Design Document – Campus Event Management System

1. Inroduction

This project is a **Campus Event Management System** that helps colleges create and manage events while allowing students to register and participate easily. The system has two parts:

- **Admin side** for staff to add events.
- **Student side** for browsing, registering, attendance, and feedback.

The main goal is to provide basic event reports such as total registrations, attendance percentage, and average feedback. It also supports reports like event popularity and student participation. The project is built to handle multiple colleges and a large number of students in a simple and scalable way.

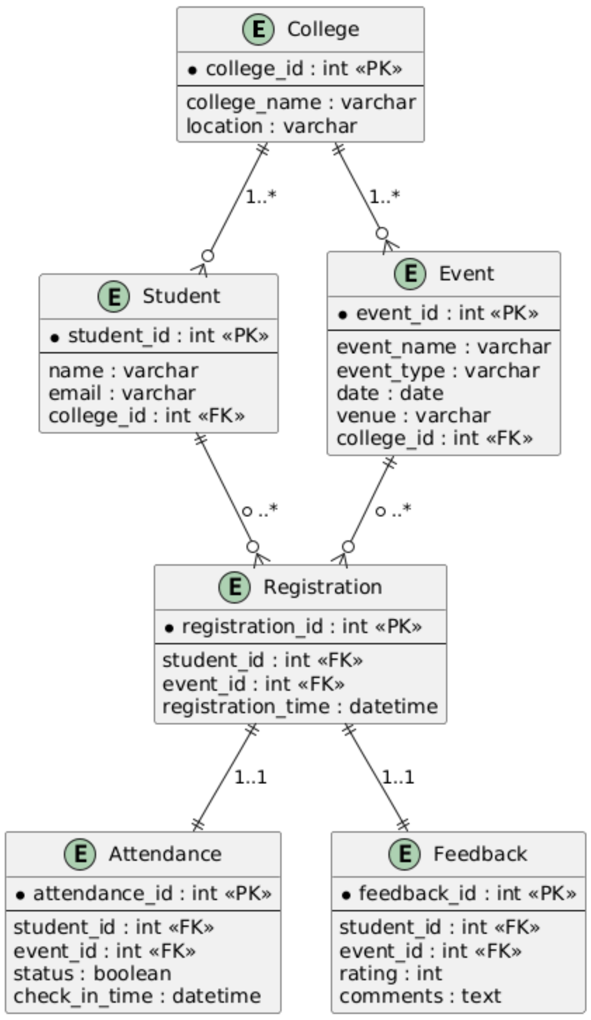
2. Assumptions & Decisions

1. **Event ID Uniqueness**
   - Event IDs are considered **unique per college**. This keeps data separation simple and avoids conflicts when multiple colleges create similar events.
2. **Duplicate Registrations**
   - A student can register for an event only once. Any attempt to register again is blocked at the database level.
3. **Attendance Logic**
   - Attendance is tracked as a **boolean flag** (Present/Absent) for simplicity.
   - Optionally, a **timestamp-based check-in** can be used if events require detailed entry times.
4. **Feedback Scale**
   - Feedback is collected on a **1–5 rating scale**, where 1 = Poor and 5 = Excellent.
   - This keeps reporting simple while still giving meaningful insights.
5. **Database Choice**
   - **SQLite** is used for the prototype to keep setup lightweight.
   - The design is easily scalable to **Postgres/MySQL** for production environments with larger datasets.

3. Data to Track

1. **Event Creation Details**
   - Event ID, event name, type (workshop, fest, seminar, etc.), date, time, venue, and the college organizing it.
2. **Student Registration**
   - Student ID, name, event ID, and registration timestamp.
3. **Attendance Status**
   - Student ID, event ID, and attendance flag (present/absent) or timestamp if detailed logging is needed.
4. **Feedback & Ratings**
   - Student ID, event ID, feedback rating (1–5 scale), and optional comments.

4. Database Schema(ER-Diagram)



The system is designed with six main entities:

1. **College**
   - Attributes: college_id, college_name, location
   - Each college can host multiple events and enroll many students.
2. **Student**
   - Attributes: student_id, name, email, college_id (FK)
   - A student belongs to one college but can register for many events.
3. **Event**
   - Attributes: event_id, event_name, event_type, date, venue, college_id (FK)
   - Each event is created by a college and can have multiple student registrations.
4. **Registration**
   - Attributes: registration_id, student_id (FK), event_id (FK), registration_time
   - A student can register for many events, but only once per event.
5. **Attendance**
   - Attributes: attendance_id, student_id (FK), event_id (FK), status (boolean), check_in_time (optional)
   - Tracks whether a student attended the event.
6. **Feedback**
   - Attributes: feedback_id, student_id (FK), event_id (FK), rating (1–5), comments (optional)
   - Captures student feedback after attending an event.

Relationships

- **College → Student**: One-to-Many (a college has many students).
- **College → Event**: One-to-Many (a college hosts many events).
- **Student → Registration → Event**: Many-to-Many (a student can join many events; an event can have many students).
- **Registration → Attendance**: One-to-One (each registration corresponds to one attendance record).
- **Registration → Feedback**: One-to-One (each registration can have one feedback entry).

