**🎓 Campus Drive Assignment – Webknot Technologies**

**Design Document**

**1. Overview**

This document describes the design of a **basic Event Reporting System** for a Campus Event Management Platform. The system allows:

* **Admin Portal (Web)**:
  + Create and manage events (hackathons, workshops, seminars, fests).
  + View reports on event registrations, attendance, and feedback.
* **Student App (Mobile)**:
  + Browse events organized by their college.
  + Register for events.
  + Check-in on the event day.
  + Provide feedback after attending events.

The system will handle ~50 colleges, each with ~500 students and ~20 events per semester.

**2. Assumptions & Edge Cases**

**Assumptions**

1. Event IDs are **unique per college**; combined with college\_id for global uniqueness.
2. Students can register **once per event**, but for multiple events.
3. Attendance is **marked only on the event day**.
4. Feedback rating is **1–5**, optional. Missing feedback is NULL.
5. All colleges share a **centralized database** for easier reporting.

**Edge Cases**

|  |  |
| --- | --- |
| **Scenario** | **Solution** |
| Duplicate registration | Prevented via unique constraint on (student\_id, event\_id) |
| Cancelled events | status = 'cancelled' in events table; prevent registrations |
| No attendance | Counted as absent in reports |
| Missing feedback | Ignored in average rating calculations |
| Student deletion | Cascade deletes registrations, attendance, and feedback |

**3. Data to Track**

**The system will track the following entities and information:**

|  |  |  |
| --- | --- | --- |
| **Entity** | **Data to Track** | **Description** |
| **Colleges** | college\_id, name, location | College details |
| **Students** | student\_id, college\_id, name, email, year | Student info, linked to a college |
| **Events** | event\_id, college\_id, title, type, date, status | Event info; types include Hackathon, Workshop, Fest, Seminar; status = active/cancelled |
| **Registrations** | reg\_id, event\_id, student\_id, reg\_date | Tracks which students registered for which event |
| **Attendance** | att\_id, event\_id, student\_id, checkin\_time | Tracks actual student attendance |
| **Feedback** | feedback\_id, event\_id, student\_id, rating, comment | Student feedback on events (1–5 rating, optional comment) |

**4. Database Schema**

**Tables**

1. **Colleges**

|  |  |  |
| --- | --- | --- |
| **Column** | **Type** | **Constraint** |
| college\_id | INT | PK |
| name | TEXT | NOT NULL |
| location | TEXT |  |

1. **Students**

|  |  |  |
| --- | --- | --- |
| **Column** | **Type** | **Constraint** |
| student\_id | INT | PK |
| college\_id | INT | FK → colleges(college\_id) |
| name | TEXT | NOT NULL |
| email | TEXT | UNIQUE, NOT NULL |
| year | INT |  |

1. **Events**

|  |  |  |
| --- | --- | --- |
| **Column** | **Type** | **Constraint** |
| event\_id | INT | PK |
| college\_id | INT | FK → colleges(college\_id) |
| title | TEXT | NOT NULL |
| type | TEXT | Workshop/Fest/Seminar |
| date | DATE | NOT NULL |
| status | TEXT | active/cancelled |

1. **Registrations**

|  |  |  |
| --- | --- | --- |
| **Column** | **Type** | **Constraint** |
| reg\_id | INT | PK |
| event\_id | INT | FK → events(event\_id) |
| student\_id | INT | FK → students(student\_id) |
| reg\_date | DATE-TIME | DEFAULT CURRENT\_TIMESTAMP |
| UNIQUE(event\_id, student\_id) |  | Prevent duplicates |

1. **Attendance**

|  |  |  |
| --- | --- | --- |
| **Column** | **Type** | **Constraint** |
| att\_id | INT | PK |
| event\_id | INT | FK → events(event\_id) |
| student\_id | INT | FK → students(student\_id) |
| checkin\_time | DATE-TIME |  |

1. **Feedback**

|  |  |  |
| --- | --- | --- |
| **Column** | **Type** | **Constraint** |
| feedback\_id | INT | PK |
| event\_id | INT | FK → events(event\_id) |
| student\_id | INT | FK → students(student\_id) |
| rating | INT | 1–5 |
| comment | TEXT | Optional |

**ER Diagram**

 Colleges → Students (1:M)

