

Campus Event Aggregator: An AI-Enabled Platform for Personalized Event Recommendation, Smart Notifications, and Secure QR-Based Attendance

M. Koushik, N. Nikileswar, M C. Namitha, M. Many, V. Meghashree

Department of Computer Science and Engineering, Rajiv Gandhi Institute of Technology, Bengaluru, Karnataka, India

Published October 2026

Abstract

Colleges host a range of academic and non-academic events that students enjoy, yet details about these events are scattered across many channels, making it hard for students to discover events that fit their interests and to manage attendance. This paper presents the Campus Event Aggregator, an integrated platform that extends an event-discovery and QR-based check-in system with intelligence-driven personalization. The proposed system combines a content-based recommendation module that ranks events using cosine similarity over TF-IDF feature vectors. It also includes a notification module that filters alerts by relevance and timing, a QR-code attendance module with fraud-prevention checks, and an event calendar. All these features are delivered through a web platform. The system architecture, methodology and a working implementation built on the MERN stack (MongoDB, Express, React, Node.js) with Socket.IO for real-time updates are described. The Campus Event Aggregator brings all information into one place, gives tailored suggestions, sends notifications and uses secure QR codes for attendance tracking, while providing organizers with real-time analytics and management tools.

Keywords: artificial intelligence; campus event management; event recommendation; QR-based attendance; smart notifications

1 Introduction

1.1 Background

Colleges and universities host academic and non-academic events throughout the year. Because information about these events is distributed across many communication channels, students often find it difficult to discover events and to manage their attendance. The Campus Event Aggregator, developed earlier by the authors, was created to ease this process by bringing event discovery, registration, and QR-based check-in onto a single platform.

This work builds upon that existing platform by introducing AI-based recommendations, smart notifications, security-enhanced QR-code attendance, and a personalized event calendar for students.

1.2 Problem statement

Information about college events is dispersed across several communication channels, making it difficult for students to find events that interest them and to manage their attendance. Existing event management systems often lack personalization. They also provide only basic alerts and simple attendance records, and remain vulnerable to misuse. An integrated, intelligent college event management system is therefore needed—one that recommends events based on user preferences,

sends timely notifications, ensures secure attendance verification and provides a personal event calendar.

1.3 Motivation

This work is motivated by a problem experienced by college students. A large number of activities take place on campus, yet students frequently miss the ones that would genuinely interest them. Information about these events is often spread across groups or social-media channels, and organizers face challenges in managing attendance and engagement. While the authors' earlier system addressed some of these issues, there remains room for improvement in personalization and event management.

This work therefore seeks to add personalization, security and usability to the platform through AI-based recommendations, notifications, secure QR attendance and a personalized event calendar.

1.4 Objectives

The primary objectives of the proposed system are:

1. To bring all information about college events onto a single platform for convenient access and registration.
2. To recommend events to users based on their interests, profile, and past attendance.
3. To send smart notifications that reduce missed events and irrelevant alerts.
4. To use QR codes for secure attendance, with measures to minimize proxy attendance.
5. To offer a personal event calendar for managing registered and recommended events.
6. To increase student engagement through a convenient and personalized event experience.

1.5 Contributions

The key contributions of the proposed system are:

1. *AI-Based Event Recommendations*: recommending suitable college events based on the interests, profile, and participation history of students.
2. *Smart Notifications*: delivering relevant event notifications while avoiding missed events and irrelevant alerts.
3. *Secured QR Attendance*: strengthening QR-based attendance with validation and fraud-prevention features.
4. *Personalized Event Calendar*: organizing registered and recommended events for each student.
5. *Campus Platform Integration*: combining event search, registration, recommendation, notification, attendance, and management in a single platform.
6. *Higher Student Engagement*: providing students with a more convenient and personalized way to engage in campus activities.

