Student Event Management System: Project Report (IT1208 CA-2)

Candidate: [G.D.S.H.Chandawansha / 23IT0463]

Module: IT1208 - Web Technologies

Submission Date: [2025/11/11]

1. System Overview and Architecture

The UoM Student Event Hub is a dynamic, full-stack web application designed to manage event listings and student registrations. The system is built upon a standard LAMP stack architecture (Linux/Windows, Apache, **MySQL**, and **PHP**), utilizing **HTML, CSS, and JavaScript** for the client-side experience.

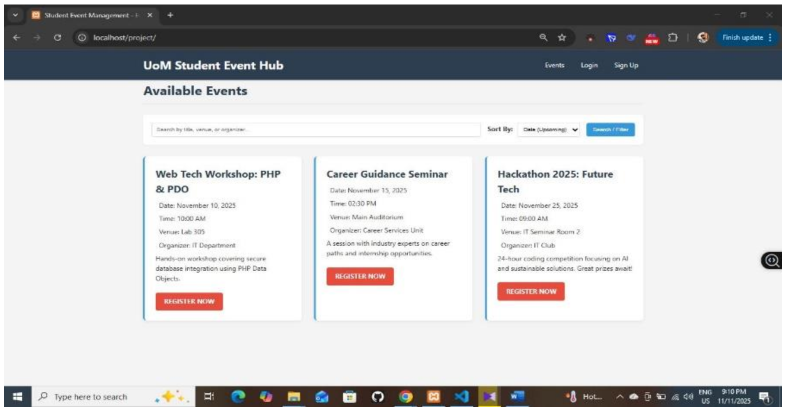
1.1 Key Features

The application successfully implemented all core requirements and two optional features:

- **Secure Authentication:** User Login and Registration using PHP sessions and secure password hashing.
- **Event CRUD:** Full Administrator control (Create, Read, Update, Delete) over event data.
- **Registration:** Secure student registration with client-side validation.
- **Responsiveness:** A fully responsive interface using modern CSS principles (`assets/style.css`).
- **Search and Filter (Optional):** Dynamic server-side filtering of events on the homepage.
- **Analytics Dashboard (Optional):** Displays key statistics in the Admin panel.

1.2 Screenshots (Placeholders)

Figure 1.1: Event Listing Homepage (index.php)



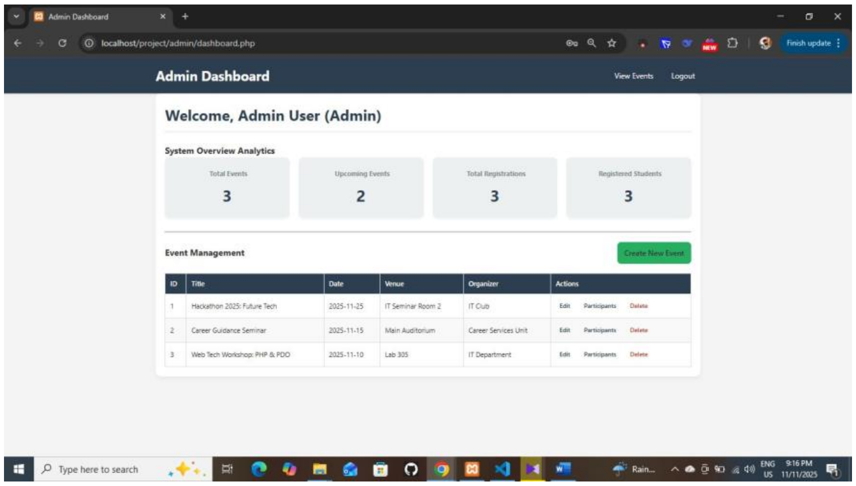
Available Events

Web Tech Workshop: PHP & PDO
Date: November 10, 2025
Time: 10:00 AM
Venue: Lab 305
Organizer: IT Department
Hands-on workshop covering secure database integration using PHP Data Objects.

Career Guidance Seminar
Date: November 15, 2025
Time: 02:30 PM
Venue: Main Auditorium
Organizer: Career Services Unit
A session with industry experts on career paths and internship opportunities.

Hackathon 2025: Future Tech
Date: November 25, 2025
Time: 09:00 AM
Venue: IT Seminar Room 2
Organizer: IT Club
24-hour coding competition focusing on AI and sustainable solutions. Great prizes await!