5. API Design

1. Event APIs

POST /events → Create a new event

```
Request:
{
  "event_name": "Hackathon 2025",
  "event_type": "Workshop",
  "date": "2025-09-10",
  "venue": "Auditorium",
  "college_id": 1
}

Response:
{
  "event_id": 101,
  "message": "Event created successfully"
}
```

GET /events → Retrieve all events

```
Response:
[
  {
    "event_id": 101,
    "event_name": "Hackathon 2025",
    "event_type": "Workshop",
    "date": "2025-09-10",
    "venue": "Auditorium",
    "college_id": 1
  },
  {
    "event_id": 102,
    "event_name": "Tech Talk on AI",
    "event_type": "Seminar",
    "date": "2025-09-15",
    "venue": "Hall 2",
    "college_id": 1
  }
]
```

2. Registration APIs

POST /register → Student registers for an event

```
Request:
{
  "student_id": 501,
  "event_id": 101
}

Response:
{
  "registration_id": 2001,
  "message": "Student registered successfully"
}
```

3. Attendance APIs

POST /attendance → Mark student attendance

```
Request:
{
  "student_id": 501,
  "event_id": 101,
  "status": true
}

Response:
{
  "message": "Attendance marked successfully"
}
```

4. Feedback APIs

POST /feedback → Collect student feedback

```
Request:
{
  "student_id": 501,
  "event_id": 101,
  "rating": 5,
  "comments": "Great event!"
}
```

```
Response:
{
  "feedback_id": 3001,
  "message": "Feedback submitted successfully"
}
```

5. Report APIs

GET /reports/popularity → Events sorted by number of registrations

```
Response:
[
  { "event_id": 101, "event_name": "Hackathon 2025", "registrations": 120 },
  { "event_id": 102, "event_name": "Tech Talk on AI", "registrations": 90 }
]
```

GET /reports/top-students → Top 3 most active students

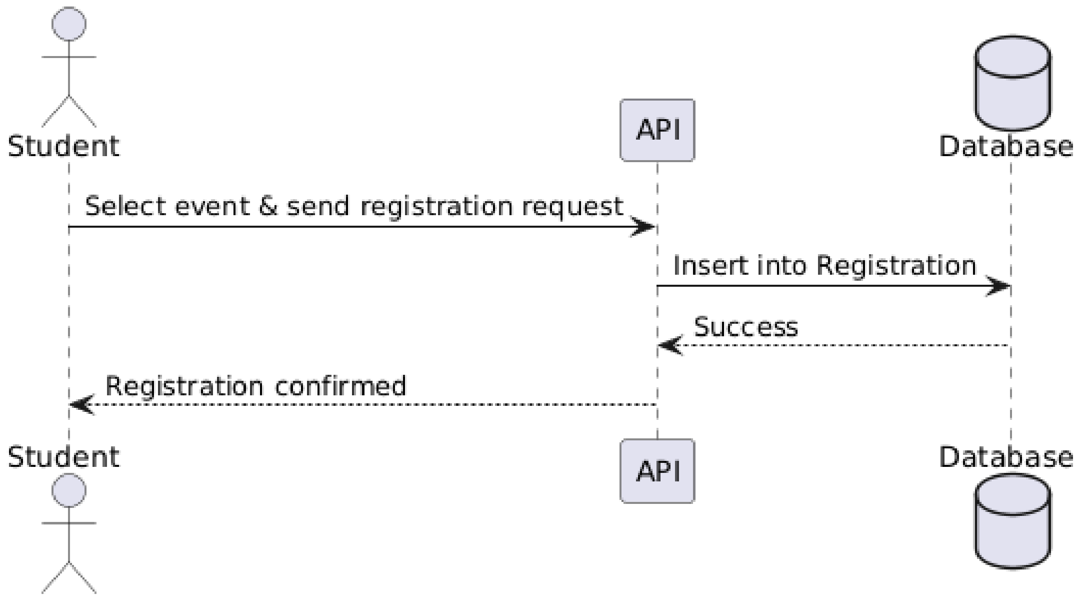
```
Response:
[
  { "student_id": 1, "student_name": "Kshitj Kumar", "events_attended": 3 },
  { "student_id": 3, "student_name": "Harshal Desai", "events_attended": 2 },
  { "student_id": 4, "student_name": "Kiran Belakeri", "events_attended": 1 }
]
```

