North South University

Department of Electrical & Computer Engineering (ECE)

CSE-327 Software Engineering Project Report

Campus Events Management System

Student IDs

Md. Abul Bashar Nirob – 2022198042

Istiak Ahasan – 2010282042

Submitted To: MdSH

Section: 03

Group: 01

iNSUre: A Campus Event Management System

Prepared by: Md Abul Bashar Nirob

Date: 15 August 2025

Table of Contents

Executive Summary

The **"iNSUre"** Campus Event Management System **(CEMS)** is a comprehensive web-based application designed to streamline the process of organizing, managing, and participating in university events. Developed as a project for CSE-327, this system addresses the logistical challenges faced by university departments, event organizers, and students. The platform provides a centralized hub where events can be created, promoted, and tracked, fostering a more vibrant and organized campus community.

The system is built on a robust, role-based access control architecture, catering to three primary user groups: Administrators, Event Organizers, and Users (students/faculty). Administrators have overarching control, managing user accounts and all events. Organizers are empowered to create and manage their specific events, view participant lists, gather feedback. General users can browse and register for events, subsequently provide valuable feedback.

Key functionalities of the CEMS include secure user authentication, dynamic event creation with email notifications via PHPMailer, comprehensive event and user management dashboards, and a user-friendly interface for event discovery and registration. The project was developed using PHP for server-side logic, MySQL for database management, and a combination of HTML, CSS, and JavaScript for the front-end interface, ensuring a responsive and intuitive user experience. This report details the project's journey from conception to completion, covering project management, requirement specifications, system architecture, design, and testing.

1. Introduction

1.1 Purpose and Scope

The primary purpose of the "**iNSUre**" Campus Event Management System is to provide a centralized, efficient, and user-friendly platform for managing all event-related activities within a university campus. The system aims to replace fragmented and manual processes (such as posters, emails, social media posts) with a single, integrated solution.

The scope of this project encompasses the entire lifecycle of an event, from its initial creation by an organizer to post-event feedback from attendees. This includes:

- ✓ **User Registration and Authentication:** Securely managing user accounts with distinct roles.
- ✓ **Event Creation and Management:** Allowing authorized organizers to publish and update event details.
- ✓ **Event Discovery and Registration:** Enabling users to easily find and sign up for events.
- ✓ **Administrative Oversight:** Providing administrators with the tools to manage users & events campus-wide.
- ✓ **Communication:** Automating notifications to keep users informed about new events.
- ✓ **Feedback Collection:** Facilitating a feedback loop between attendees and organizers.

1.2 Product Overview

CEMS is a dynamic, database-driven web application. It offers a role-based experience to streamline event management. Key Capabilities:

1. **Role-Based Dashboards:** Each user role (Admin, Organizer, User) is presented with a tailored dashboard that provides access to relevant functionalities.
2. **Centralized Event Listing:** A public-facing page displays all upcoming and past events, which can be filtered or sorted.
3. **Secure User Management:** Administrators can add, edit, and delete user accounts, with safeguards to prevent the deletion of users with active registrations or feedback history.
4. **Dynamic Event Creation:** Organizers can create events with detailed information, including title, description, date, time, location, category, and associated department.
5. **Automated Email Notifications:** The system leverages the PHPMailer library to automatically send email notifications to all registered users upon the creation of a new event, ensuring wide visibility.
6. **Participant and Feedback Viewing:** Organizers have dedicated sections to view a list of registered participants for their events and to read feedback submitted by attendees.

Scenarios for Using the Product:

- **Scenario 1 (A Student Discovering and Registering for a Workshop):**
  A student logs into the CEMS portal to see what's happening on campus. They browse the "Upcoming Events" section and find a programming workshop hosted by the CSE department. They click on the event to view details and then click the "Register" button. They are now registered, and the event organizer can see their name on the participant list.

- **Scenario 2 (An Organizer Creating a New Seminar):**
  A faculty member, acting as an event organizer, logs in and navigates to their dashboard. They select "Create Event," fill out a form with details about an upcoming seminar, and submit it. The system immediately adds the seminar to the public event list and sends an email notification to all students and faculty.

- **Scenario 3 (An Administrator Managing a User Account):**
  An administrator receives a request to update a user's role. They log into the admin dashboard, go to "Manage Users," find the specific user, and edit their profile to assign them a new role, for instance, promoting a user to an organizer.