- *[I***Figure 1.2: Administrator Dashboard (admin/dashboard.php)**



Admin Dashboard

localhost/project/admin/dashboard.php

Admin Dashboard View Events Logout

Welcome, Admin User (Admin)

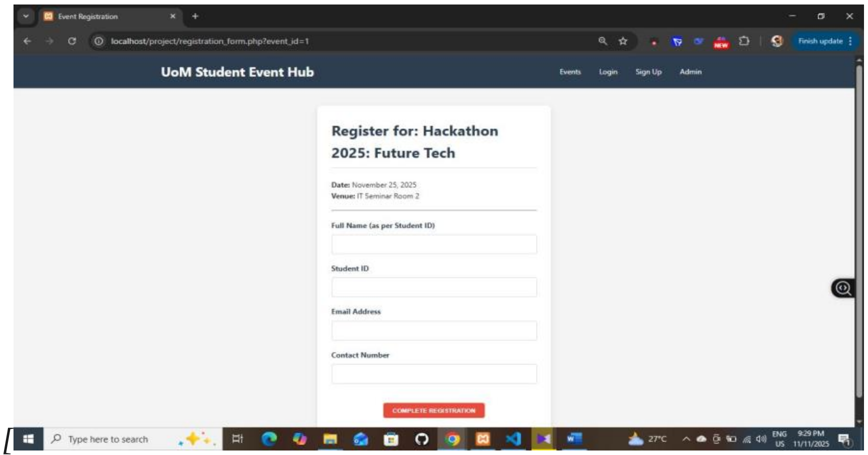
System Overview Analytics

Total Events	Upcoming Events	Total Registrations	Registered Students
3	2	3	3

Event Management Create New Event

ID	Title	Date	Venue	Organizer	Actions
1	Hackathon 2025: Future Tech	2025-11-25	IT Seminar Room 2	IT Club	Edit Participants Delete
2	Career Guidance Seminar	2025-11-15	Main Auditorium	Career Services Unit	Edit Participants Delete
3	Web Tech Workshop: PHP & PDO	2025-11-10	Lab 305	IT Department	Edit Participants Delete

- **Figure 1.3: Registration Form with Client-Side Validation**

[

Event Registration

localhost/project/registration_form.php?event_id=1

UoM Student Event Hub Events Login Sign Up Admin

Register for: Hackathon 2025: Future Tech

Date: November 25, 2025
Venue: IT Seminar Room 2

Full Name (as per Student ID)

Student ID

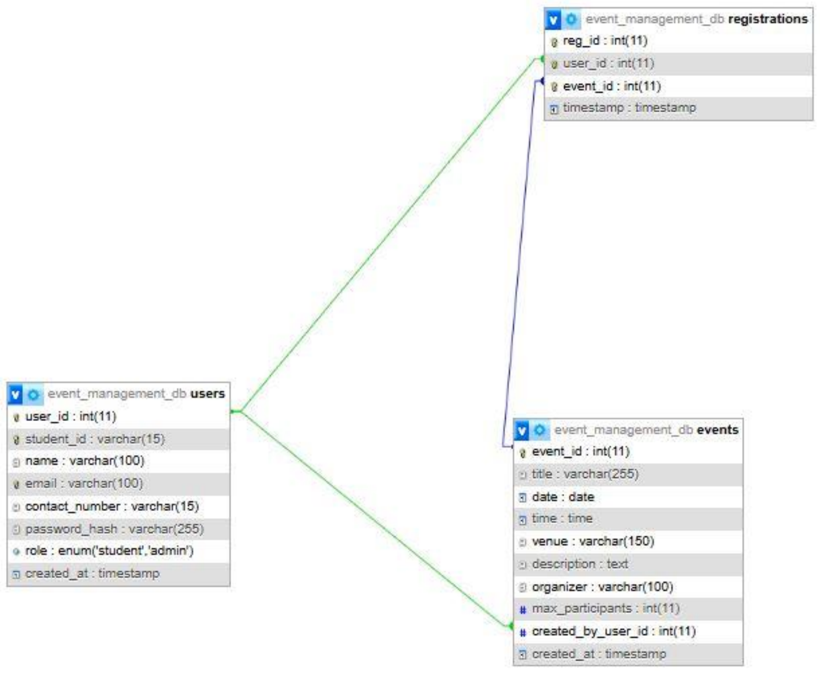
Email Address

Contact Number

COMPLETE REGISTRATION

2. Database Schema Diagram

The database, `event_management_db`, consists of three relational tables (`users`, `events`, and `registrations`) to maintain data integrity and support the many-to-many relationship between students and events.



