1. Overview

A minimal **Campus Event Reporting** backend using **Flask + MySQL** to track events, student registrations, attendance, and feedback, and to generate simple reports (popularity, participation, attendance%, avg feedback)

2. Assumptions

- Unique event_id across system.
- One feedback per student per event.
- No duplicate registrations for the same student & event.
- Attendance marked per (student,event) as Present/Absent.
- Ratings are integers 1–5

3. Data Model (MySQL)

Students

- student_id INT PK AUTO_INCREMENT
- name VARCHAR(100)
- email VARCHAR(100) UNIQUE
- college_id INT

Events

- event_id INT PK AUTO_INCREMENT
- event_name VARCHAR(100)
- event_type ENUM('Workshop','Seminar','Hackathon','Fest')
- event_date DATE
- college_id INT

Registrations

- reg_id INT PK AUTO_INCREMENT
- student_id INT FK → Students
- event_id INT FK → Events
- UNIQUE(student_id, event_id)

Attendance

- attendance_id INT PK AUTO_INCREMENT
- student_id INT FK
- event_id INT FK

- status ENUM('Present','Absent')
- UNIQUE(student_id, event_id)

Feedback

- feedback_id INT PK AUTO_INCREMENT
- student_id INT FK
- event_id INT FK
- rating INT CHECK 1–5
- comments TEXT
- UNIQUE(student_id, event_id).

4. APIs (JSON)

Base: http://127.0.0.1:5000

POST /students:

{ "name":"Rahul", "email":"rahul@example.com", "college_id":101 }

POST /events:

{ "event_name":"Hackathon 2025", "event_type":"Hackathon", "event_date":"2025-09-15", "college_id":101 }

POST /register:

{ "student_id":1, "event_id":1 }

POST /attendance:

{ "student_id":1, "event_id":1, "status":"Present" }

POST /feedback:

{ "student_id":1, "event_id":1, "rating":5, "comments":"Great!" }

GET /reports/popularity → registrations per event
GET /reports/participation → events attended per student
GET /reports/attendance → attendance% per event
GET /reports/feedback → average feedback per event
GET /reports/top-students → top 3 by events attended

5. Core Workflows

1. **Register**: student → /register → row in Registrations.
2. **Attendance**: staff marks /attendance.
3. **Feedback**: student posts /feedback.
4. **Reports**: GET endpoints aggregate from tables above.

7. Edge Cases / Validation

- Prevent duplicate registrations with UNIQUE(student_id,event_id).
- NULLIF(...,0) avoids divide-by-zero in attendance%.
- Feedback rating clamped to 1–5 via CHECK (or validate in API).

8. Security & Simplicity Choices

- No auth (kept intentionally simple for prototype).
- Parameterized queries used via mysql-connector.
- Single file app.py to keep it easy to explain.

9. Limitations & Future Work

- Add login/roles (admin/student).
- Pagination & filters on reports.
- Simple UI for event browsing and registration.