

Information Technology Education Program 1st SEMESTER: AY: 2025 - 2026



NAME: _	SCHEDULE:		SCORE:
SUBJECT	WEB SYSTEMS AND TECHNOLOGIES	INSTRUCTOR:	DATE:
LADODATORY EVERGES 0			

LABORATORY EXERCISE 8 REAL-TIME NOTIFICATIONS WITH JQUERY

Learning Objectives

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

- Implement AJAX functionality using jQuery to fetch data from the server without refreshing the page.
- ☐ Create a dynamic notification system that displays real-time updates to the user.
- Update the user interface (UI) based on server-side data, specifically by managing a notification badge.
- Utilize Bootstrap components for styling interactive alerts and badges.
- ☐ Manage application state by marking notifications as "read" via an AJAX call.

Prerequisite student experiences and knowledge

Before starting this exercise, students should have:

- Completed Laboratory Exercise 7 (File Uploads for Course Materials).
- A solid understanding of the Codelgniter MVC structure and database operations.
- Proficiency in writing basic jQuery and JavaScript code.
- Experience with handling jQuery AJAX requests (GET, POST).
- Familiarity with manipulating the DOM with jQuery (e.g., showing/hiding elements, updating text).
- Knowledge of Bootstrap classes for badges and alerts.

100 01000011

Background

A key feature of modern, interactive web applications is the ability to provide real-time feedback and updates to users. Notifications inform users of important events, such as new course enrollments or available materials, without requiring a page reload. jQuery's AJAX methods allow the client-side browser to asynchronously communicate with the server, fetching new data in the background. This data can then be dynamically inserted into the webpage, creating a seamless user experience. This exercise will guide you in building a notification system that displays a badge count in the navigation bar and a dropdown list of alerts, all styled with Bootstrap.

Materials/Resources

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

Information Technology Education Program

1st SEMESTER: AY: 2025 - 2026



NAME: _	SCHEDULE:	SCORE:
SUBJECT:	WEB SYSTEMS AND TECHNOLOGIES INSTRUCTOR	R: DATE:

Laboratory Activity

Step 1: Database Setup for Notifications

1. Create a new migration file for a notifications table.

Run: php spark make: migration CreateNotificationsTable

- 2. Open the new migration file in app/Database/Migrations/.
 - In the up() method, define the table with the following fields:
 - id (primary key, auto-increment)
 - user id (int, foreign key to users table)
 - message (varchar, e.g., "You have been enrolled in [Course Name]")
 - is_read (tinyint, default 0)
 - created_at (datetime)
- 3. In the down() method, drop the notifications table.
- 4. Run the migration: php spark migrate

Step 2: Create a Notification Model

- Navigate to app/Models/ and create a file named Notification Model. php.
- Create methods for:
 - ✓ getUnreadCount(\$userId)
 - Fetches the count of unread notifications for a user.
 - ✓ getNotificationsForUser(\$userId)
 - Fetches the latest notifications (e.g., limit 5) for a user.
 - markAsRead(\$notificationId)
 - Updates a specific notification's is_read field to 1.

Step 3: Update the Base Controller/Layout

- 1. To display the notification badge on all pages, we need to fetch the unread count for the logged-in user and make it available to the main layout.
- 2. In your base controller (or a custom controller that others extend), add logic to load the unread notification count and pass it to the view. Alternatively, you can create a view fragment that uses an AJAX call to get the count (more complex but more efficient).
- 3. For simplicity, modify your main layout file (e.g., app/Views/templates/header.php) to include a placeholder for the notification badge..

Step 4: Create a Notifications Controller and API Endpoints

- 1. Create a controller named Notifications.php in app/Controllers/.
- 2. Add the following methods:
 - ✓ get()
- o A method that returns a JSON response containing the current user's unread notification count and list of notifications. This will be called via AJAX.
- ✓ mark as read(\$id)
 - o A method that accepts a notification ID via POST and marks it as read. Returns a success/failure JSON response.

Information Technology Education Program
1st SEMESTER: AY: 2025 - 2026



NAME: _	SCHEDULE:	SCORE:
SUBJECT:	WEB SYSTEMS AND TECHNOLOGIES INSTRUCTOR	R: DATE:

- 3. Ensure these routes are added to app/Config/Routes.php:
 - \\$routes-\>get('/notifications', 'Notifications::get');
 - \\$routes-\>post('/notifications/mark_read/(:num)', 'Notifications::mark_as_read/\$1');

Step 5: Build the Notification UI with jQuery and Bootstrap

- 1. In your main layout file (e.g., header.php), add the Bootstrap-styled notification dropdown to the navigation bar.
- 2. Include a badge ('...') to show the unread count. Initially, it can be hidden or show 0.
- 3. Create the dropdown menu structure to list notifications. It can initially be empty.
- 4. Write a jQuery function (in a separate .js file or within a `<script>` tag) that uses `\$.get()` to call your /notifications endpoint.
- 5. In the AJAX success callback, update the badge count with the returned data. If the count is 0, hide the badge; otherwise, show it.
- 6. Populate the dropdown menu with the list of notifications. Use Bootstrap's alert classes (e.g., `alert alert-info`) for each notification item to improve styling.
- 7. For each notification, add a Mark as Read button/link that triggers another jQuery function.
 - This function should use \$.post() to call the /notifications/mark_read/[id] endpoint and, upon success, remove the notification from the list and update the badge count.

