Campus Drive Assignment – Webknot Technologies

Name: NITHESH G

USN: ENG22CS0112

College: Dayananda Sagar University

Problem Statement: Imagine you're part of a team building a Campus Event Management Platform.

● Admin Portal (Web): Used by college staff to create events (hackathons, workshops, tech talks, fests, etc.).

● Student App (Mobile): Used by students to browse events, register, and check-in on the event day. Your mission is to design and implement a basic event reporting system for this platform.

1. Assumptions and Decisions

- Each college has its own set of events, but all stored in one database (with a `college_id` field).
- Event IDs will be globally unique (auto-increment IDs) but tied to a `college_id`.
- Students can register for multiple events.
- Duplicate registrations are prevented by a composite constraint (`student_id + event_id`).
- Attendance is recorded only for registered students.
- Feedback is optional but tied to attendance.

2. Design

a) Data to Track

- Event creation (title, type, date, college).
- Student registration (event_id, student_id).
- Attendance (event_id, student_id, status).
- Feedback (event_id, student_id, rating 1–5).

b) Database Schema

Tables:

```
Colleges(college_id PK, name)
```

```
Students(student_id PK, name, email, college_id FK)
Events(event_id PK, title, type, date, college_id FK)
Registrations(reg_id PK, student_id FK, event_id FK, UNIQUE(student_id,
event_id))
Attendance(att_id PK, student_id FK, event_id FK, status)
Feedback(feedback_id PK, student_id FK, event_id FK, rating)
```

c) API Design

Endpoint	Method	Description
`/events`	POST	Create new event
`/events/:id/register`	POST	Register a student
`/events/:id/attendance`	POST	Mark attendance
`/events/:id/feedback`	POST	Submit feedback
`/reports/popularity`	GET	Events sorted by registrations
`/reports/student/:id`	GET	Participation report for a student
`/reports/top-students`	GET	Top 3 active students

d) Workflows

Registration → Attendance → Reporting

1. Student registers → stored in `Registrations`.
2. On event day, admin marks attendance → stored in `Attendance`.
3. After event, feedback submitted → stored in `Feedback`.
4. Reports generated from above tables.

---

e) Assumptions & Edge Cases

- Prevent duplicate registration → `UNIQUE(student_id, event_id)`.
- If feedback missing → exclude from average calculation.
- Cancelled events → mark in `Events` table with `status="cancelled"`.
- A student can attend multiple events, and events can have many students (many-to-many).

3. Prototype Implementation (Node.js + Express + SQLite)

I have create a **small working prototype** with:

- `server.js` → Express server.
- `db.js` → SQLite schema setup.

- `routes/` → Event, Registration, Attendance, Reports.
- `package.json` → Dependencies.

Error Handling & Validation

- Duplicate registration prevention
- Invalid rating validation (1-5)
- Missing field validation
- Non-existent student/event checks
- Foreign key constraint enforcement
- Comprehensive error messages

(You can run with: `npm install && node server.js`)

4. Reports

- **Event Popularity**:

```
SELECT e.title, COUNT(r.reg_id) AS registrations
FROM Events e
LEFT JOIN Registrations r ON e.event_id = r.event_id
GROUP BY e.event_id
ORDER BY registrations DESC;
```

- **Student Participation**:

```
SELECT s.name, COUNT(a.att_id) AS events_attended
FROM Students s
JOIN Attendance a ON s.student_id = a.student_id
WHERE a.status = 'present'
GROUP BY s.student_id;
```

- **Top 3 Most Active Students**:

```
SELECT s.name, COUNT(a.att_id) AS events_attended
FROM Students s
JOIN Attendance a ON s.student_id = a.student_id
WHERE a.status = 'present'
GROUP BY s.student_id
ORDER BY events_attended DESC
LIMIT 3;
```

Reports – sample outputs

GET /reports/popularity

```
[
  {
    "event_id": 6,
    "title": "API Test Workshop",
    "type": "Workshop",
    "date": "2025-03-15",
    "college_name": "Computer Science College",
    "registrations": 1
  },
  {
    "event_id": 1,
    "title": "Tech Conference 2025",
    "type": "Conference",
    "date": "2025-02-15",
    "college_name": "Computer Science College",
    "registrations": 0
  }
]
```

GET /reports/student/1

```
{
  "student_id": 1,
  "name": "John Doe",
  "email": "john.doe@email.com",
  "college_name": "Computer Science College",
  "events_attended": 1,
  "events_present": 1,
  "events_absent": 0,
  "total_registrations": 1
}
```

GET /reports/top-students

```
[
  {
    "student_id": 1,
    "name": "John Doe",
    "email": "john.doe@email.com",
    "college_name": "Computer Science College",
    "events_attended": 1
  }
]
```

GET /reports/overview

```
{
  "totalEvents": 6,
  "totalStudents": 7,
  "totalRegistrations": 1,
  "totalAttendance": 1,
  "totalFeedback": 1,
  "avgRating": 5
}
```

GET /events

```
[
  {
    "event_id": 6,
    "title": "API Test Workshop",
    "type": "Workshop",
    "date": "2025-03-15",
    "college_id": 1,
    "college_name": "Computer Science College"
  },
  {
    "event_id": 1,
    "title": "Tech Conference 2025",
    "type": "Conference",
    "date": "2025-02-15",
    "college_id": 1,
    "college_name": "Computer Science College"
  }
]
```

GET /events/6/registrations

```
[
  {
    "reg_id": 1,
    "student_id": 1,
    "name": "John Doe",
    "email": "john.doe@email.com",
    "college_name": "Computer Science College"
  }
]
```

GET /events/6/attendance

```
[
  {
    "att_id": 1,
    "student_id": 1,
    "name": "John Doe",
    "email": "john.doe@email.com",
    "status": "present",
    "college_name": "Computer Science College"
  }
]
```

GET /events/6/feedback

```
{
  "feedback": [
    {
      "feedback_id": 1,
      "student_id": 1,
      "name": "John Doe",
      "email": "john.doe@email.com",
      "rating": 5,
      "college_name": "Computer Science College"
    }
  ],
  "average_rating": 5,
  "total_responses": 1
}
```