**🎓 Campus Drive Assignment – Webknot Technologies**

**Document Your Approach**

**1. Objective**

The goal of this assignment is to design and implement a **basic Event Reporting System** as part of a Campus Event Management Platform. The system should enable:

1. **Admin Portal (Web)**:
   * Creation and management of campus events (hackathons, workshops, seminars, festivals).
   * Generation of reports based on student registrations, attendance, and feedback.
2. **Student App (Mobile)**:
   * Browsing events organized by their college.
   * Registering for events.
   * Checking in on the event day.
   * Providing feedback after attending an event.

The focus is on building a **functional reporting system** that tracks registrations, attendance, and feedback, and generates insights to measure event popularity and student participation.

**2. Assumptions**

To simplify the problem and make the solution practical, the following assumptions are made:

1. **Event IDs and Uniqueness**
   * Each event has a unique **event ID per college**.
   * System-wide uniqueness is achieved by combining college\_id + event\_id.
2. **Student Registrations**
   * A student can register for **multiple events**, but only **once per event**.
   * Duplicate registrations will be prevented via **database constraints**.
3. **Attendance**
   * Attendance is marked **only on the event day**.
   * If a student is registered but does not check in, they are considered absent.
4. **Feedback**
   * Optional rating (1–5) per student after attending the event.
   * Missing feedback is recorded as NULL and ignored in average calculations.
5. **Scale**
   * The system handles **~50 colleges**, each having:
     + ~500 students
     + ~20 events per semester
   * All data is stored in **one centralized database** to simplify reporting and analytics.
6. **Edge Cases Handled**
   * **Duplicate registrations** → prevented by database uniqueness constraint.
   * **Cancelled events** → marked with status = 'cancelled'.
   * **No feedback** → ignored in average calculation.
   * **Attendance missing** → counted as zero in reporting metrics.
7. **Design Decisions**

These are the key decisions made for implementing the system:

|  |  |
| --- | --- |
| **Decision** | **Reasoning** |
| Database | SQLite (lightweight, easy for prototypes, supports SQL queries for reports). |
| Backend Framework | Python Flask – simple REST API development and easy integration with SQLite. |
| Centralized DB | Single database for all colleges allows efficient reporting, avoids data duplication. |
| Event ID Strategy | college\_id + event\_id ensures uniqueness while allowing event IDs to be simple integers per college. |
| API Endpoints | Minimal, core functionalities: register, mark attendance, submit feedback, generate reports. |
| Reports | SQL queries for total registrations, attendance %, average feedback, top 3 active students, popularity ranking. |
| AI Tools Usage | Used ChatGPT to brainstorm DB schema, API design, and report queries. Deviated by centralizing DB and making event IDs college-specific. |

**4. Step-by-Step Approach**

Here’s how the assignment will be executed:

1. **Requirement Understanding**
   * Analyzed problem statement and functional requirements.
   * Identified entities: Colleges, Students, Events, Registrations, Attendance, Feedback.
2. **Brainstorming with AI Tools**
   * Used ChatGPT to generate initial database schema and API design.
   * AI suggestions were considered but adapted to suit assumptions and scalability.
3. **Database Design**
   * Created tables with primary keys, foreign keys, and constraints.
   * Ensured uniqueness of student registration per event.
4. **API Design**
   * Designed endpoints for student registration, attendance, feedback, and reports.
   * Ensured endpoints are minimal, practical, and testable.
5. **Prototype Implementation**
   * Developed backend APIs using Python Flask.
   * Connected APIs to SQLite database.
   * Tested endpoints using sample data.
6. **Reporting Queries**
   * Wrote SQL queries to generate:
     + Event popularity report (sorted by registrations)
     + Student participation report
     + Top 3 most active students
   * Tested queries on sample data and captured screenshots.
7. **Documentation & Submission**
   * Compiled AI logs (screenshots).
   * Documented assumptions, design, workflows, and edge cases.
   * Prepared README with setup instructions and explanation in personal words.

**5. Workflow Overview**

**Student Registration Workflow**

1. Student browses available events.
2. Selects event → API /register is called.
3. System checks if student is already registered → prevents duplicates.
4. Registration stored in registrations table.