1.3 Structure of the Document

This report is structured to provide a comprehensive overview of the CEMS project. Section 1 introduces the project's purpose and scope. Section 2 details the project management plan, including the lifecycle model, risk analysis, and scheduling. Section 3 outlines the requirement specifications, including stakeholders and use cases. Section 4 describes the system's architecture. Section 5 delves into the detailed design of the system, including GUI, static, and dynamic models. Finally, Section 6 presents the test plan, detailing the testing strategy and cases.

1.4 Terms, Acronyms, and Abbreviations

- ✓ **CEMS:** Campus Event Management System
- ✓ **GUI:** Graphical User Interface
- ✓ **PHP:** Hypertext Preprocessor
- ✓ **MySQL:** My Structured Query Language
- ✓ **JS:** JavaScript
- ✓ **UC:** Use Case

2. Project Management Plan

2.1 Project Organization

The project was developed by a team of two members, with responsibilities divided to ensure efficient workflow and comprehensive coverage of the project's requirements.

- ➢ **Md Abul Bashar Nirob:** Led the front-end development, focusing on user interface (UI) and user experience (UX) design using HTML, CSS, and JavaScript, as well as integrating the front-end with the back-end services.

- ➢ **Istiak Ahasan:** Focused on back-end development, including database design, server-side scripting with PHP, and implementing the core application logic.

Both team members collaborated on system design, testing, and documentation.

2.2 Lifecycle Model Used Project

The project adopted an Iterative Development Model. This approach was chosen over a strict Waterfall model to allow for flexibility and incremental progress. The development process was broken down into smaller, manageable cycles. Each iteration involved planning, requirement analysis, design, implementation, testing of a specific subset of features.

For example, the first iteration focused on user authentication and basic dashboards, while subsequent iterations added event creation, management, and feedback functionalities. This allowed for early feedback and the ability to refine features as the project progressed.

2.3 Risk Analysis

Risk	Probability	Impact	Mitigation Strategy
Scope Creep	Medium	High	Clearly defined project scope from the outset. All change requests were evaluated for necessity and impact on the schedule before implementation.
Technical Difficulties	Medium	Medium	Regular team meetings to discuss technical challenges. Prototyping complex features (like email integration) before full implementation.
Time Management	High	High	A project schedule with milestones was created. Weekly progress reviews were conducted to ensure the project stayed on track.
Data Security	Low	High	Implemented password hashing (PASSWORD_DEFAULT) for user credentials. Ensured proper session management and role-based access control to protect sensitive data.

2.4 Hardware and Software Resource Requirements

Hardware:

- Developer Workstations (Laptops) with standard configurations (Intel Core i5/i7, 8GB+ RAM, SSD).
- A web server for hosting the application (XAMPP/WAMP for local development).

Software:

- Operating System: Windows 10/11
- Web Server: Apache (via XAMPP)
- Database: MySQL (via XAMPP)
- Programming/Scripting Languages: PHP 7.4+, HTML5, CSS3, JavaScript (ES6)
- IDE/Text Editor: Visual Studio Code
- Version Control: Git / GitHub
- Web Browser: Google Chrome, Mozilla Firefox for testing.
- Libraries: PHPMailer for email notifications.

New Skills Acquired:

1. Gained in-depth experience with advanced PHP features, secure session management, and integrating third-party libraries like PHPMailer for handling complex functionalities like email services.

2. Enhanced skills in responsive web design using modern CSS techniques and JavaScript for creating dynamic and interactive user interfaces. Gained experience in connecting front-end forms and actions to back-end PHP scripts.

2.5 Deliverables and Schedule

Deliverable	Planned Date	Actual Date
Project Proposal	Week 1	Week 2
Requirement Specification	Week 4	Week 4
System Architecture & Design	Week 6	Week 7
Initial Prototype (Login & Dashboards)	Week 8	Week 8
Mid-term Presentation	Week 9	Week 9
Feature Complete System	Week 12	Week 13
Final Testing & Debugging	Week 13	Week 14
Final Report & Presentation	Week 14	Week 14

2.6 Professional Standards

The project adhered to several professional standards:

a. **Coding Standards:** Followed PSR (PHP Standards Recommendations) for PHP code to ensure readability and maintainability.

b. **Documentation:** Maintained clear and concise comments within the code. This final report follows a standard software engineering template.

c. **Version Control:** Used Git for version control, with meaningful commit messages to track changes systematically.

d. **Security:** Followed best practices for web security, including password hashing and protection against SQL injection (though the current code uses variable interpolation, prepared statements would be the next step for enhanced security).