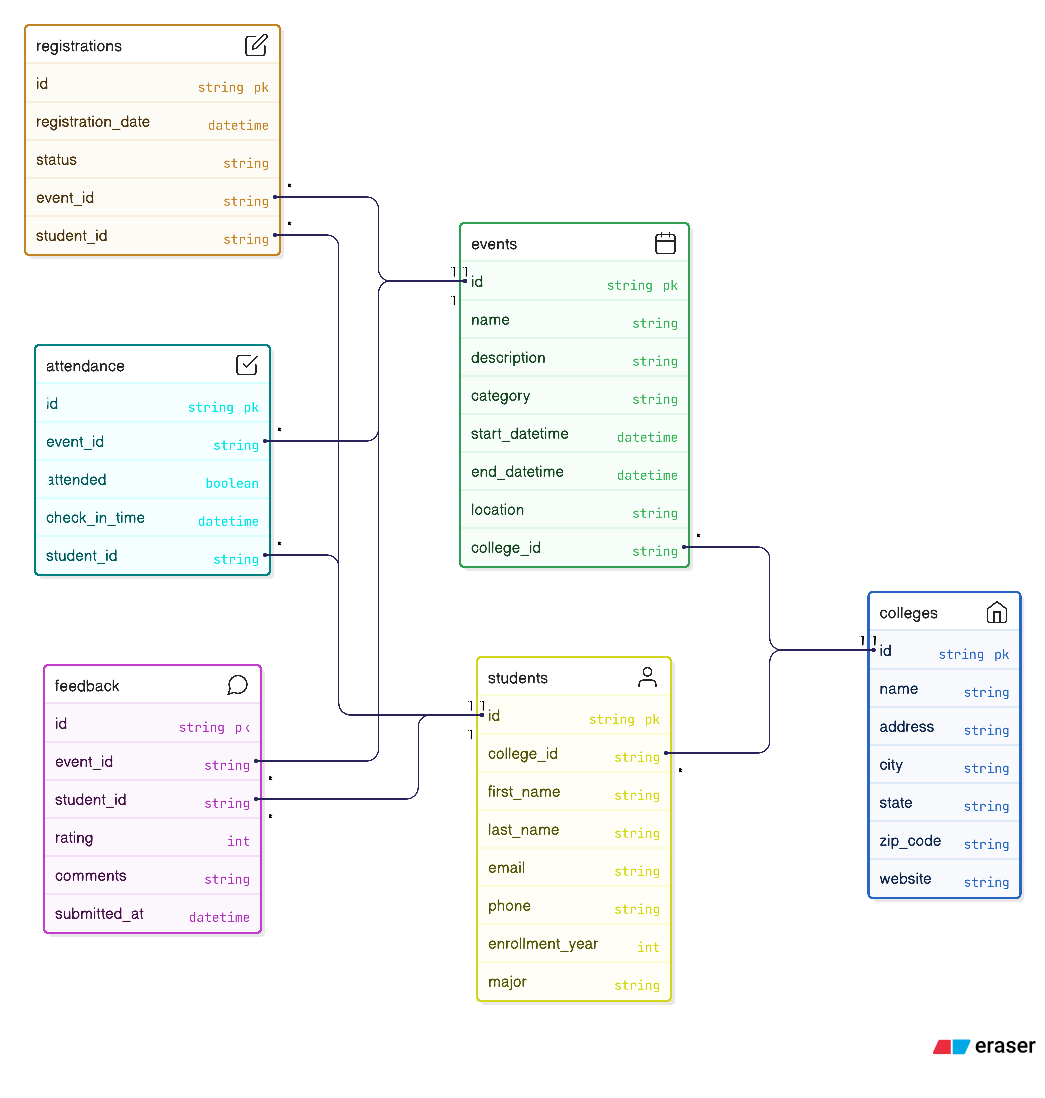
 Colleges → Events (1:M)

 Events → Registrations (1:M)

 Events → Attendance (1:M)

 Events → Feedback (1:M)

 Students → Registrations / Attendance / Feedback (1:M)



**5. API Design**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Method** | **Endpoint** | **Input** | **Output** | **Description** |
| POST | /events | title, type, date, college\_id | success/failure | Create event |
| GET | /events?college\_id= | college\_id | List of events | Get all events for a college |
| POST | /register | student\_id, event\_id | success/failure | Register student for event |
| POST | /attendance | student\_id, event\_id | success/failure | Mark attendance |
| POST | /feedback | student\_id, event\_id, rating, comment | success/failure | Submit feedback |
| GET | /reports/registrations/:event\_id | event\_id | Total registrations | Total registrations per event |
| GET | /reports/attendance/:event\_id | event\_id | Attendance % | Percentage of students present |
| GET | /reports/feedback/:event\_id | event\_id | Average rating | Average feedback score |
| GET | /reports/popularity | - | List of events | Sorted by total registrations |
| GET | /reports/student/:student\_id | student\_id | Events attended | Student participation report |
| GET | /reports/top-students | - | Top 3 students | Most active students by attendance |

**6. Workflows / Sequence Diagrams**

**6.1 Student Registration**

1. Student opens mobile app → views list of events.
2. Selects an event → calls /register API.
3. System checks for duplicate registration → inserts into registrations table.



**6.2 Attendance**

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



**6.3 Feedback Submission**

1. Student submits rating/comment → /feedback API called.
2. Feedback stored in feedback table.



**6.4 Reporting**

1. Admin requests report via web portal → calls /reports/... API.
2. Server executes SQL query → returns result.