2 Literature survey

2.1 Personalized event recommendation

Event recommendation has been extensively studied in event-based social networks. SoCaST, proposed by Ogundele et al., accounts for geographical, categorical, social, and temporal preferences

when recommending events [1]. EventRec explored personalized recommendations using user and event features [2]. Liu et al. presented SERGE, which models the user–event relationship for successive event recommendation [3]. Bok et al. demonstrated the value of analyzing user preferences and participation histories for personalized event recommendation [4]. These works show that user preferences, event features, and participation histories can meaningfully improve recommendations; however, most focus on general event-based social networks rather than on college settings specifically.

2.2 Smart campus and student event platforms

Research on smart campuses focuses on using digital technologies to improve campus services and experience. Yang et al. presented a situational-awareness framework for smart campuses [12], while Zhang et al. discussed technologies and applications for human-centered smart-campus environments [13]. More recently, Dong et al. examined frameworks and services for integrated smart-campus systems [15]. The authors' earlier Campus Event Aggregator supported event discovery, QR check-ins, rewards, dashboards, and real-time communication, but lacked personalized event discovery—leaving room for AI-driven event recommendation.

2.3 QR-based attendance and secure check-in

QR-based attendance has been proposed as an efficient alternative to manual attendance systems [5], [6]. Other work has incorporated location tracking and geolocation to strengthen attendance-verification processes [7], [8]. The authors' previously developed system used time-based QR codes together with attendance logging for event check-in; however, QR-code credentials can still be compromised. The current system therefore integrates additional time-based security measures into the QR attendance workflow.

2.4 Intelligent notifications and personalized scheduling

Research on intelligent notifications includes analysis of user context, behavior, and reaction models to determine the timing and relevance of notifications [9]. Predicting user reactions to notifications using machine-learning techniques is another active area of research [10], [11]. These results support the use of personalized notification strategies on campus, whereas traditional event systems typically send notifications unrelated to a student's interests or schedule.

2.5 Research gap

The reviewed literature covers event recommendation, smart-campus applications, QR-code attendance, and intelligent notification services largely as independent areas [1]–[15]. Little attention has been paid to an integrated platform that combines all of these capabilities for the college community. The proposed system therefore builds on the authors' previous Campus Event Aggregator by adding intelligent notifications, secure QR-code attendance with fraud protection, and AI-based personalized event scheduling—directions that prior work identified as opportunities for further study.

3 Proposed system

3.1 System overview

The proposed Campus Event Aggregator is a central platform through which students can discover, register for, and manage college events. The system provides event information, personalized

recommendations and notifications, event registration, and a personal calendar for upcoming events. It further integrates QR-based registration with verification to prevent duplicate attendance.

3.2 System architecture

The system follows a modular design comprising a user interface, an application server, a database, AI-enabled services, and an attendance module, as illustrated in Fig. 1. The user interface handles interaction between users and the system, while the application server manages event registration, notifications, recommendations, and attendance requests. The database stores information about users, events, registrations, and attendance. The AI-enabled services generate recommendations and notifications related to events, while the QR-based attendance module authenticates attendees and records attendance in order to prevent duplicate check-ins.