2.7 Monitoring, Reporting, and Controlling Mechanisms

A. **Monitoring:** Progress was monitored through weekly team meetings and a shared task list.

B. **Reporting:** Each team member reported their weekly progress and any impediments during the meetings.

C. **Controlling:** The project schedule and task list served as the primary control mechanisms. Any deviations from the plan were discussed and corrective actions were taken, such as re-allocating tasks or adjusting timelines for specific features.

2.8 Impact of the Project

The CEMS project has a significant positive impact on both individuals and the university as an organization:

- ➢ **For Individuals (Students & Faculty):** It provides a single, reliable source for information on campus events, increasing participation and engagement. It simplifies the registration process and offers a platform for their voices to be heard through feedback.
- ➢ **For Organizations (University & Departments):** It offers a powerful tool for promoting events and managing them efficiently. It reduces the administrative overhead associated with event management and provides valuable data through participant lists and feedback, which can be used to improve future events.

2.9 Configuration Management

All project artifacts, including source code, documentation, and this report, were placed under configuration management using Git and GitHub. A central repository was used to host the project. This ensured that:

- ✓ Every change to the codebase is tracked.
- ✓ Team members could work on different features concurrently without conflicts.
- ✓ A complete history of the project's development is maintained, allowing for easy rollbacks if necessary.
- ✓ The final, stable version of the code is securely stored and versioned.

3. Requirement Specifications

3.1 Stakeholders

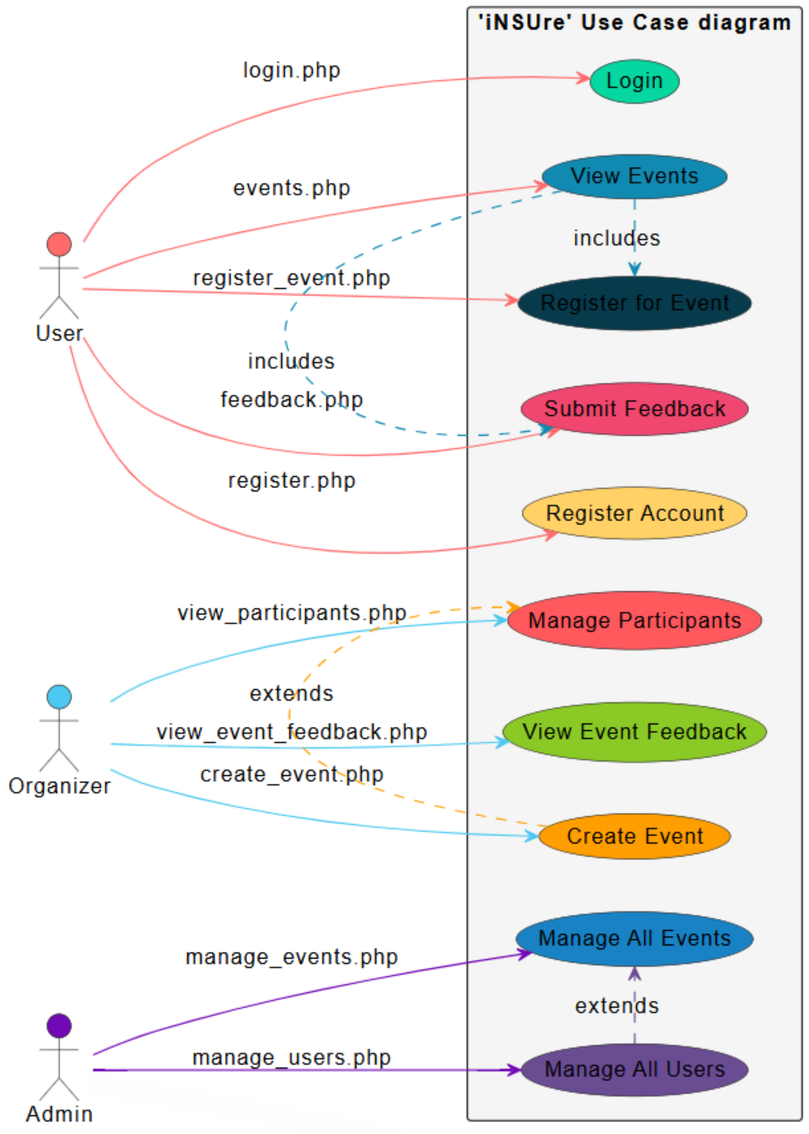
a. **University Students & Faculty (Users):**
They need a simple way to find and register for events.

