Smart Campus Event Management and Notification System with Role-Based Access

---

1. Project Overview

This is a **web-based system** designed to help campus administrators, event organizers, and students efficiently manage and participate in campus events. The system allows:

- **Admins** to oversee the entire system and approve events.
- **Event organizers** to create and manage their events.
- **Attendees (students/faculty)** to view, register, and get notifications for events.

It uses **Role-Based Access Control (RBAC)** to ensure users see only what they're allowed to.

---

2. Objectives

- Provide an easy platform to create and manage campus events.
- Automate notifications to keep all users informed.
- Ensure security with user roles and permissions.
- Replace inefficient manual or email-based event announcements.
- Improve event participation and campus engagement.

---

3. Key Technologies

Technology	Purpose
HTML, CSS, Bootstrap	Frontend UI design and responsiveness
JavaScript	Frontend interactivity, form validation, notifications
Node.js (or PHP)	Backend logic, role-based access control, API endpoints
MySQL / SQLite	Database for storing users, events, roles, registrations
Email API (SendGrid, etc.)	Sending notifications and alerts
Session / JWT	Authentication and user session management

---

4. User Roles and Permissions

Role	Permissions
Admin	Manage users, approve events, view all events & registrations, manage system settings
Event Organizer	Create, update, cancel own events, view registrations for their events
Attendee	View available events, register/RSVP for events, receive notifications

---

5. Features in Detail

A. User Management

- **Registration/Login:** Users create accounts with role assignment done by Admin (or default to attendee).
- **Authentication:** Secure login using sessions or JWT tokens.
- **Profile Management:** Users can update personal info.

B. Event Management

- **Create Event:** Organizer fills out event form — title, description, date/time, location, category.
- **Edit/Delete Event:** Organizers can modify or cancel their events.
- **Approval Workflow:** Newly created events must be approved by Admin before becoming public.
- **Event Categories:** Workshops, Seminars, Sports, Cultural, etc.

C. Event Registration

- Attendees browse events and RSVP or register.
- Organizer and Admin can view the attendee list.

D. Notification System

- **Automated Emails:** When events are created, approved, updated, or canceled.
- **Reminders:** Sent before event start (e.g., 1 day or 1 hour prior).
- **In-app Notifications:** Displayed in user dashboard when logged in.

E. Dashboard & Analytics

- **User Dashboard:** Shows personalized upcoming events and notifications.

- **Admin Dashboard:** Overview of events, registrations, pending approvals, and basic participation analytics (number of attendees, popular events).

F. Search & Filter

- Users can filter events by date, category, location, or keyword.

G. Security

- Password encryption and secure login.
- Role-based access control prevents unauthorized actions.
- Input validation to avoid injection attacks.

---

6. Database Design (Simplified)

Table Name	Fields (examples)	Description
Users	user_id (PK), name, email, password_hash, role_id	Stores user info & roles
Roles	role_id (PK), role_name	Defines user roles (Admin, Organizer, Attendee)
Events	event_id (PK), title, description, date, time, location, organizer_id (FK), status (pending/approved)	Stores event details
Registrations	reg_id (PK), event_id (FK), user_id (FK), registration_date	Tracks user event registrations
Notifications	notif_id (PK), user_id (FK), message, status (read/unread), created_at	Stores notifications for users

---

7. System Architecture

```
[Frontend (HTML/CSS/JS/Bootstrap)]
          |
          v
[Backend (Node.js or PHP)] <---> [Database (MySQL)]
          |
          v
[External APIs: Email Service (SendGrid, etc.)]
```

- Frontend interacts with backend via REST API.
- Backend applies business logic (RBAC, event workflows).
- Database stores all persistent data.
- Email API sends notifications.

---

8. Development Plan

Phase 1: Setup & Basic Functionality

- Setup project environment and database.
- Implement user registration/login with role assignment.
- Build event creation and listing pages.

Phase 2: Role-Based Access Control

- Implement RBAC middleware on backend routes.
- Restrict event approval, creation, and editing based on roles.

Phase 3: Event Approval & Notification System

- Admin panel for event approval.
- Integrate email API for notifications.
- Implement notification dashboard and reminders.

Phase 4: Advanced Features & UI Enhancements

- Search and filter events.
- Event registration system.
- User dashboards and analytics for admins.

Phase 5: Testing & Deployment

- Test all user flows and security.
- Deploy on server or cloud.
- Final documentation and user manual.

---

9. Possible Challenges & Solutions

Challenge	Solution
Managing role permissions properly	Use middleware or centralized RBAC functions
Ensuring notification delivery	Use reliable email APIs, fallback to in-app alerts
Handling concurrent event edits	Locking or versioning mechanisms
Keeping UI simple and responsive	Use Bootstrap and client-side JS validation

---

10. Why This Project?

- **Real-World Impact:** Campus-wide solution improving communication and event management.
- **Learning Opportunity:** Covers full web app development, databases, security, networking, and system administration.
- **Innovation:** Automated notifications + role-based system tailored to campus needs.
- **Scope & Feasibility:** Balanced complexity that is achievable with your skillset and resources.