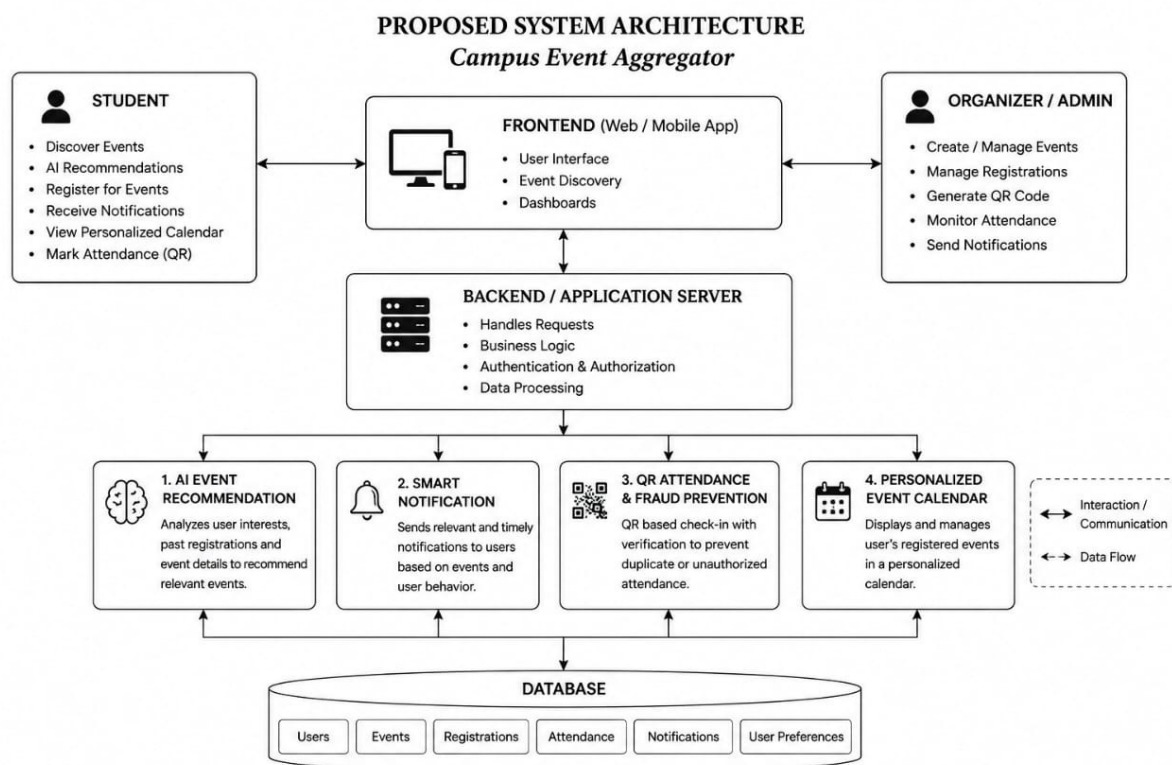


Figure 1. Proposed system architecture of the Campus Event Aggregator.

3.3 User roles

The proposed solution defines two primary user roles: Student and Organizer/Administrator. Students can discover events, receive recommendations and notifications, register for events, view upcoming events on a personal calendar, and check in using QR codes. Administrators can create and manage events, handle registrations, generate QR codes, track attendance, and update event details.

3.4 System workflow

As shown in Fig. 2, the workflow begins when a student signs in and browses available campus events. Event recommendations are generated based on the student's interests and past actions. After selecting an event, the student registers for it; timely notifications are sent, and the registered event is added to the student's personal calendar. During the event, the student presents the event's QR

code, which the system verifies against registration and presence records before logging participation.

4 Methodology

The proposed Campus Event Aggregator uses web technologies together with AI to support event discovery and attendance for campus events. The system offers event management, recommendation, notification, personalized calendaring, and QR-based attendance with fraud prevention.

4.1 Technical stack

The system consists of a frontend, backend, database, AI module, and QR attendance module. The principal technologies and their roles are summarized in Table 1.

