**Webknot Technologies Assignment**

**I. Problem Statement:**

As part of a Campus Event Management Platform that includes an Admin Portal (web) for college staff to manage events and a Student App (mobile) for students to join the events, there is currently no system in place to track and report on event performance.

This system should integrate with both the Admin Portal and Student App, collecting data in a user-friendly format.

**II. Introduction:**

This document describes the design of a Campus Event Reporting System. The goal is to allow college staff to create events and for students to register, attend, and give feedback. The system generates reports for admins to analyse event popularity and student participation.

**III. My Approach:**

* To build a website to for college staff to organize events and an app for the students to enrol through the app and give feedback.
* This system should also track and report the following things like event popularity, student participation, etc.
* This system should be scalable as to include 50 colleges, 500 students and 20 events per semester.

**IV. Technology Stack Used:**

* Backend / API: NodeJS, Express.
* Database: Supabase/PostgreSQL.
* Frontend: ReactJS.
* Tools for diagrams: Canva.
* AI tools used: ChatGPT, GitHub Copilot, Loveable.

**V. Data To Track:**

* Events: Name, Type of event, Organizer, Date, Time, Location.
* Colleges: Name, Location.
* Students: Name, Phone number, Department.
* Registrations: How many people have registered and for how many events, Registration date.
* Attendance: How many have attended.
* Feedback: After the attendance they will be given a link to give their feedback as rating and comments.

**VI. Database Schema:**

A diagram of a planner database

AI-generated content may be incorrect.

**VII. API Design:**

1. Event APIs:

* POST /api/events - Create new event.
* GET /api /events - List all events.
* PUT /api /events/{id} - Update event.
* DELETE /api /events/{id} - Cancel event.

2. Student APIs:

* POST /api /students - Register new student.
* GET/api /students/{id} - Get student details.

3. Registration APIs:

* POST /api /registrations - Student registers for an event.
* GET /api /registrations/{event\_id} - List all registered students for an event.

4. Attendance APIs:

* POST /api /attendance - Mark attendance of the students.
* GET /api /attendance/{event\_id} - Show attendance list.

5. Feedback APIs:

* POST /api / feedback - Submit feedback those who have attended.
* GET /api / feedback/{event\_id} - View feedback.

6. Report APIs:

* GET /api /reports/event-popularity - Events sorted by number of registrations.
* GET /api /reports/student-participation - Events attended by each student.
* GET /api /reports/top-students - Top 3 most active students.
* GET /api /reports/filter?type=Workshop - Reports filtered by event type.

**VIII. Workflow Diagram:**

A diagram of a flowchart

AI-generated content may be incorrect.

**IX. Assumptions & Edge Cases:**

* Event ID’s should be unique for all colleges, Eg: for REVA University the ID should start from RU and event number after the RU wordings, the same goes for other colleges (RU001, RU002, etc.).
* The students cannot register more than once to a single event but can register for multiple events once.
* Only registered students get the attendance.
* Only present students get the feedback link to submit their feedback.
* If the event is cancelled then the registrations will be closed, the students already registered will get a mail about the cancellation.
* If the feedback is not received the report should ignore it and send the report on the submitted feedbacks.

**X. Code for Reporting:**

1. Event Popularity Report:

SELECT event\_id, COUNT(student\_id) AS registrations FROM registrations GROUP BY event\_id ORDER BY registrations DESC;

“This gives the number of students who have registered to the said event, We can access it through the event\_id”

2. Student Participation Report:

SELECT student\_id, COUNT(event\_id) AS attended\_events FROM attendance WHERE status='Present' GROUP BY student\_id;

“This gives the events attended by the respective student when they are marked present by the admin”

3. Top 3 Most Active Students:

SELECT student\_id, COUNT(event\_id) AS attended FROM attendance WHERE status='Present' GROUP BY student\_id ORDER BY attended DESC LIMIT 3;

“This gives the list of the top 3 students who have attended more events and are very active”

4. Flexible Reports by Event Type:

SELECT e.type, COUNT(r.student\_id) AS registrations FROM events e JOIN registrations r ON e.event\_id=r.event\_id WHERE e.type='Workshop' GROUP BY e.type;

“Here we can access the reports based on the event type”

**XI. Scalable Considerations:**

* Event ID’s must be different across different colleges for easy management.
* The data should be separated per college for better and fast response from the system, A large database will complicate things.

**XII. Output:**

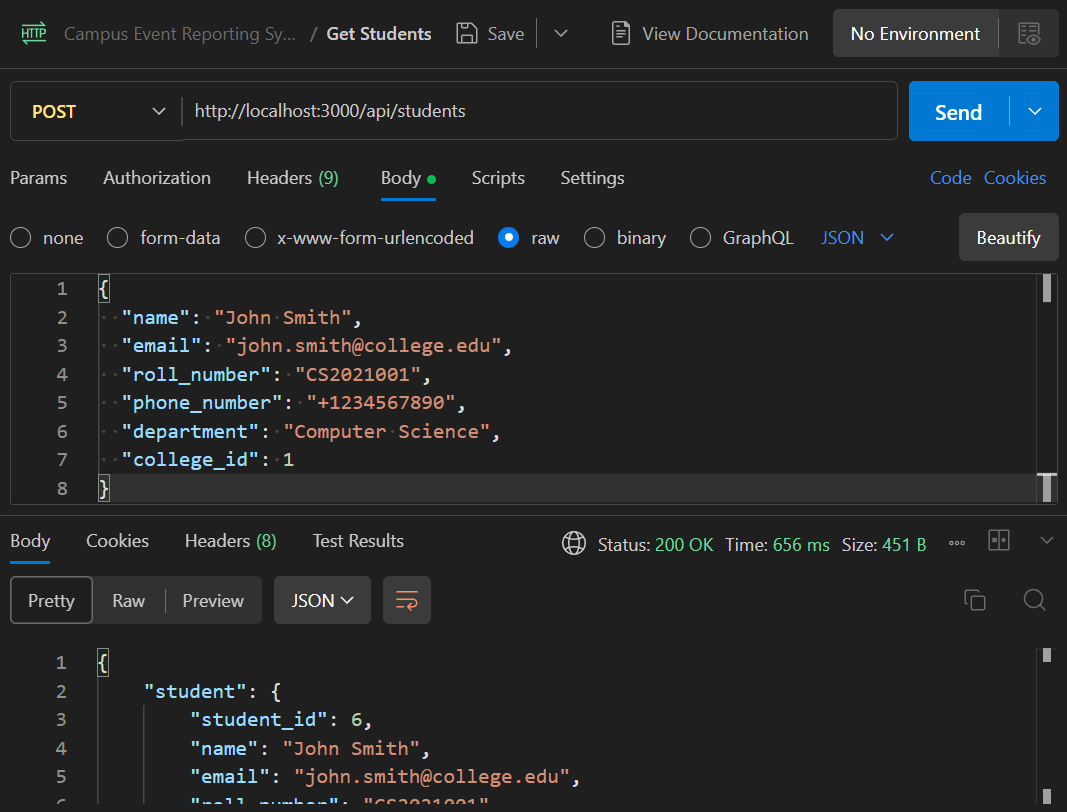
1. **Backend API Responses:**

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

****

**A screenshot of a computer program

AI-generated content may be incorrect.**

1. **Frontend Snapshots:**

**A screenshot of a computer

AI-generated content may be incorrect.**

**A white background with black text

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.**

**XIII. References:**

Chatgpt Chat Link: https://chatgpt.com/share/68bd3fac-9b3c-8002-8dcb-df80fd23a973