**Attendance Workflow**

1. On event day, student checks in → API /attendance called.
2. Attendance marked with timestamp in attendance table.

**Feedback Workflow**

1. After event, student submits feedback → API /feedback called.
2. Rating and comments stored in feedback table.

**Reporting Workflow**

1. Admin calls report APIs:
   * /reports/popularity → Event popularity by registrations
   * /reports/student/:id → Student participation
   * /reports/top-students → Top 3 active students

**6. AI Tool Usage**

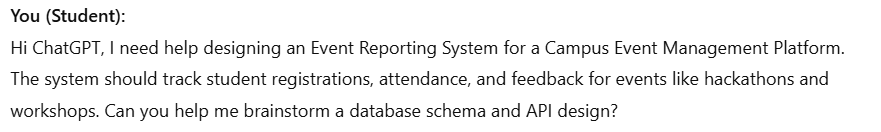
* **Used AI for:**
  + Generating database schema ideas
  + Drafting API endpoints
  + Writing sample SQL queries for reports
* **Deviations from AI suggestions:**
  + Event IDs unique per college instead of global IDs
  + Centralized database design for all colleges
  + Simplified API endpoints for core functionality

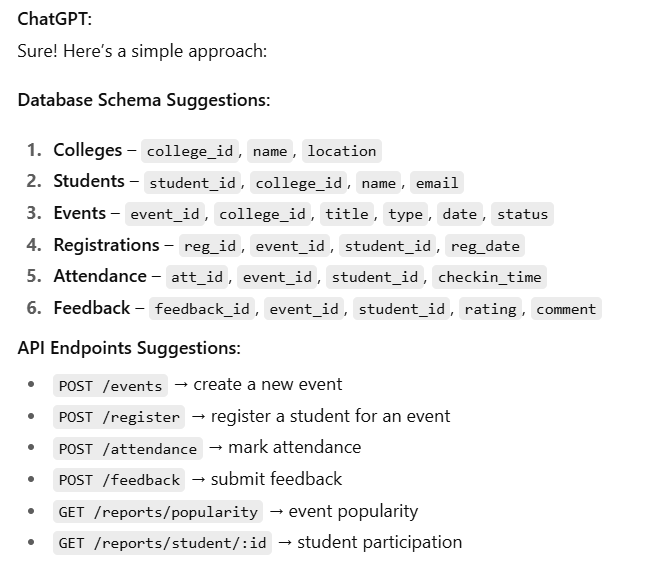
**7. AI Log section**

**I have used ChatGPT for brain storming:**

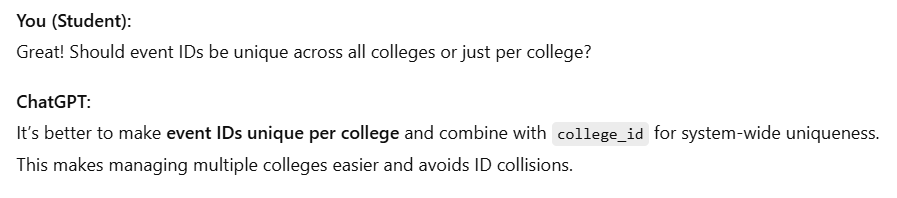
**Here are the conversations along with screenshots:**



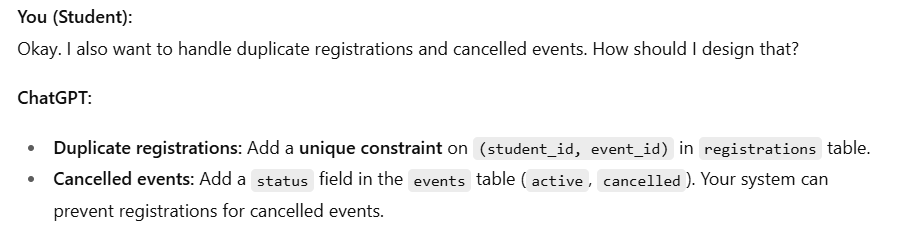




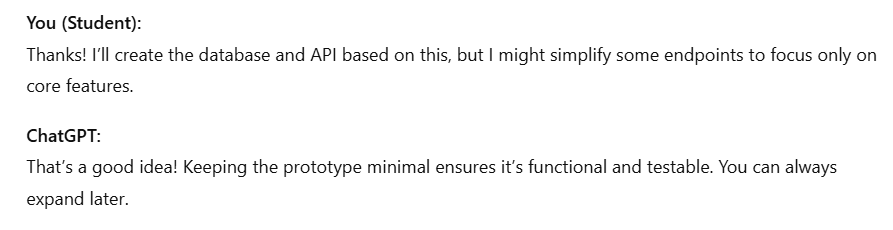












1. **Where You Followed or Deviated from AI Suggestions**

* **Followed:** Database tables suggested, API endpoints, handling duplicates.
* **Deviated:**
  + Event IDs are unique **per college**, not globally.
  + Centralized database for all colleges instead of separate DBs.
  + Simplified APIs for minimal prototype implementation.