# CampusConnect - Event Management API

## 1. Project Overview

CampusConnect is a campus event management prototype that allows admins to create events, students to register and check in, collect feedback, and generate summary reports.

## 2. Assumptions & Edge Cases

- Each event belongs to one college.

- A student cannot register for the same event twice.

- Attendance can be marked only for registered students.

- Feedback is allowed only after attendance is marked.

- Scalability: should support thousands of students/events, concurrent registrations, and scale to a cloud database (e.g., PostgreSQL/MySQL).

## 3. Data to Track

- Events (title, type, date, college).

- Students (name, email, college).

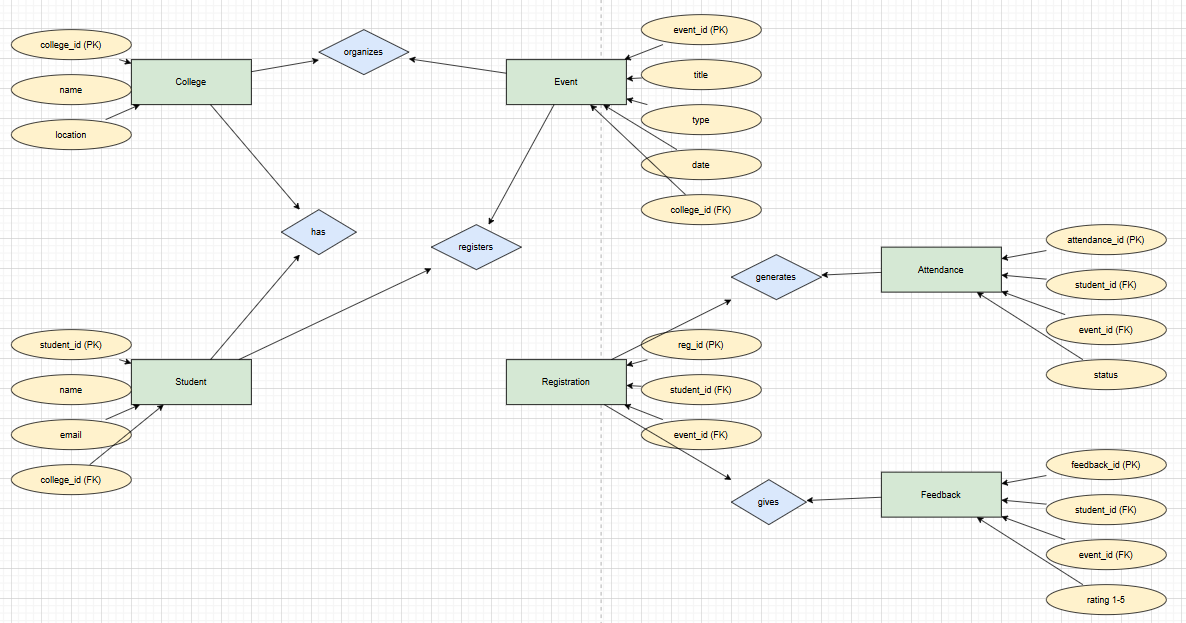
- Registrations (student <-> event).

- Attendance (present/absent).

- Feedback (rating 1-5).

## 4. Database Design

Entity Relationship Diagram (ERD):



SQL Schema (DDL):

CREATE TABLE colleges (...);  
CREATE TABLE students (...);  
CREATE TABLE events (...);  
CREATE TABLE registrations (...);  
CREATE TABLE attendance (...);  
CREATE TABLE feedback (...);

## 5. API Design (Endpoints)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Endpoint | Method | Description | Request Body | Response |
| / | GET | Welcome message | None | {"message":"Welcome"} |
| /events | POST | Create new event | {"title","type","date","college\_id"} | {"message":"Event created","event":{...}} |
| /register | POST | Register student | {"student\_id","event\_id"} | {"message":"Student registered"} |
| /attendance | POST | Mark attendance | {"student\_id","event\_id","status"} | {"message":"Attendance marked"} |
| /feedback | POST | Submit feedback | {"student\_id","event\_id","rating"} | {"message":"Feedback submitted"} |
| /reports/events | GET | Event popularity | None | [{"title":"TechFest 2025","registrations":120}] |
| /reports/students | GET | Student participation | None | [{"name":"Alice","events\_attended":3}] |
| /reports/top-students | GET | Top 3 active students | None | [{"name":"Maria","events\_attended":4}] |