b. **Event Organizers (e.g., Club Heads, Faculty):**
They need tools to create, promote, and manage their events effectively.

c. **University Administration (Admins):**
They require oversight of all campus activities and the ability to manage user access.

d. **Department Heads:**
They are interested in events related to their specific department

3.2 Use Case Diagram Model



3.2.1 Textual Description for each Use Case

__UC-1: Register for Account__

- ✓ **Actor:** Guest (Unregistered User)
- ✓ **Description:** A guest can register for a new account by providing their name, email, password, role, department, and contact number.
- ✓ **Pre-conditions:** The user must not have an existing account with the provided email.
- ✓ **Post-conditions:** A new user account is created in the system, and the user can now log in.

UC-2: Login

- **Actor:** User, Organizer, Admin
- **Description:** A registered user can log into the system using their email and password.
- **Pre-conditions:** The user must have a registered account.
- **Post-conditions:** The user is authenticated and redirected to their respective dashboard based on their role.

UC-3: Manage Users

- **Actor:** Admin
- **Description:** An admin can view, edit, and delete user accounts.
- **Pre-conditions:** The admin must be logged in.
- **Post-conditions:** The user list is updated.

UC-4: Manage Events

- **Actor:** Admin, Organizer
- **Description:** An admin can manage all events. An organizer can manage only the events they have created. This includes creating, editing, and deleting events.
- **Pre-conditions:** The actor must be logged in.
- **Post-conditions:** The event list is updated.

UC-5: View Events

- **Actor:** User, Organizer, Admin, Guest
- **Description:** Any user (logged in or not) can view the list of upcoming and past events. Logged-in users can also see a more detailed view and have the option to register.
- **Pre-conditions:** None.
- **Post-conditions:** The user is shown a list of events.

UC-6: Register for Event

- **Actor:** User
- **Description:** A logged-in user can register for an upcoming event.
- **Pre-conditions:** The user must be logged in and not already registered for the event.
- **Post-conditions:** The user's registration is recorded in the database.

UC-7: Submit Feedback

- **Actor:** User
- **Description:** A logged-in user can submit feedback and a rating for an event they have attended
- **Pre-conditions:** The user must be logged in and must have been registered for the event, and the event must have already occurred.
- **Post-conditions:** The feedback is stored in the database.

UC-8: View Participants

- ✓ **Actor:** Organizer
- ✓ **Description:** An organizer can view the list of participants who have registered for their events.
- ✓ **Pre-conditions:** The organizer must be logged in.
- ✓ **Post-conditions:** A list of participants is displayed.

3.3 Rationale for Use Case Model

The use case model was designed to be comprehensive yet straightforward, directly reflecting the core functionalities of the CEMS. The actors (Admin, Organizer, User) were chosen to represent the distinct roles within the system, and the use cases were derived from the primary goals of these actors. This model provides a clear and unambiguous representation of the system's intended behavior and was instrumental in guiding the development process.

3.4 Non-functional Requirements

a. **Performance:** The system should load all pages within 3 seconds. Database queries should be optimized for fast retrieval of event and user data.

b. **Security:** All user passwords must be hashed. The system must prevent unauthorized access to role-specific dashboards.

c. **Usability:** The user interface should be intuitive and easy to navigate for all user types. The registration and event creation processes should be straightforward.

d. **Reliability:** The system should be available 24/7, with minimal downtime.

e. **Scalability:** The system should be able to handle an increasing number of users and events without a significant degradation in performance.

