Design Document — Campus Event Management (Reporting Prototype)

Author: <Your Name> Date: 2025-09-06

Stack: React (frontend) + Flask (backend) + MySQL (database)

Overview & Goals

Minimal prototype for campus event reporting: Admin portal (create/cancel events, check-in, reports) and Student app (browse, register, feedback). Demonstrates data model, APIs, workflows and report generation.

Assumptions & Decisions

Multi-college supported via college_id. Numeric auto-increment IDs used for simplicity. Admin actions protected by X-Admin-Token. Prototype auto-registers walk-ins during check-in. CORS enabled for local dev.

Data Tracked (key tables)

colleges(id,name,domain); students(id,college_id,student_uid,name,email);
events(id,college_id,title,event_type,starts_at,ends_at,location,status);
registrations(id,event_id,student_id,registered_at);
attendances(id,event_id,student_id,checkin_at,method);
feedback(id,event_id,student_id,rating,comments,submitted_at).

API (selected endpoints)

GET /events?college_id=1 — list events
GET /events/{id} — event details
POST /events/{id}/register — register student (student_uid + college_id)
POST /events/{id}/attendance — mark attendance (auto-creates registration for walk-ins)
POST /events/{id}/feedback — submit feedback (rating 1-5)
GET /reports/event_popularity?college_id=1 — registrations per event
GET /reports/student_participation?college_id=1 — events attended per student

Workflows (brief)

1) Student registers for event → backend creates student and registration.
2) On event day, staff marks attendance (by UID & event id); backend records attendance and may auto-register.
3) Post-event, student submits feedback; reports aggregate registrations, attendance %, average rating, and top active students.

Reporting Queries (examples)

Event popularity: COUNT(registrations) per event; Attendance %: attendees / registrations * 100; Avg feedback: AVG(feedback.rating) per event; Top students: COUNT(attendances) grouped by student, ordered desc.

Run Notes

Backend: activate venv, pip install -r requirements.txt, pip install flask-cors, create DB and run py run.py.
Frontend: npm install, npm start. API base URL: http://localhost:5000/api/v1. Default demo college_id = 1.

AI Usage

AI was used sparingly (~20%) for debugging runtime issues (missing packages, CORS, network errors) and for discussing auto-registration behavior. Core design, code, UI and documentation are authored by me.

Design document (condensed) — add details or diagrams in the final PDF if needed.