2.1 Table Structure and Relationships

Table	**Purpose**	**Primary Key (PK)**	**Foreign Keys (FK)**	**Relationship Logic**
users	Stores student and admin credentials.	`user_id`	N/A	Supports two user roles: `student` and `admin`.
events	Stores all event details.	`event_id`	`created_by_user_id` (FK to `users`)	Tracks which administrator created the event.
registrations	Links users to events.	`reg_id` (Composite Unique Key: `user_id`, `event_id`)	`user_id` (FK to `users`), `event_id` (FK to `events`)	Enforces that one user can register for one event only once. Uses **ON DELETE CASCADE** to

				clean up registrations if an event is deleted.

3. Explanation of Core Functionalities

This section details how the Intended Learning Outcomes (ILOs) were addressed through code implementation.

3.1 Server-Side Logic and Security (PHP/ILO 3)

All critical operations are managed by PHP scripts using the **PDO (PHP Data Objects)** extension.

- **Secure Authentication:** User passwords are not stored directly. The `password_hash()` function is used during registration (`register_user.php`), and `password_verify()` is used during login (`login.php`) to securely check credentials.
- **Database Security:** To prevent SQL Injection (a major security vulnerability), every interaction with the MySQL database (including inserts, updates, deletes, and search queries) utilizes **prepared statements**. This separates the SQL command structure from user data.
- **Session Management:** The `$_SESSION` superglobal is used to maintain state (`user_id`, `user_name`, `user_role`) after successful login, ensuring personalized content and access control (e.g., restricting `admin/dashboard.php` to admin roles only).

3.2 Client-Side Interactivity and Validation (JavaScript/ILO 2)

Client-side code enhances usability and reduces unnecessary server load.

- **Form Validation:** The `assets/script.js` file implements client-side validation for the registration form (`registration_form.php`). It verifies mandatory fields, email format (`/^[^\s@]+@[^\s@]+\.[^\s@]+$/`), and student ID format before the data is sent to the server.
- **Admin Confirmation:** A JavaScript `prompt()` function is used in the Admin Dashboard (`admin/dashboard.php`) to require a confirmation phrase (`'DELETE'`) before executing the event deletion script, preventing accidental data loss.

3.3 Database Integration and CRUD (MySQL/ILO 4, 5)

The application demonstrates full integration between the front-end and the relational database.

- **CRUD Implementation:** The `admin/event_form.php` script handles both the **C**reate (INSERT) and **U**pdate (UPDATE) operations within a single, unified form, simplifying maintenance. The **D**elete operation is managed by `admin/delete_event.php`.
- **Filtering and Querying:** The `index.php` page dynamically builds complex SQL queries based on user inputs (`search_term` and `sort_by`), executing them via prepared statements to fetch the filtered event listing.

4. Reflection on Learning Outcomes and Challenges

4.1 Learning Outcomes Achieved (ILO 1-6)

ILO	Achieved Implementation
ILO 1: Design responsive web pages.	Achieved via flexible box model (`flex`) and CSS media queries in `assets/style.css`.
ILO 2: Implement client-side validation.	Achieved using `assets/script.js` for real-time form validation checks.
ILO 3: Develop server-side logic and session management.	Achieved through PHP scripts for Login/Logout and the `$_SESSION` array.
ILO 4: Design and query a relational database.	Achieved via the 3-table schema (`users`, `events`, `registrations`) and complex JOIN queries for the Admin Dashboard.
ILO 5: Integrate front-end and back-end.	Achieved successfully, linking HTML forms to PHP processing and using PHP to dynamically render data retrieved from MySQL.
ILO 6: Demonstrate code modularity and security.	Achieved by separating configuration (`db_connect.php`), logic, and views, and strictly using PDO prepared statements.

4.2 Challenges Faced

1. **Database Connection Scope:** A major technical challenge was ensuring the database connection object (`$pdo`) was correctly scoped within multiple PHP files and functions (`get_participants`, `is_admin`). This was resolved by using the `global $pdo;` keyword in necessary functions and ensuring all included files were referenced using the correct relative paths (e.g., `../config/db_connect.php`).
2. **Password Hashing:** Debugging failed login attempts required careful verification that the initial password hash inserted by `db_structure.sql` was correctly generated and

that the `password_verify()` function was being executed against the correct column (`password_hash`).

End of Report.