4. Architecture

4.1 Architectural Style

The CEMS application is built using a combination of two architectural styles:

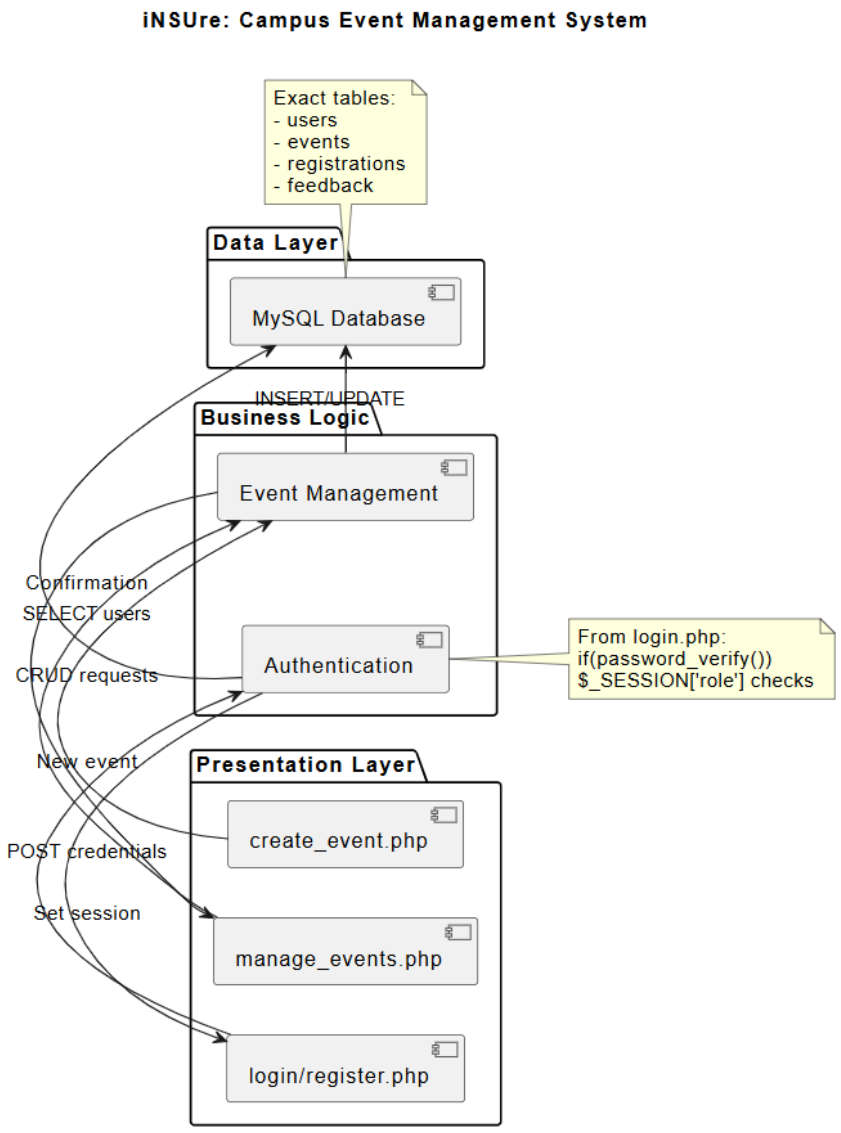
1. **Layered (3-Tier) Architecture:** This is the primary style used. The application is logically divided into:

    - **Presentation Tier:** The user interface, consisting of HTML, CSS, and JavaScript files that are rendered in the user's browser.

- **Logic/Application Tier:** The PHP scripts that handle the business logic, process user requests, and interact with the database.
- **Data Tier:** The MySQL database that stores all the application data, such as user information, events, and registrations.

2. **Client-Server Architecture:** The system operates on a client-server model where the user's web browser (the client) sends HTTP requests to the web server (the server), which then processes the requests and sends back responses.

4.2 Architectural Model



4.3 Technology, Software, and Hardware Used

As detailed in Section 2.4, the core technologies are the LAMP stack (Linux, Apache, MySQL, PHP), although developed in a Windows environment using XAMPP. The front-end relies on standard web technologies (HTML, CSS, JS), and the back-end uses PHP for its logic.

4.4 Rationale for Architectural Choices

The 3-Tier Architecture was chosen for its clear separation of concerns. This separation makes the application easier to develop, maintain, and scale. For instance, the front-end developers could work on the UI independently of the back-end developers. It also improves security, as the data tier is not directly accessible from the presentation tier.