GET /reports/student/{student_id} → Student participation report

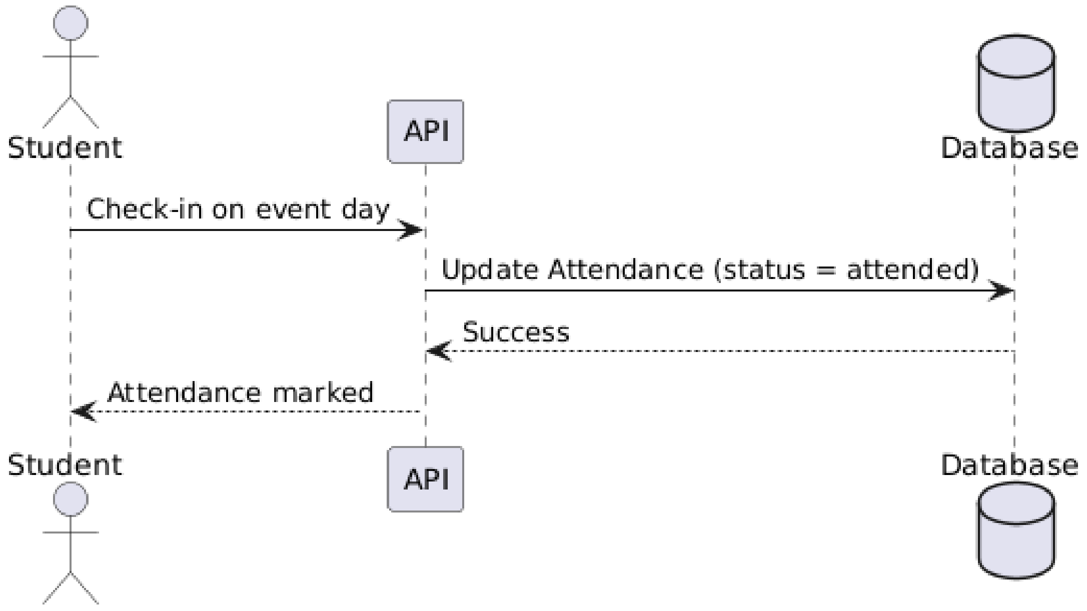
```
[
    {
        "attended_events": 2,
        "name": "Kiran Belakeri",
        "student_id": 4
    },
    {
        "attended_events": 1,
        "name": "Harshal Desai",
        "student_id": 3
    },
    {
        "attended_events": 1,
        "name": "Kshitij Kumar",
        "student_id": 1
    }
]
```

6. Workflows

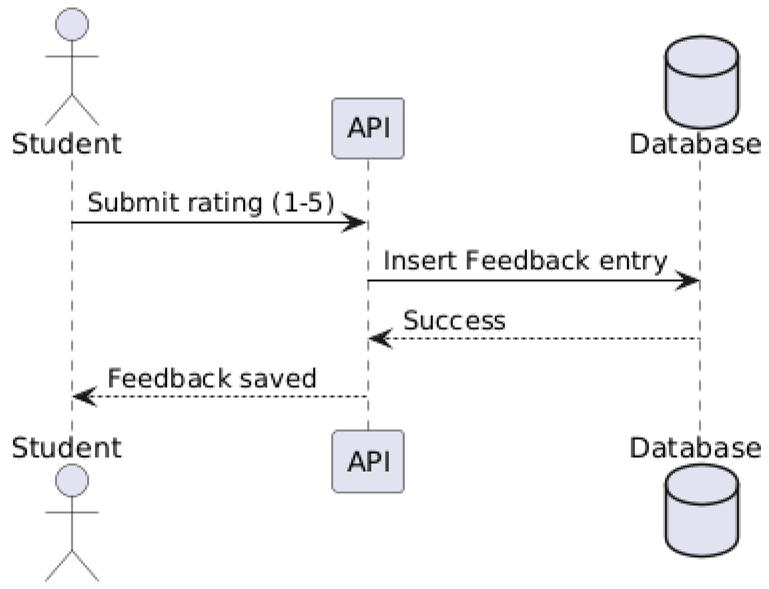
1. Registration Workflow



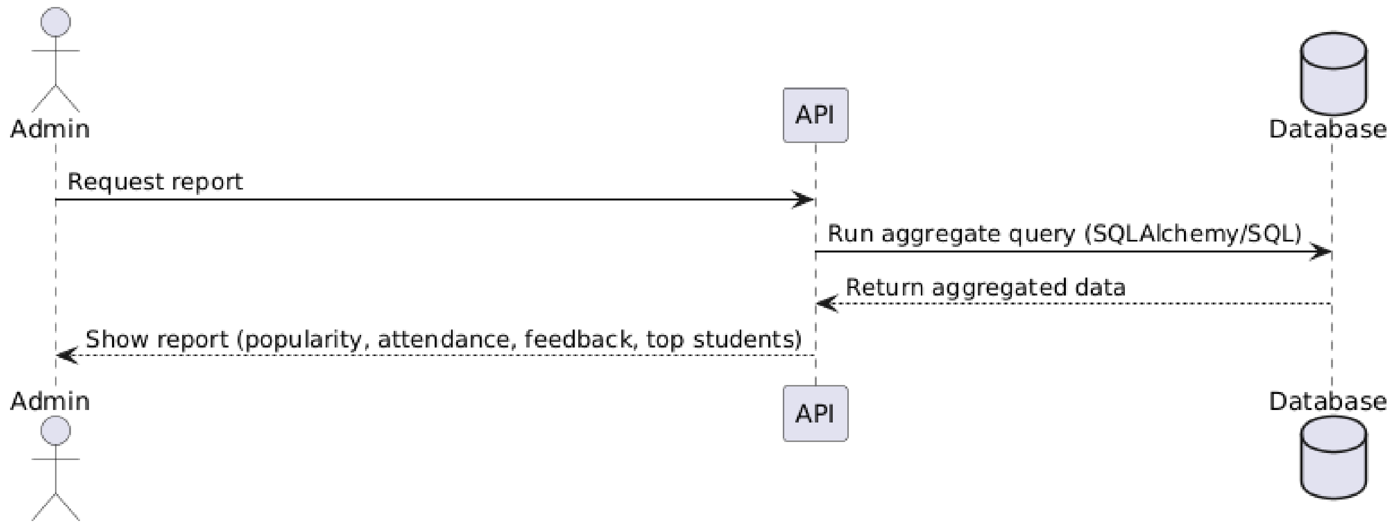
2. Attendance Workflow



3. Feedback Workflow



4. Reporting Workflow


Admin
API
Database
Request report
Run aggregate query (SQLAlchemy/SQL)
Return aggregated data
Show report (popularity, attendance, feedback, top students)
Admin
API
Database

