Campus Event Management System – Design Document

1. Project Overview

- **Purpose:** Manage college events including student registrations, attendance, feedback, and reports.
- **Scope:** Backend prototype only (APIs and reports).
- **Target Users:** College admins and students.

2. Functional Requirements

- Register students to events.
- Mark attendance for registered students.
- Collect feedback (rating 1–5).
- Generate reports:

a. Total registrations per event

b. Attendance percentage per event

c. Average feedback score per event

d. Event popularity (sorted by registrations)

e. Student participation (events attended)

3. Non-Functional Requirements

- **Performance:** Quick response for CRUD operations.
- **Scalability:** Can switch DB to MySQL/Postgres.
- **Maintainability:** Modular service, controller, repository layers.
- **Usability:** APIs can be tested via Postman.

4. Technology Stack

- **Backend:** Java Spring Boot 3.5.3
- **Database:** H2 (switchable to MySQL/Postgres)
- **Tools:** IntelliJ IDEA CE, Postman
- **Libraries:** Spring Data JPA, Lombok, Jackson

5. Architecture & Layers

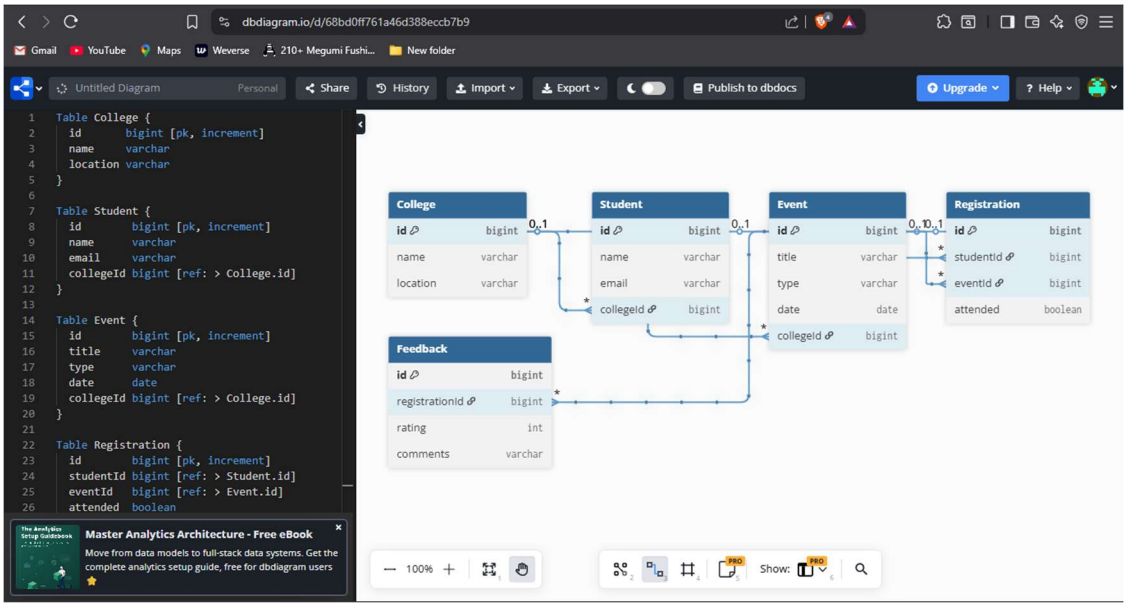
Spring Boot layered architecture:

1. **Controller Layer:** Exposes REST endpoints for College, Student, Event, Registration, Feedback, and Reports.
2. **Service Layer:** Business logic (registering students, marking attendance, computing reports).
3. **Repository Layer:** Database access using JPA repositories.
4. **Database Layer:** Entities mapped to tables via JPA/Hibernate.

6. Database Design

Entities:

Entity	Attributes	Relationships
College	id, name, location	1:N Students, 1:N Events
Student	id, name, email, collegeId	N:M Events via Registration
Event	id, title, type, date, collegeId	N:M Students via Registration
Registration	id, studentId, eventId, attended	1:1 Feedback
Feedback	id, registrationId, rating, comments	1:1 Registration



7. API Endpoints

College

- POST /api/colleges → Add college
- GET /api/colleges → Get all colleges

Student

- POST /api/students → Add student
- GET /api/students → Get all students

Event

- POST /api/events → Create event
- GET /api/events → Get all events

Registration

- POST /api/registrations → Register student
- PUT /api/registrations/{id}/attendance → Mark attendance
- GET /api/registrations/event/{eventId} → Get registrations per event

Feedback

- POST /api/feedback → Submit feedback
- GET /api/feedback/event/{eventId} → Get feedback per event

Reports

- GET /api/reports/registrations-per-event
- GET /api/reports/attendance-percentage
- GET /api/reports/average-feedback
- GET /api/reports/event-popularity
- GET /api/reports/student-participation

8. Sequence / Workflow Diagrams

1. Student registration → attendance → feedback → report generation.
2. Event creation → student registration → report endpoints.

9. Data Flow

1. **Client (Postman/Frontend)** → REST request
2. **Controller** → Receives and validates request
3. **Service** → Processes business logic
4. **Repository** → Saves/reads from DB
5. **Controller** → Returns JSON response to client

10. Tools & References

- **Postman** → API testing
- **dbdiagram.io** → ER diagram
- **IntelliJ IDEA CE** → Development
- **ChatGPT** → Brainstorming and sample code (link: https://chatgpt.com/share/68bd0ae9-77c8-8002-8437-7dc2298fe1b9)