered-By. The Headers table has columns for Name, Headers, Payload, Preview, Response, Initiator, Timing, and Cookies. The Headers row for 'enroll' is expanded, showing raw header values. The developer tools also show 2 requests in total.

Name	Headers	Payload	Preview	Response	Initiator	Timing	Cookies
enroll	Request URL: http://localhost/ITE311-ALEJADO/course/enroll Request Method: POST Status Code: 201 Created Remote Address: [::1]:80 Referrer Policy: strict-origin-when-cross-origin						

Request Headers:

Name	Value
Accept	/*
Accept-Encoding	gzip, deflate, br, zstd



# RAMON MAGSAYSAY MEMORIAL COLLEGES

Information Technology Education Program

1<sup>st</sup> SEMESTER: AY: 2025 - 2026



GitHub Link: <https://github.com/Marjovic/WebSystem-ITE311>

Marjovic / WebSystem-ITE311

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

WebSystem-ITE311 Public

main 1 Branch 0 Tags

Go to file Add file Code

Marjovic Add CSRF tokens to the dashboard for security purposes 5799a1d · 1 minute ago 28 Commits

app Add CSRF tokens to the dashboard for security purposes 1 minute ago

public Refactor baseURL and uriProtocol in App configuration: last month

system Initial commit - CodeIgniter setup 2 months ago

tests Initial commit - CodeIgniter setup 2 months ago

writable Add CSRF tokens to the dashboard for security purposes 1 minute ago

.htaccess Improves Vulnerability and added Manage Users Functionality 5 hours ago

LICENSE Initial commit - CodeIgniter setup 2 months ago

README.md Initial commit - CodeIgniter setup 2 months ago

composer.json Initial commit - CodeIgniter setup 2 months ago

env Initial commit - CodeIgniter setup 2 months ago

index.php Refactor baseURL and uriProtocol in App configuration: last month

phpunit.xml.dist Initial commit - CodeIgniter setup 2 months ago

About Web System Development

Readme MIT license Activity 0 stars 0 watching 0 forks

Releases No releases published Create a new release

Packages No packages published Publish your first package

Languages PHP 98.0% JavaScript 1.1%

Commits

All users All time

Commits on Oct 8, 2025

Add CSRF tokens to the dashboard for security purposes Marjovic committed 1 minute ago 5799a1d

Commits on Oct 7, 2025

Enhance student dashboard to display enrolled and available courses with progress tracking Marjovic committed 29 minutes ago 76888d9

Add Course controller and seeder for course enrollment functionality Marjovic committed 30 minutes ago d6ebbcc

Add EnrollmentModel for managing user enrollments and course registrations Marjovic committed 30 minutes ago c146dc4

Enhance enrollment and submission tables to replace 'student\_id' with 'user\_id' Marjovic committed 31 minutes ago eb4a2af

Refactor dashboard view and auth controller for role based implementation Marjovic committed 4 hours ago 7af0fd3d

Improves Vulnerability and added Manage Users Functionality Marjovic committed 5 hours ago 3f14e27



# RAMON MAGSAYSAY MEMORIAL COLLEGES

Information Technology Education Program

1<sup>st</sup> SEMESTER: AY: 2025 - 2026



## Vulnerability Checking

The screenshot shows the Postman interface for testing an API endpoint. The URL is `http://localhost:1TE311-ALEJADO/login`. The method is POST. The body is set to x-www-form-urlencoded with two parameters: email (maryjoy.alejado@student.lms.com) and password (student123). The response status is 200 OK, indicating a successful login. Below the request, a preview of the MGOD LMS Student Dashboard is shown, displaying a welcome message and four icons.

The screenshot shows the Postman interface for testing an API endpoint. The URL is `http://localhost:1TE311-ALEJADO/course/enroll`. The method is POST. The body is set to x-www-form-urlencoded with two parameters: course\_id (1) and csrf\_token\_name (c413d2fe59979b582063949b8aaabdc1). The response status is 409 Conflict, indicating that the user is already enrolled in the course. The response body is a JSON object:

```
1 {  
2   "success": false,  
3   "message": "You are already enrolled in this course.",  
4   "error_code": "ALREADY_ENROLLED",  
5   "csrf_hash": "c413d2fe59979b582063949b8aaabdc1"  
6 }
```



## RAMON MAGSAYSAY MEMORIAL COLLEGES

Information Technology Education Program

1<sup>st</sup> SEMESTER: AY: 2025 - 2026



### Conclusion

In conclusion, thanks to this Laboratory Exercise 5, I learned how to create course enrollment system by designing database tables with proper relationships and implementing secure server-side logic to handle student enrollments.

In addition, I also learned how to use jQuery and AJAX to create a interactive user experience where students can enroll in courses without refreshing the page.