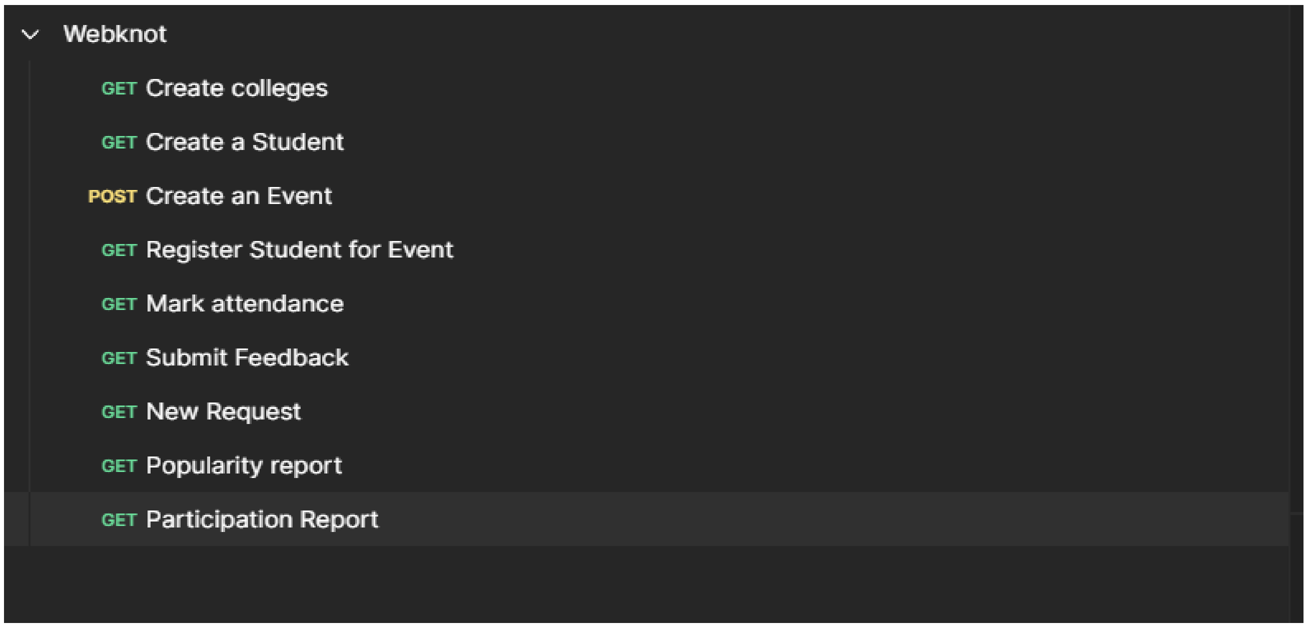
7. Edge Cases & Error Handling

1. **Duplicate Registration Attempts**
    - If a student tries to register again for the same event, the system checks for an existing registration entry and prevents duplication.
    - Response includes the existing reg_id with a status message.
2. **Event Cancellation**
    - If an event is cancelled, related registrations remain in the database but are marked as inactive/cancelled.
    - Reports exclude cancelled events.
3. **Student Absent**
    - If a student does not check in, their attendance status remains unmarked (registered only).
    - Attendance reports only count students with status = attended.
4. **Feedback Missing**
    - Feedback is optional. Students who do not submit ratings are excluded from average calculations.
    - Reports only consider non-null feedback entries.
5. **Concurrent Student Registrations**
    - Database constraints (unique student-event pair) ensure no duplicate entries even when multiple requests are sent at the same time.

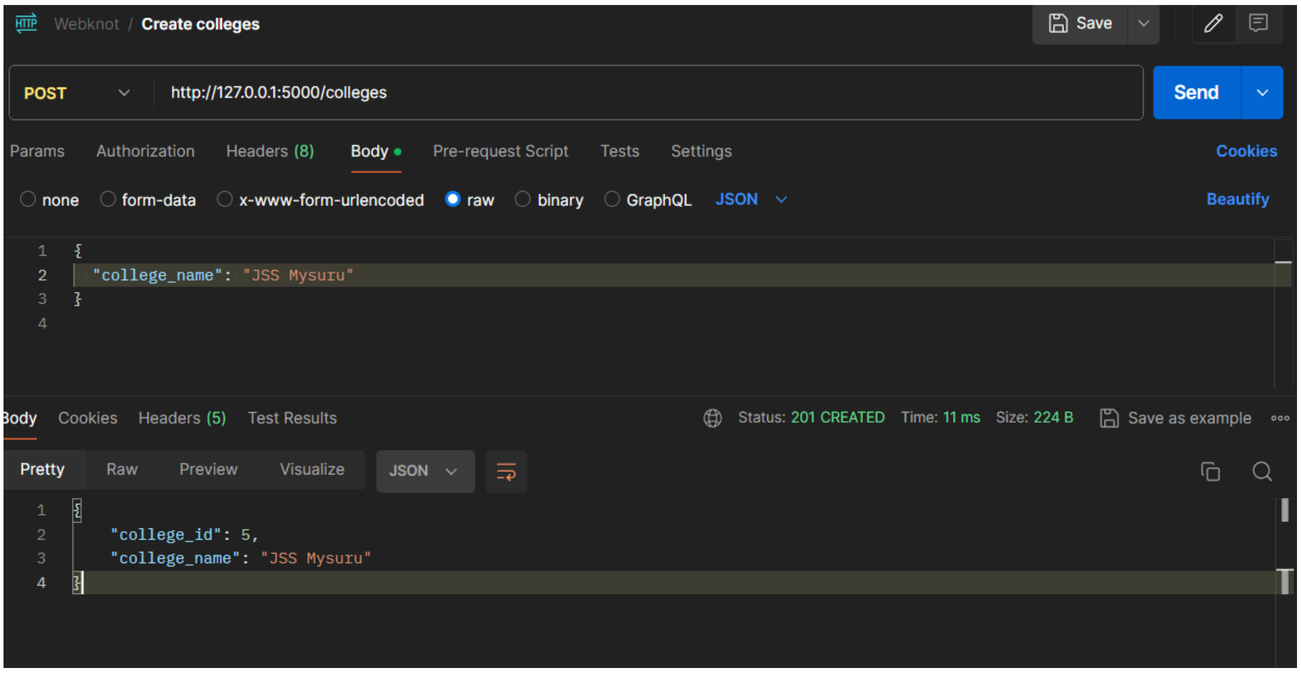
8. Prototype Implementation (Summary)

- **Framework:** Implemented using **Flask** (for REST APIs) and **SQLAlchemy** (ORM).
- **Database:** Used **SQLite** for lightweight setup during prototype phase.
- **Models Implemented:**
    - College
    - Student
    - Event
    - Registration (also stores attendance status and feedback)
- **Schema Creation:** Tables created automatically using db.create_all().
- **API Endpoints:** Implemented for event creation, student registration, attendance marking, feedback submission, and generating reports.
- **Testing:** APIs were tested successfully using **Postman** with sample requests and responses.

Screenshots of my Postman API reportings and test cases



Webknot
- GET Create colleges
- GET Create a Student
- POST Create an Event
- GET Register Student for Event
- GET Mark attendance
- GET Submit Feedback
- GET New Request
- GET Popularity report
- GET Participation Report



Webknot / Create colleges

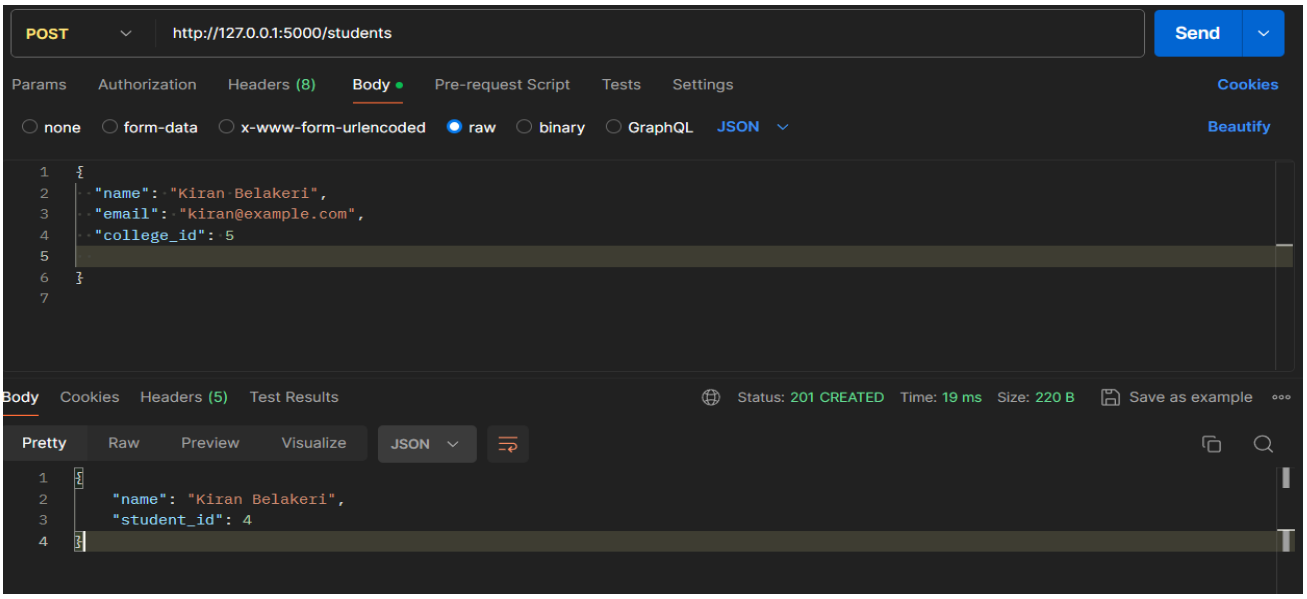
POST http://127.0.0.1:5000/colleges

Body: raw JSON

```
{
"college_name": "JSS Mysuru"
}
```

Status: 201 CREATED Time: 11 ms Size: 224 B

```
{
    "college_id": 5,
    "college_name": "JSS Mysuru"
}
```



POST http://127.0.0.1:5000/students

Body: raw JSON

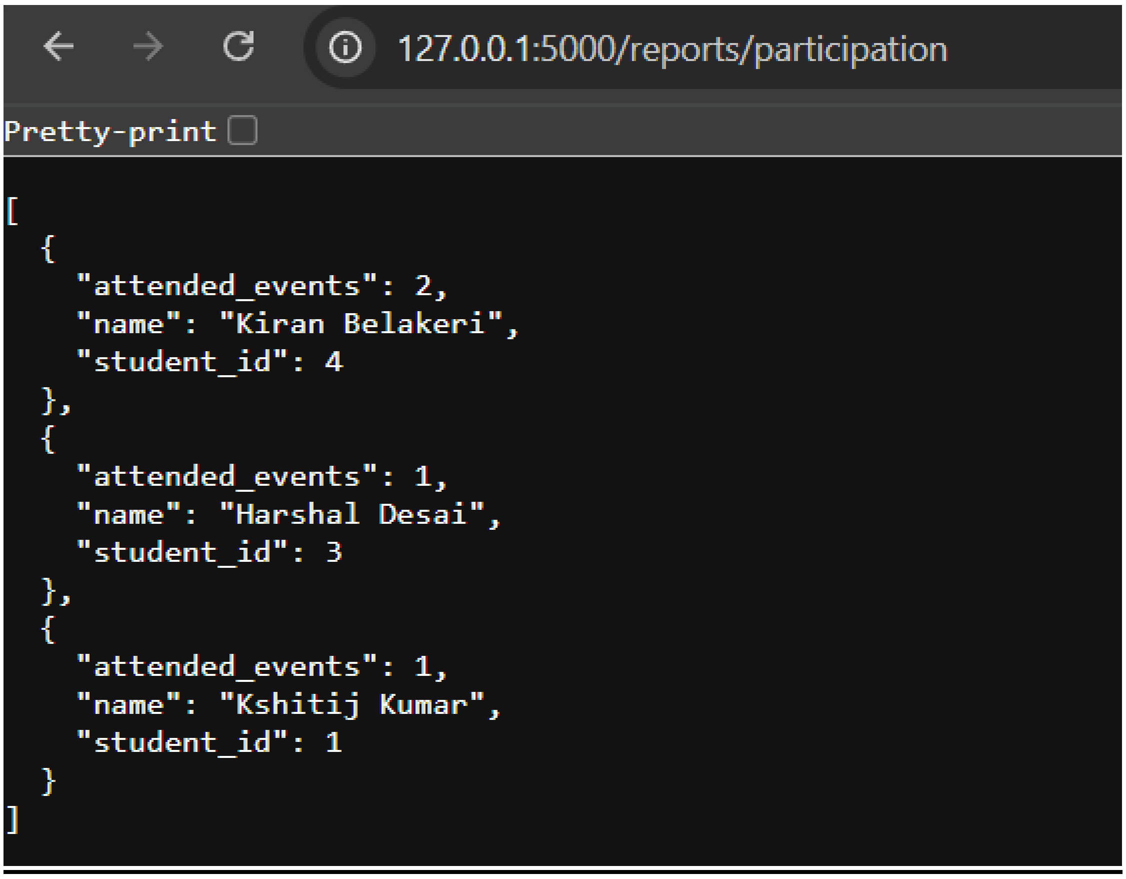
```
{
  "name": "Kiran Belakeri",
  "email": "kiran@example.com",
  "college_id": 5

}
```

Status: 201 CREATED Time: 19 ms Size: 220 B

```
{
    "name": "Kiran Belakeri",
    "student_id": 4
}
```

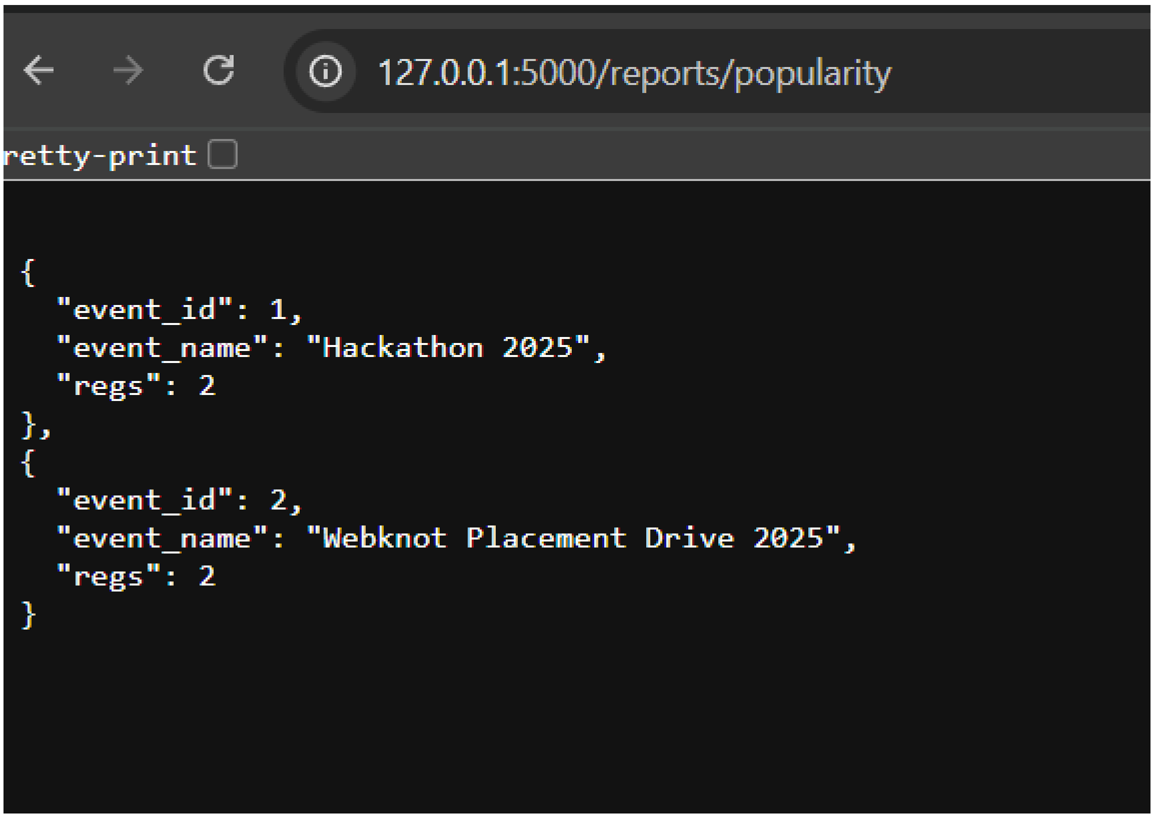
9. Screenshots of Browser

Participation Report



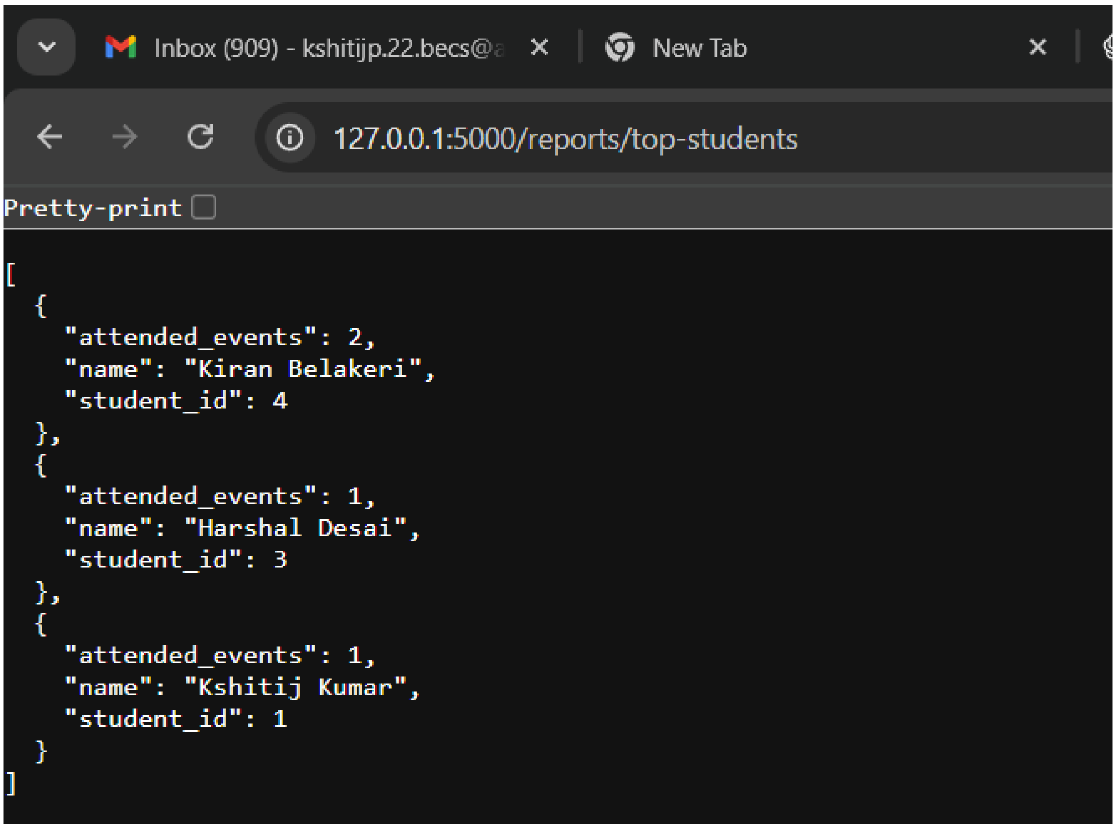
```
[
  {
    "attended_events": 2,
    "name": "Kiran Belakeri",
    "student_id": 4
  },
  {
    "attended_events": 1,
    "name": "Harshal Desai",
    "student_id": 3
  },
  {
    "attended_events": 1,
    "name": "Kshitij Kumar",
    "student_id": 1
  }
]
```

Popularity report :