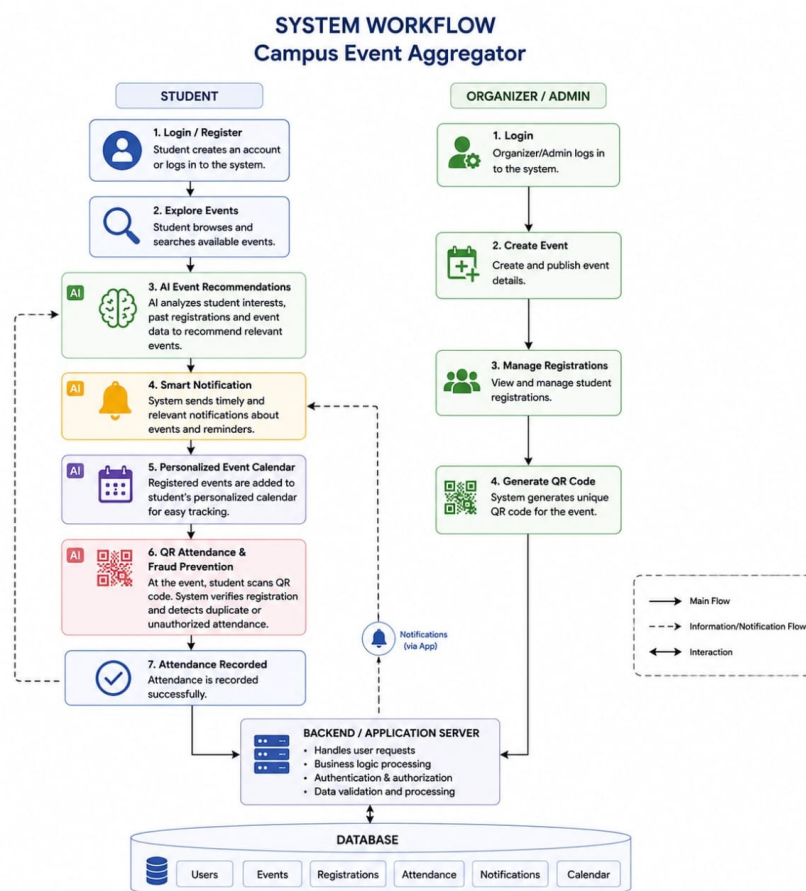


Figure 2. System workflow of the Campus Event Aggregator.

Table 1. Technical stack.

Component	Technology	Purpose
Frontend	React.js / HTML, CSS, JS	User interface
Backend	Node.js / Express.js	Server and business logic
Database	MongoDB / Firebase	Data storage
API	REST API	Frontend–backend communication
AI	Python / ML	Event recommendation

Component	Technology	Purpose
QR	QR code	Attendance verification

4.2 Database design

The database stores user, event, registration, notification, and attendance information. User interests and event details are also consumed by the AI recommendation module, as summarized in Table 2.

Table 2. Database entities.

Entity	Purpose
User	Stores user details and interests
Club	Stores club information
Event	Stores event details
Registration	Stores event registrations
Attendance	Stores verified attendance
Notification	Stores notification records

4.3 API design

REST APIs are used for communication between the frontend and backend. The principal APIs are listed in Table 3.

Table 3. Major APIs.

API	Method	Purpose
/auth/login	POST	User authentication
/events	GET / POST	View and create events
/events/{id}	PUT / DELETE	Update / delete events
/events/{id}/register	POST	Event registration
/recommendations	GET	Get recommendations
/attendance/verify	POST	Verify QR attendance

4.4 AI-based event recommendation

The recommendation module aligns student interests with event information. It builds a TF-IDF matrix from event titles, descriptions, and categories and computes the cosine similarity between a user vector U and an event vector E :

$$\text{Similarity}(U, E) = \frac{U \cdot E}{\|U\| \|E\|} \quad (1)$$

A higher similarity value indicates that an event is more relevant to the user's interests, and events are ranked accordingly in the recommendation list.

4.5 Smart notification

The smart notification module informs students about relevant events. It considers a student's interests, the importance of an event, the student's registration status and event timing in order to reduce notifications and increase engagement, as summarized in Table 4.

Table 4. Notification parameters.

Parameter	Purpose
User interest	Checks event relevance
Event timing	Provides timely reminders
Registration	Prioritizes registered events
Notification history	Avoids repeated alerts

4.6 QR attendance and fraud prevention

A QR code is generated for each event. When the code is scanned, the system checks whether the individual is authenticated, registered for the event, holds a valid QR code and has not already attended the event. If any check fails, attendance is not marked as valid:

$$\text{Attendance} = \text{Auth} \wedge \text{Reg} \wedge \text{QRValid} \wedge \neg \text{Duplicate} \quad (2)$$

This condition limits attendance to authorized, registered individuals and prevents duplicate check-ins.

4.7 Personalized event calendar

The personalized event calendar helps students organize events they are interested in, registered for, or have been recommended. The calendar combines events the student has signed up for with events suggested through the AI recommendation module.

4.8 Algorithms

The two core procedures of the platform — generating recommendations and verifying QR-based attendance — are summarized in Algorithm 1 and Algorithm 2.

