Campus Event Management Platform – Design Document

1. Introduction & Assumptions

This project implements the backend for a Campus Event Management Platform to support event creation, student registration, attendance tracking, and feedback collection, along with event reporting. The solution assumes:

- Each event has a unique ID across all colleges.
- Students can only register once per event.
- Attendance is marked only for registered students, and feedback is only accepted from those marked as attended.
- Duplicate registrations, attendance entries, and feedback are prevented using constraints.
- Data for all colleges is kept in a single schema for unified reporting.
- System scales to \sim 50 colleges, each with 500 students and up to 20 events per semester.

2. Data to Track

• Colleges: id, name

• Students: id, name, college_id

• Events: id, name, type, date, college id

• Registrations: id, student id, event id (unique pair)

• Attendance: id, student_id, event_id, status (unique pair)

• Feedback: id, student id, event id, rating (1–5, unique pair)

3. Database Schema

Table Structure

Table	Columns	Constraints
colleges	id [PK], name	Unique id
students	id [PK], name, college_id [FK colleges(id)]	FK, Unique id

Table	Columns	Constraints	
	id [PK], name, type, date, college_id [FK colleges]	FK, Unique id	
registrations		Each pair registered only once	
llaffendance		Only one attendance per event per student	
liteedhack	id [PK], student_id [FK], event_id [FK], rating (1–5), unique(student_id, event_id)	Only one feedback per event per student	

4. API Design

Method	Endpoint	Description	Request Body Example	Success Response
POST	/api/events	Create new event	{ "name": "Hackathon", "type": "Workshop", "date": "", "college_id": 1 }	Event JSON
GET	/api/events	List all events	_	[Event]
POST	/api/registrations	Register student to event	{ "student_id": 2, "event_id": 5 }	Registration JSON
GET	/api/registrations	List all registrations	_	[Registration]
POST	/api/attendance	Mark attendance for student/event	{ "student_id": 2, "event_id": 5, "status": true }	Attendance JSON
GET	/api/attendance	List all attendance		[Attendance]

Method	Endpoint	Description	Request Body Example	Success Response
POST	/api/feedback	Submit feedback for event	{ "student_id": 2, "event_id": 5, "rating": 4 }	Feedback JSON
GET	/api/feedback	List all feedback entries		[Feedback]
GET	/api/reports/popularity	Event popularity report		[Event + count]
GET	/api/reports/participation	Student participation report		[Student + count]
GET	/api/reports/top-students	Top 3 most active students		[Student + count]
GET	/api/reports/by-type	Events filtered by type	?type=Seminar	[Event]

5. Workflows

Registration:

Student browses events \rightarrow Registers for event \rightarrow Entry in registrations table.

Attendance:

On event day, admin marks attendance \rightarrow Only registered student can be marked present/absent \rightarrow Entry in attendance table.

Feedback:

After event, student (if attended) can submit feedback $(1-5) \rightarrow$ Feedback saved or updated.

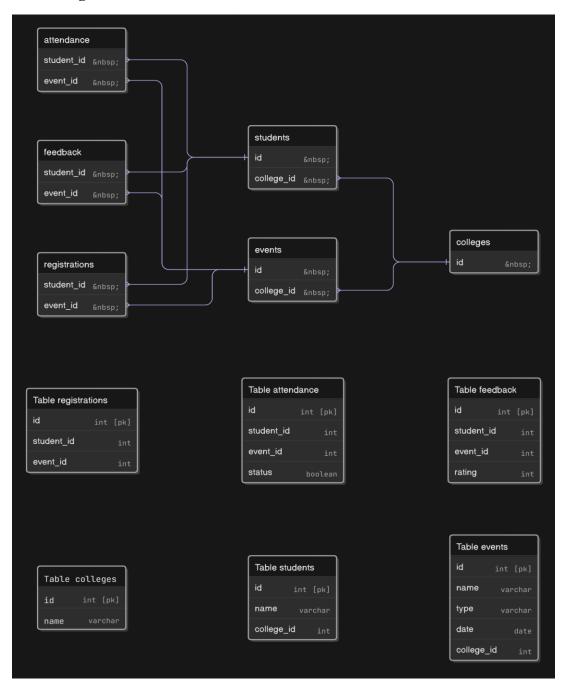
Reporting:

Staff fetches reports (popularity, participation, top students, etc.) using the corresponding endpoints.

6. Assumptions & Edge Cases

- Duplicate registrations, attendance, or feedback are blocked by unique constraints.
- Feedback is only possible if attendance is marked present.
- Attempts to register or submit feedback for invalid events/students return clear errors.
- Events and students must exist before relations are created.
- The same approach can be scaled for more colleges and events.

7. ER Diagram



8. API Examples

API Endpoints

Event Management

• Create Event:

POST /api/events

Create a new event.

• List All Events:

GET /api/events *Fetch all events*.

Student Registration

• Register Student to Event:

POST /api/registrations
Register a student for an event.

• List All Registrations:

GET /api/registrations *View all registrations.*

Attendance

• Mark Attendance:

POST /api/attendance Mark or update a student's attendance for an event.

• List All Attendance Records:

GET /api/attendance
Get all attendance entries.

Feedback

• Submit Feedback:

POST /api/feedback

Student submits or updates feedback for an attended event.

• List All Feedback Entries:

GET /api/feedback

View all feedback provided by students.

Reporting

• Event Popularity:

GET /api/reports/popularity

Events sorted by registration count.

• Student Participation:

GET /api/reports/participation

Number of events attended by each student.

• Top 3 Most Active Students:

GET /api/reports/top-students

Top three students ranked by event attendance.

• Events Filtered by Type:

GET /api/reports/by-type?type=XXX List of events filtered by event type (Workshop, Fest, Seminar, etc.).