The Client-Server model is the standard for web applications and was a natural choice for this project. PHP was selected as the server-side language due to its wide adoption, strong community support, and ease of integration with MySQL and Apache.

5. Design

5.1 GUI Design

The GUI was designed to be clean, professional, and user-friendly. A consistent theme is used across all pages, with a common header and footer. Key design elements include:

a. **Header:** Contains the application logo and a navigation bar that dynamically changes based on the user's login status and role.

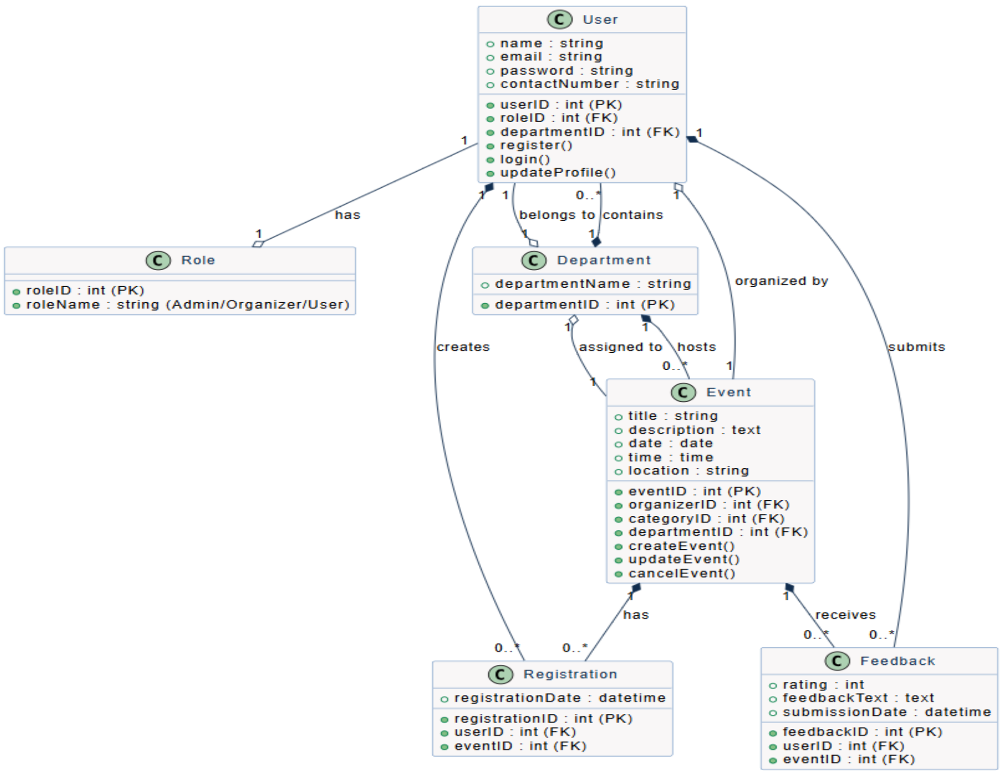
b. **Forms:** All forms (login, registration, event creation) are designed for clarity, with clear labels and required field indicators.

c. **Tables:** Data, such as lists of users and events, is presented in well-formatted tables for easy reading.

d. **Responsive Design:** CSS media queries and flexible layouts are used to ensure the application is usable on various screen sizes, from desktops to mobile devices.

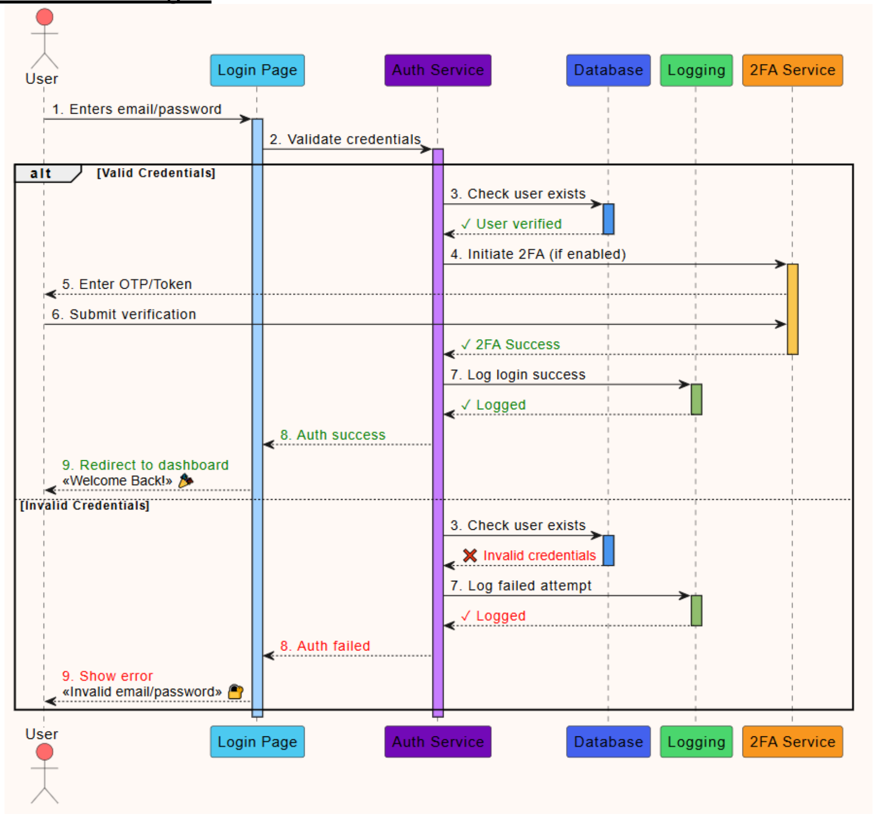
5.2 Static Model (Class Diagrams)

The static model is primarily represented by the database schema, which can be visualized as a class diagram where each table is a class.

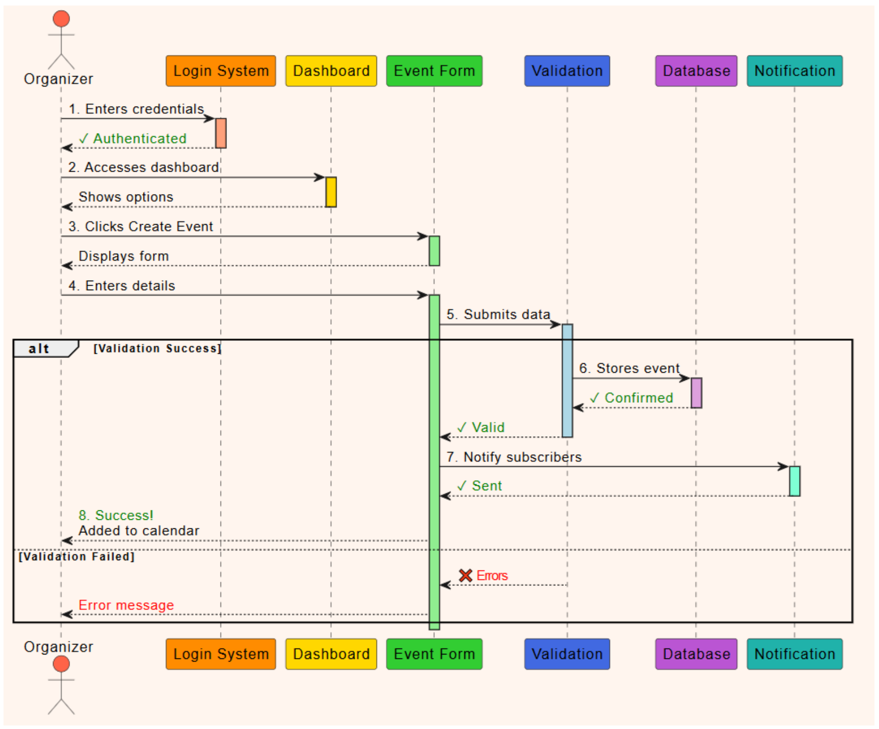


5.3 Dynamic Model (Sequence Diagrams)

Sequence Diagram for User Login:



Sequence Diagram for Create Event:



5.4 Rationale for Detailed Design Model

The design choices were driven by the need for a maintainable and scalable system. The database schema is normalized to reduce data redundancy. The use of separate PHP files for different functionalities (e.g., login.php, create_event.php) promotes modularity. The sequence diagrams illustrate the flow of interactions within the system for key use cases, which helped in identifying potential bottlenecks and ensuring that the logic was sound before implementation.