Algorithm 1. Event recommendation

```

1: Input: user interests, event list
2: Get user interests and event information.
3: Preprocess the text data.
4: Generate TF-IDF vectors.
5: Compute cosine similarity.
6: Sort events by similarity.
7: Present the highest-similarity events.
```

Algorithm 2. QR attendance

```

1: Input: user, event, QR code
2: Authenticate the user.
3: Validate the QR code and event.
4: Check whether the user is registered.
5: Check for a duplicate check-in.
```

```
6: if valid then
7: Record attendance.
8: else
9: Reject the request.
10: end if
```

5 Implementation

5.1 Frontend

The interface was built with React and consists of three role-based views—Student, Organizer, and Admin—corresponding to the user roles defined earlier. Each view is a functional component built with React Hooks, while data shared across components, such as the student’s name and points balance, is kept in a Redux store so that updates are reflected immediately throughout the app.

Styling uses Tailwind CSS with a mobile-first approach, since the check-in and ticket screens are used on mobile devices. Form inputs are validated on the client using the same schema library (Zod) as the server, catching input errors before they are submitted. The Organizer Console subscribes to a Socket.IO channel to display live scan data on a Chart.js dashboard without polling, while the Student Dashboard renders a QR ticket from a token returned by the API and provides a searchable event catalog and a live rewards ledger. Client-side routing keeps navigation on the browser: the landing and sign-up pages are public, while the student, coordinator, and faculty dashboards described above are role-specific.

5.2 Backend

The backend is built with Node.js and Express. QR-scan logic is kept separate from the HTTP layer, which simplifies backend testing. A JWT-based authentication middleware verifies each secured request and identifies the calling user, and access to sensitive endpoints—such as creating events—is restricted to specific roles. Incoming requests are further validated with express-validator, and the Helmet library is applied across the application. A rate limiter protects the check-in endpoint from being overloaded by a malfunctioning scanner. Because Socket.IO runs on the server as Express, the application can broadcast real-time updates, such as a newly recorded attendance entry, to connected clients.

5.3 Database

Mongoose is used to persist data in MongoDB across five collections: Users, Events, Registrations, Attendance Logs and Rewards. Mongoose schemas enforce field-level rules such as required fields, allowed roles and ObjectId references, preventing invalid data from being saved even if API-level validation is bypassed.

Two indexes are particularly important for performance: a unique index on each registration’s QR token enables constant-time lookup during scanning, and a compound index on the user–event pair prevents duplicate registrations at the database level. Live analytics are computed using MongoDB aggregation pipelines that summarize Attendance Log entries by event or time bucket, keeping the dashboard data current. The database runs on a managed, replicated MongoDB deployment with a measured uptime of 99.7%.

5.4 AI module

The AI module implements the recommendation functionality described in Section 4.4 as a separate Node.js service, allowing it to be developed and deployed independently of the core registration and attendance path. Each event is represented as a feature vector over category, department, and time-of-day, and each student as an interaction vector built from registration and attendance history; cosine similarity between these vectors ranks a shortlist of suggested events. A content-based approach was chosen over collaborative filtering because it remains effective for new students with little history, avoiding the cold-start problem. Full production rollout of this module remains future work.

5.5 Notification module

Notifications are delivered through two channels. In-app alerts are pushed over the same Socket.IO connection used by the dashboard, covering registration confirmations, successful scans, and reward updates, so students see confirmation instantly. Email notifications—mainly QR tickets and check-in reminders—are sent through Nodemailer over SMTP from a queued job, so a slow mail provider cannot block the registration response. Messages are generated from a small set of templates for consistency. Because unreliable university email was a limitation observed during the pilot, the module uses a pluggable transport so that SMS or in-app-only delivery can be added later without changing the calling code.

5.6 QR module

When a registration succeeds, the backend generates a single-use UUID v4 token, stores it on the registration record together with the event's check-in window, and produces a QR image (using the `qrcode` package) for the confirmation email and in-app display; the client can also render the same token as a lightweight ticket view. At the event, the Organizer Console uses a camera-based reader (`html5-qrcode`) to scan the code and sends the decoded token to the scan endpoint, which checks the token against the Registrations collection, confirms that the check-in window is open and verifies that the token has not already been used. Only if all checks pass is an Attendance Log entry written, the corresponding reward added, and a Socket.IO update pushed—all within a transaction. If any check fails, the system falls back to manual verification and no attendance is logged.

6 Conclusion and future work

This paper presented the Campus Event Aggregator, an integrated platform that combines AI-based event recommendation, smart notifications, secure QR-based attendance, and a personalized event calendar for college communities. Building on a previously developed event-discovery and check-in system, the proposed architecture adds a content-based recommendation module, a rate-limited and role-secured backend, and real-time dashboards, implemented on a MERN stack with Socket.IO. Together, these components address the fragmentation, weak personalization, and attendance-integrity issues identified in existing campus event systems.

Future work includes completing production rollout of the AI recommendation service, extending the notification module with pluggable SMS delivery, incorporating collaborative-filtering signals once sufficient interaction data is available, and conducting a formal user study to evaluate the system's impact on student engagement and attendance accuracy.

References

- [1] Balasubramaniyan C, Manivannan D. IoT Enabled Air Quality Monitoring System (AQMS) using Raspberry Pi. *Indian Journal of Science and Technology*. 2016;9(39). <https://doi.org/10.17485/ijst/2016/v9i39/90414>
- [2] Adeagbo AA. *IoT Based Environment Monitoring System Using ESP32* [Technical report]. Southern Illinois University Edwardsville; 2022.
- [3] Omkar N, Rahul S, Vijaykrishna R, Swapna N, Sura R. Smart environmental monitoring using ESP32 microcontroller. *IOSR Journal of Electronics and Communication Engineering*. 2024;19(3):7–12. <https://doi.org/10.9790/2834-1903010712>
- [4] Uddin MS, Praharaj MK. IoT based weather monitoring system using ESP32 microcontroller. *International Journal of Advanced Research in Science, Communication and Technology*. 2025;5(2):693–701. <https://doi.org/10.48175/IJAR SCT-29283>
- [5] Espressif Systems. *ESP32 Series Datasheet*. Version 3.8. Espressif Systems; 2021.
- [6] Aosong Electronics. *AM2302/DHT22 Digital Temperature and Humidity Sensor Datasheet*. Available from: <https://www.aosong.com/>
- [7] u-blox. *NEO-6 u-blox 6 GPS Modules Data Sheet*. Available from: <https://www.u-blox.com/>
- [8] Google. *Gemini API Reference*. Google AI for Developers. Available from: <https://ai.google.dev/api>
- [9] Google. *Flutter Documentation*. Available from: <https://docs.flutter.dev/>
- [10] flutter_bluetooth_serial_plus [Software package]. *Dart Packages*. Available from: https://pub.dev/packages/flutter_bluetooth_serial_plus
- [11] World Health Organization. *WHO Global Air Quality Guidelines: Particulate Matter (PM_{2.5} and PM₁₀), Ozone, Nitrogen Dioxide, Sulfur Dioxide and Carbon Monoxide*. Geneva, Switzerland: World Health Organization; 2021.
- [12] Jo JH, Jo B, Kim JH, Kim S, Han W. Development of IoT based indoor air quality monitoring platform. *Journal of Sensors*. 2020;2020:8749764. <https://doi.org/10.1155/2020/8749764>
- [13] de Medeiros HPL, Girao G. An IoT-based air quality monitoring platform. In: *Proc. 2020 IEEE International Smart Cities Conference (ISC2)*; 2020. <https://doi.org/10.1109/ISC251055.2020.9239070>
- [14] Jo JH, Jo B, Kim JH, Choi I. Development of IoT-based air quality monitoring system for the investigation of particulate matter (PM₁₀) in subway tunnels. *International Journal of Environmental Research and Public Health*. 2020;17(15):5429. <https://doi.org/10.3390/ijerph17155429>
- [15] Muhammad K, Ramadhan Putra RR, Purnomo E. Design and implementation of microcontroller-based air filter system with android application interface. *Emitor: Jurnal Teknik Elektro*. 2026;26(1):71–81. <https://doi.org/10.23917/emitor.v26i1.16143>