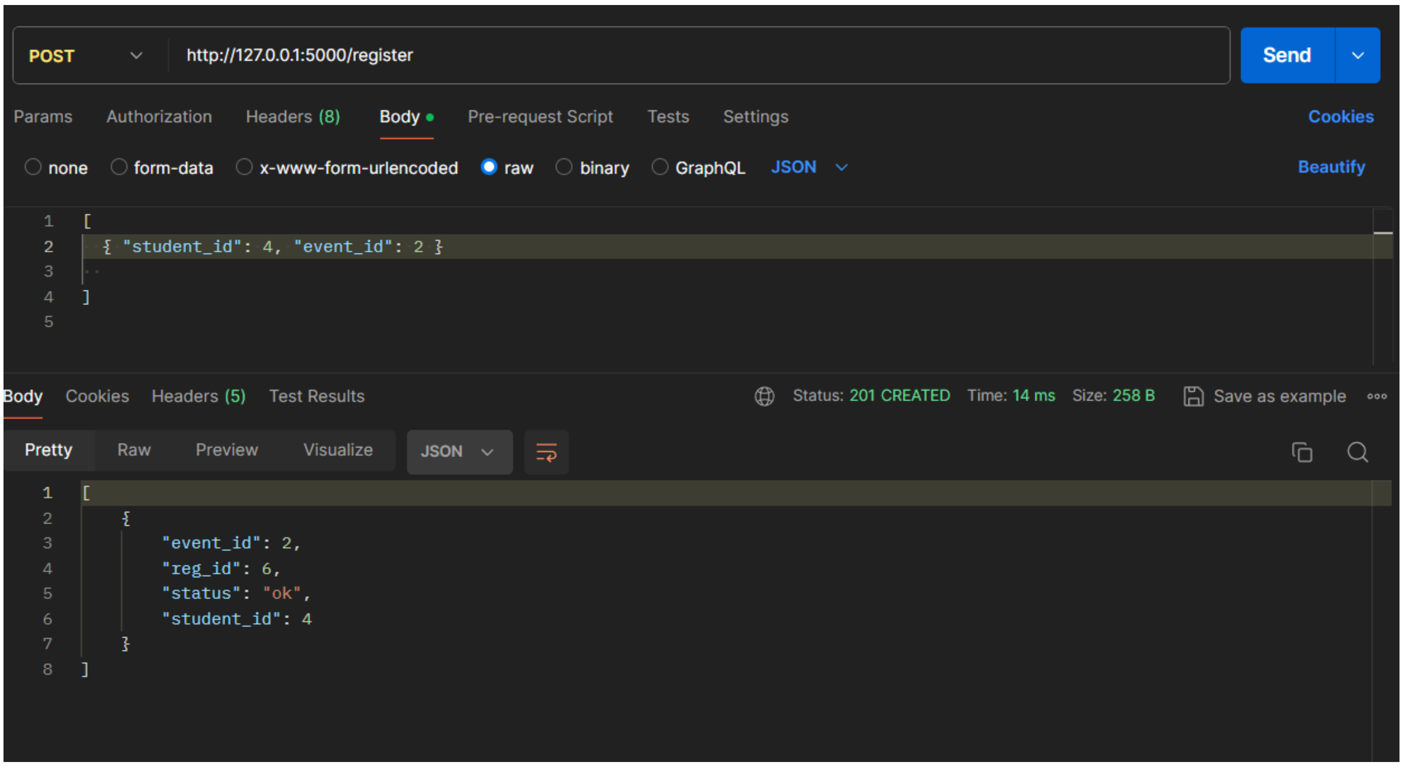
```
 {
   "event_id": 1,
   "event_name": "Hackathon 2025",
   "regs": 2
 },
 {
   "event_id": 2,
   "event_name": "Webknot Placement Drive 2025",
   "regs": 2
 }
```

Top -3 students


Inbox (909) - kshitijp.22.becs@a
New Tab
127.0.0.1:5000/reports/top-students
Pretty-print

```
[
  {
    "attended_events": 2,
    "name": "Kiran Belakeri",
    "student_id": 4
  },
  {
    "attended_events": 1,
    "name": "Harshal Desai",
    "student_id": 3
  },
  {
    "attended_events": 1,
    "name": "Kshitij Kumar",
    "student_id": 1
  }
]
```

Register Student for an Event



```
[
  { "student_id": 4, "event_id": 2 }

]
```

Status: 201 CREATED Time: 14 ms Size: 258 B

```
[
    {
        "event_id": 2,
        "reg_id": 6,
        "status": "ok",
        "student_id": 4
    }
]
```

10. Conclusion

This project successfully implements a **Campus Event Management System** with the following key features:

- Event creation and student registration
- Attendance tracking
- Feedback collection (1–5 rating scale)
- Reporting system (event popularity, attendance percentage, student participation, top active students)

The system was built using **Flask + SQLAlchemy + SQLite** for rapid prototyping and tested using Postman. It demonstrates how event data can be tracked, stored, and analyzed efficiently.

Possible Extensions

- **Authentication & Roles:** Separate login for admins and students with role-based permissions.
- **Email / SMS Notifications:** Reminders for registrations, event updates, or cancellations.
- **User Interface:** Full-featured web portal and mobile app for improved usability.
- **Scalability Enhancements:** Deploy with PostgreSQL/MySQL, add caching, and containerize using Docker.

This prototype provides a strong foundation for a scalable and production-ready event management platform.

ChatGPT Session Link-

https://chatgpt.com/share/68bb384e-a838-8000-9136-765d56a4b850