## 6. Workflows

Admin: Create Event -> View Reports

Student: Register -> Attend -> Feedback -> Included in Reports

## 7. Report Queries (SQL)

SELECT e.title, COUNT(r.student\_id) AS registrations FROM events e LEFT JOIN registrations r ON e.event\_id = r.event\_id GROUP BY e.event\_id ORDER BY registrations DESC;

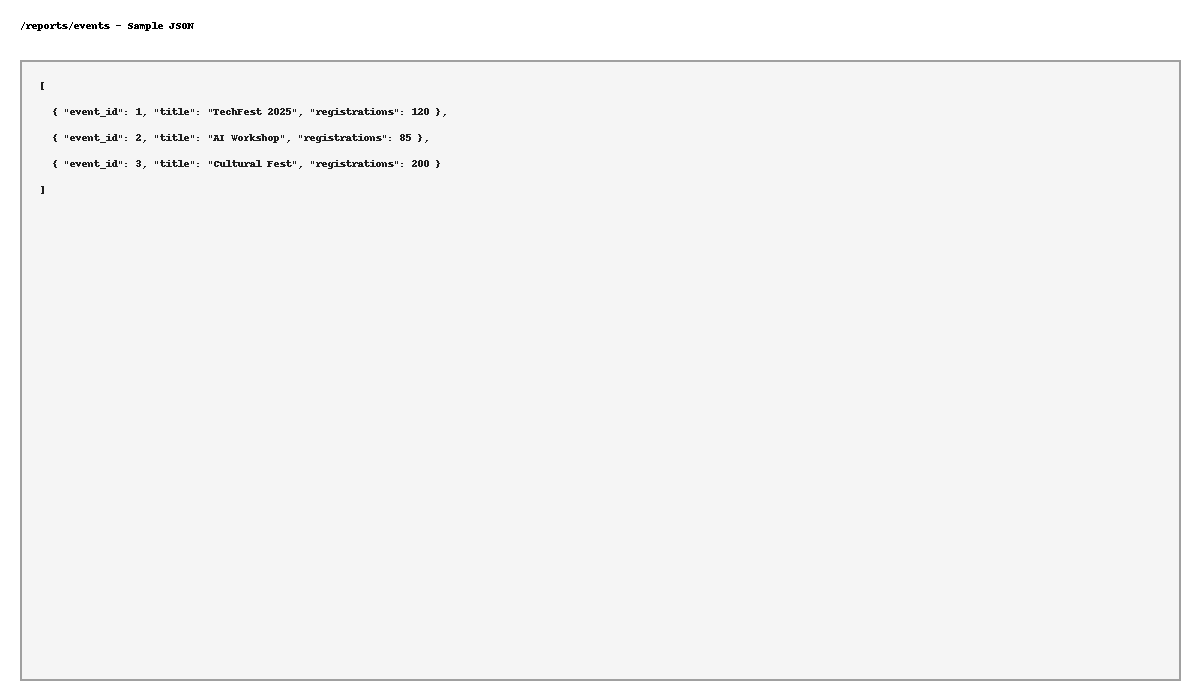
SELECT s.name, COUNT(a.event\_id) AS events\_attended FROM students s JOIN attendance a ON s.student\_id = a.student\_id WHERE a.status = 1 GROUP BY s.student\_id;

SELECT s.name, COUNT(a.event\_id) AS events\_attended FROM students s JOIN attendance a ON s.student\_id = a.student\_id WHERE a.status = 1 GROUP BY s.student\_id ORDER BY events\_attended DESC LIMIT 3;

## 8. Sample Outputs (JSON)

Event Popularity (/reports/events):

[  
 { "event\_id": 1, "title": "TechFest 2025", "registrations": 120 },  
 { "event\_id": 2, "title": "AI Workshop", "registrations": 85 },  
 { "event\_id": 3, "title": "Cultural Fest", "registrations": 200 }  
]



Student Participation (/reports/students):

[  
 { "student\_id": 101, "name": "Alice Johnson", "events\_attended": 3 },  
 { "student\_id": 102, "name": "Ravi Sharma", "events\_attended": 2 },  
 { "student\_id": 103, "name": "Maria Lopez", "events\_attended": 4 }  
]



Top Students (/reports/top-students):

[  
 { "student\_id": 103, "name": "Maria Lopez", "events\_attended": 4 },  
 { "student\_id": 101, "name": "Alice Johnson", "events\_attended": 3 },  
 { "student\_id": 104, "name": "James Smith", "events\_attended": 3 }  
]



## 9. UI Mockups (Wireframes)

Admin Portal:



Student App:



AI Conversation screenshots:

