

## 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

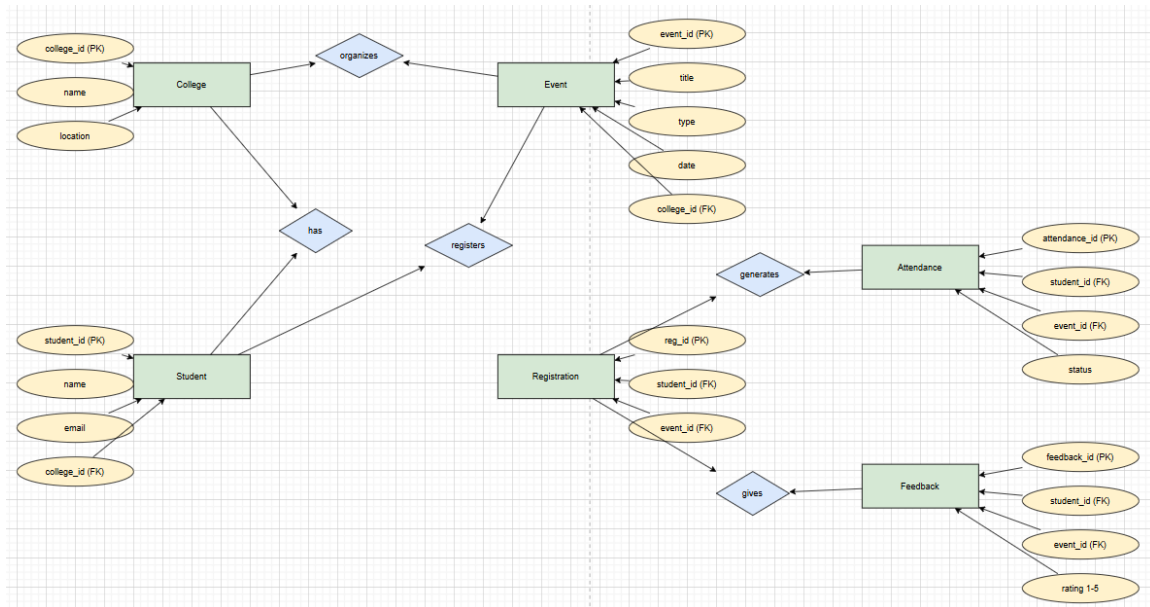
- Each event belongs to one college.
- A student cannot register for the same event twice.
- Attendance can be marked only for registered students.
- Feedback is allowed only after attendance is marked.
- Scalability: should support thousands of students/events, concurrent registrations, and scale to 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 }
]
```

/reports/events - Sample JSON

```
[
  { "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 }
]
```

/reports/students - Sample JSON

```
[
  { "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 }
]
```

/reports/top-students - Sample JSON

```
[
  { "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:

Admin Portal - Dashboard & Reports

Navbar: Dashboard | Events | Reports | Settings

Create Event Form: Title, Type, Date, College

Events List: Title | Type | Date | Registrations

Reports: Event Popularity | Student Participation | Top Students

Student App:

Student App - Browse & Register

Home: Event Cards (Title, Date, Type)

Event Details: Register Button + Info

My Events: Check-in (Attendance)

Feedback: 1-5 rating after attendance

AI Conversation screenshots:


Campus Drive Assignment – Webknot ...  
PDF

Give me a draft on how each and everything is to be done

Great — I've read the full assignment brief you uploaded 

Campus Drive Assignment – Webkn...



. Since you asked for a **draft on how each and everything is to be done**, here's a structured plan you can follow step by step:

 **Draft Plan for Campus Drive Assignment – Webknot Technologies**

1. Document Your Approach

- Assumptions
  - Each college has multiple events.
  - Event IDs should be unique **per college**, not globally.
  - Students can register for multiple events but cannot register twice for the same one.
  - Attendance can only be marked for registered students.
  - Feedback is optional, but only accepted if attendance is marked.
- AI Support
  - Use ChatGPT (me), Claude, Gemini, or others to brainstorm schema, APIs, and workflows.
  - Keep screenshots of conversations.
  - Note where you followed AI's suggestions and where you deviated (e.g., schema changes for

+ Ask anything



ChatGPT can make mistakes. Check important info. See [Cookie Preferences](#).

### 3. Prototype Implementation

- **Tech Choices**
  - Backend: Python (Flask/FastAPI) or Node.js (Express).
  - DB: SQLite for simplicity.
- **Core Features to Implement**
  - `register_student(event_id, student_id)`
  - `mark_attendance(event_id, student_id)`
  - `collect_feedback(event_id, student_id, rating)`
  - **Queries:**
    - Registrations per event.
    - Attendance % per event.
    - Avg. feedback score per event.
- **Setup**
  - Provide `README.md` with steps:

```
bash

git clone <repo>
cd project
pip install -r requirements.txt
python app.py
```

 Copy code



### 4. Reports

+ Ask anything



#### 4. Reports

- **Event Popularity Report** → `SELECT event_id, COUNT(student_id) AS registrations FROM Registration GROUP BY event_id ORDER BY registrations DESC;`
- **Student Participation Report** → `SELECT student_id, COUNT(event_id) AS attended FROM Attendance WHERE status=1 GROUP BY student_id;`
- **Top 3 Active Students** → Above query + `ORDER BY attended DESC LIMIT 3;`
- **Flexible Reports** → Add `WHERE event_type='Workshop'` filter.

### 5. Bonus

- Mockup UI (Figma or simple HTML/CSS wireframe).
- Dropdown filter for event type in reports.

### 6. Deliverables

- **AI Logs** → Screenshots of brainstorming.
- **Design Document** → PDF with schema, ERD, sequence diagram.
- **Prototype Code** → Clean, minimal APIs + database.
- **Reports/Outputs** → SQL query outputs or screenshots.
- **README.md** → Personal write-up (not AI-generated).