5.5 Traceability

The design is directly traceable to the requirements. **For example:**

- ✓ Login is implemented in login.php and its corresponding sequence diagram shows the interaction flow.
- ✓ Manage Events is implemented through the create_event.php, edit_event.php, and delete_event.php files, and the database schema includes the Events table to support this.
- ✓ Non-functional requirement for Security is addressed by using password_hash() in register.php and password_verify() in login.php.

6. Test Plan

6.1 Requirements/Specifications-Based System Level Test Cases

Test Case ID	Requirement	Test Description	Actual Outcome	Expected Outcome	Status
TC-01	UC-1: Register for Account	Attempt to register with valid, unique information.	User is registered successfully and can log in.	User is registered successfully.	Pass
TC-02	UC-1: Register for Account	Attempt to register with an existing email.	System displays an error message.	System displays an error.	Pass
TC-03	UC-2: Login	Attempt to log in with valid credentials for each role (Admin, Organizer, User).	User is logged in and redirected to the correct dashboard.	User is redirected correctly.	Pass
TC-04	UC-2: Login	Attempt to log in with an invalid password.	System displays an "Invalid password" error.	Error message is displayed.	Pass
TC-05	UC-4: Create Event	Log in as an Organizer and create a new event.	The event is created and appears in the "Manage My Events" list and the public events list.	Event is created and displayed.	Pass
TC-06	UC-6: Register for Event	Log in as a User and register for an upcoming event.	A success message is shown, and the user cannot register for the same event again.	Success message is shown, and re-registration is blocked.	Pass

6.2 Traceability of Test Cases to Use Cases

Use Case	Associated Test Case(s)
UC-1: Register for Account	TC-01, TC-02
UC-2: Login	TC-03, TC-04
UC-3: Manage Users	TC-08
UC-4: Manage Events	TC-05
UC-6: Register for Event	TC-06
UC-7: Submit Feedback	TC-07

6.3 Techniques Used for Test Generation

- **Black-Box Testing:**
  Test cases were designed based on the system's requirements and specifications without looking at the internal code structure. This includes techniques like:

  a. **Equivalence Partitioning:** Grouping inputs into classes (e.g., valid emails, invalid emails).
  b. **Boundary Value Analysis:** Testing at the boundaries of input domains (e.g., password length).

- **White-Box Testing**:
  The code was inspected to ensure that different logical paths were covered. For example, testing the if-else conditions in the login and registration logic.

- **Usability Testing:**
  The system was tested by team members acting as end-users to evaluate the ease of use and overall user experience.

6.4 Assessment of the Goodness of Your Test Suite

The quality of the test suite was assessed using the following metrics:

1. **Requirements Coverage:** This was the primary metric. The goal was to have at least one test case for every functional requirement defined in the use case model. The traceability matrix (Table 3) demonstrates that all major use cases were covered.

2. **Defect Detection Rate:** The number of bugs found during the testing phase. The test suite was effective in identifying several logical errors and UI inconsistencies, which were subsequently fixed before the final delivery.

3. **User Feedback:** Informal usability testing provided qualitative feedback that helped in refining the user interface and improving the overall user experience.

The test suite was deemed effective as it successfully verified all core functionalities and helped in delivering a stable and reliable application.

Acknowledgment

We would like to express our sincere gratitude to our course instructor MdSH Sir for CSE-327 for their guidance, support, and valuable feedback throughout the duration of this project. Their insights were instrumental in shaping the project and helping us navigate the challenges of software development.

References

[1] W3Schools. "PHP Tutorial." [Online]. Available: https://www.w3schools.com/php/ . [Accessed: Aug. 10, 2025].

[2] PHPMailer Project. "PHPMailer." [Online]. Available: https://github.com/PHPMailer/PHPMailer [July 25, 2025].

[3] Oracle Corporation. "MySQL Documentation." [Online]. Available: https://dev.mysql.com/doc/ Aug. 5, 2025].

GitHub Link

https://github.com/istiakahasan/Campus_event_management_system/blob/main/README.md