Step 6: Trigger Notification Updates

- 1. Call your jQuery notification-fetching function when the page loads (`\$(document).ready()`).
- 2. To simulate real-time updates, you can set an interval to fetch notifications every 60 seconds (optional advanced task).

Step 7: Generate Test Notifications

1. Temporarily modify your course enrollment logic (from a previous lab) to create a new notification in the **notifications** table for the student when they enroll in a course.

Step 8: Test the Functionality

- 1. Log in as a student and enroll in a new course (or create a notification manually in the database).
- 2. Refresh the page and verify that the notification badge appears with the correct count.
- 3. Click the notification dropdown and verify the list is populated correctly.
- 4. Click the **Mark as Read** button on a notification and verify that it disappears from the list and the badge count decreases.

Step 9: Push to GitHub

1. Commit and push your completed notification system code to your GitHub repository.

Output / Results



Information Technology Education Program 1st SEMESTER: AY: 2025 - 2026



NAME:	SCHEDULE:	SCORE: _	
SUBJECT: \	WEB SYSTEMS AND TECHNOLOGIES INSTRUCTOR	: DATE:	

- ✓ Screenshot of the `notifications` table schema from your database (phpMyAdmin or equivalent).
- ✓ Screenshot of the browser's Developer Tools "Network" tab showing the successful AJAX call to the `/notifications` endpoint and its JSON response.
- ✓ Screenshots of the navigation bar:
- ✓ With the notification badge visible (showing a count > 0).
- With the dropdown open, showing the list of notifications styled with Bootstrap alerts.
- ✓ After marking a notification as read, showing the updated badge and list.





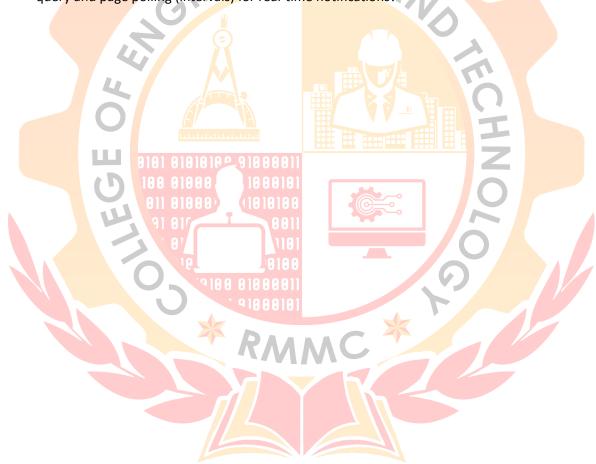
Information Technology Education Program 1st SEMESTER: AY: 2025 - 2026



NAME:	SCHEDULE:	SCORE: _	
SUBJECT: \	WEB SYSTEMS AND TECHNOLOGIES INSTRUCTOR	: DATE:	

QUESTIONS:

- 1. What are the benefits of using AJAX to load notifications compared to loading them directly with the initial page load in PHP?
- 2. Explain the role of the JSON format in the communication between your jQuery code and the Codelgniter controller.
- 3. In a production environment, what are more scalable alternatives to using a simple database query and page polling (intervals) for real-time notifications?



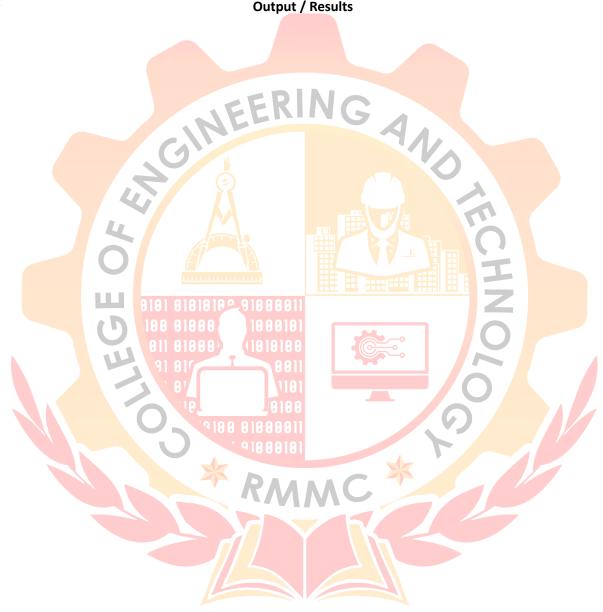


Information Technology Education Program 1st SEMESTER: AY: 2025 - 2026



NAME:	SCHEDULE:	SCORE: _	
SUBJECT: \	WEB SYSTEMS AND TECHNOLOGIES INSTRUCTOR	: DATE:	

Output / Results





Information Technology Education Program 1st SEMESTER: AY: 2025 - 2026



NAME:	SCHEDULE:	SCORE:
SUBJECT: WEB SYSTEMS AND TECHNO	LOGIES INSTRUCTOR:	DATE:

Conclusion GINEERING



Information Technology Education Program 1st SEMESTER: AY: 2025 - 2026



NAME:	SCHE	SCHEDULE:		
SUBJECT	: WEB SYSTEMS AND TECHNOLOGIES	INSTRUCTOR:	DATE:	

