Event Management System - Documentation Report

# Document Your Approach

## Assumptions and Decisions

* + Each student can register for multiple events but only once per event.
  + Events are associated with a specific college.
  + Attendance can only be marked for registered students.
  + Feedback is optional and rated on a scale of 1–5.

## AI Tool Usage

* + Used ChatGPT to brainstorm database schema, API endpoints, and workflows.
  + Adapted suggestions to match project requirements, such as using SQLite for simplicity.

## AI Conversation Log

* + Chat logs from ChatGPT brainstorming were saved as screenshots and links.
  + Some suggestions were followed (ER diagram, sequence diagrams), others adapted (frontend interactivity, table designs).

# Design Document

## Data to Track

* + Event creation (name, type, date, description, college).
  + Student registration (student, event).
  + Attendance (present/absent).
  + Feedback (rating 1–5, comments).

## Database Schema (ER Diagram)

+ + + + + +

| College | | Event | | Student |

+ + + + + +

| id (PK) |<-----<| id (PK) | | |

| name | | name | | name |

+----------------+ | event\_type | | email |

| date | + +

| |

|college\_id (FK) | >+

+ + |

|

+ +

| Registration |

+ +

| id (PK) |

| student\_id (FK)|

| event\_id (FK) |

+ +

| |

| |

+ + + +

| |

+---------------+ + +

| Attendance | | Feedback |

+---------------+ + +

| id (PK) | | id (PK) |

| registration\_id (FK) | | registration\_id

(FK) |

## API Design

| status | | rating |

+---------------+ | comments |

+ +

* + POST /student/register\_event → Register student to an event.
  + POST /admin/mark\_attendance → Mark attendance for students.
  + POST /student/feedback → Submit feedback.
  + GET /admin/dashboard\_data → Fetch all data for admin dashboard.
  + GET /report/registrations → Get total registrations per event.
  + GET /report/attendance → Get attendance percentage.
  + GET /report/feedback → Get average feedback score.

## Workflows (Sequence Diagram)

Student WebApp / Frontend Backend / DB

| | |

|---Request Events-->| |

| |---Fetch Events >|

| |<-----Events |

|<---Display Events| |

|---Register Event--> | |

| |---Create Registration->|

| |<-----Success |

Admin/Teacher | |

|---Mark Attendance->| |

| |---Update Attendance-->|

| |<-----Success |

Student | |

|---Submit Feedback->| |

| |---Store Feedback >|

| |<-----Success |

## Assumptions & Edge Cases

* + Duplicate registrations are prevented.
  + Cancelled events remove pending registrations.
  + Feedback is optional and defaults to null if not provided.

# Prototype Implementation

## Backend: Flask (Python)

* + Database: SQLite
  + Models: College, Event, Student, Registration, Attendance, Feedback
  + Example endpoint: /admin/dashboard\_data returns JSON for frontend.

## Frontend: HTML + CSS + JS

* + Dynamic tables for colleges, events, students, registrations, top 3 active students.
  + Interactive buttons and hover effects.
  + Fetch API used to load data from backend.

*<!-- Example:Dashboard -->*

<**table** id="registrations-event-table">

<**caption**>Registrations per Event</**caption**>

<**thead**>

<**tr**>

<**th**>Event ID</**th**>

<**th**>Event Name</**th**>

<**th**>Registrations Count</**th**>

</**tr**>

</**thead**>

<**tbody**></**tbody**>

</**table**>

*// Fetch dashboard data*

**async function** loadDashboard() {

**const** response = **await** fetch("/admin/dashboard\_data");

**const** data = **await** response.json();

*// Populate tables dynamically*

**const** tableBody = document.querySelector("#registrations-event-table tbod y");

data.registrations\_per\_event.forEach(ev **=>** {

**const** row = document.createElement("tr");

row.innerHTML = `<td>${ev.event\_id}</td><td>${ev.event\_name}</td><td>

${ev.count}</td>`;

tableBody.appendChild(row);

});

}

loadDashboard();

## Sample Queries / Reports

* + Total registrations per event: SELECT event\_id, COUNT(\*) FROM registration GROUP BY event\_id;
  + Attendance percentage: SELECT student\_id, (SUM(CASE WHEN status='Present' THEN 1 ELSE 0 END)/COUNT(\*))\*100 FROM attendance GROUP BY student\_id;
  + Average feedback score: SELECT event\_id, AVG(rating) FROM feedback GROUP BY event\_id;

Screenshots of LLM conversations:





