

## 1. 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).

---

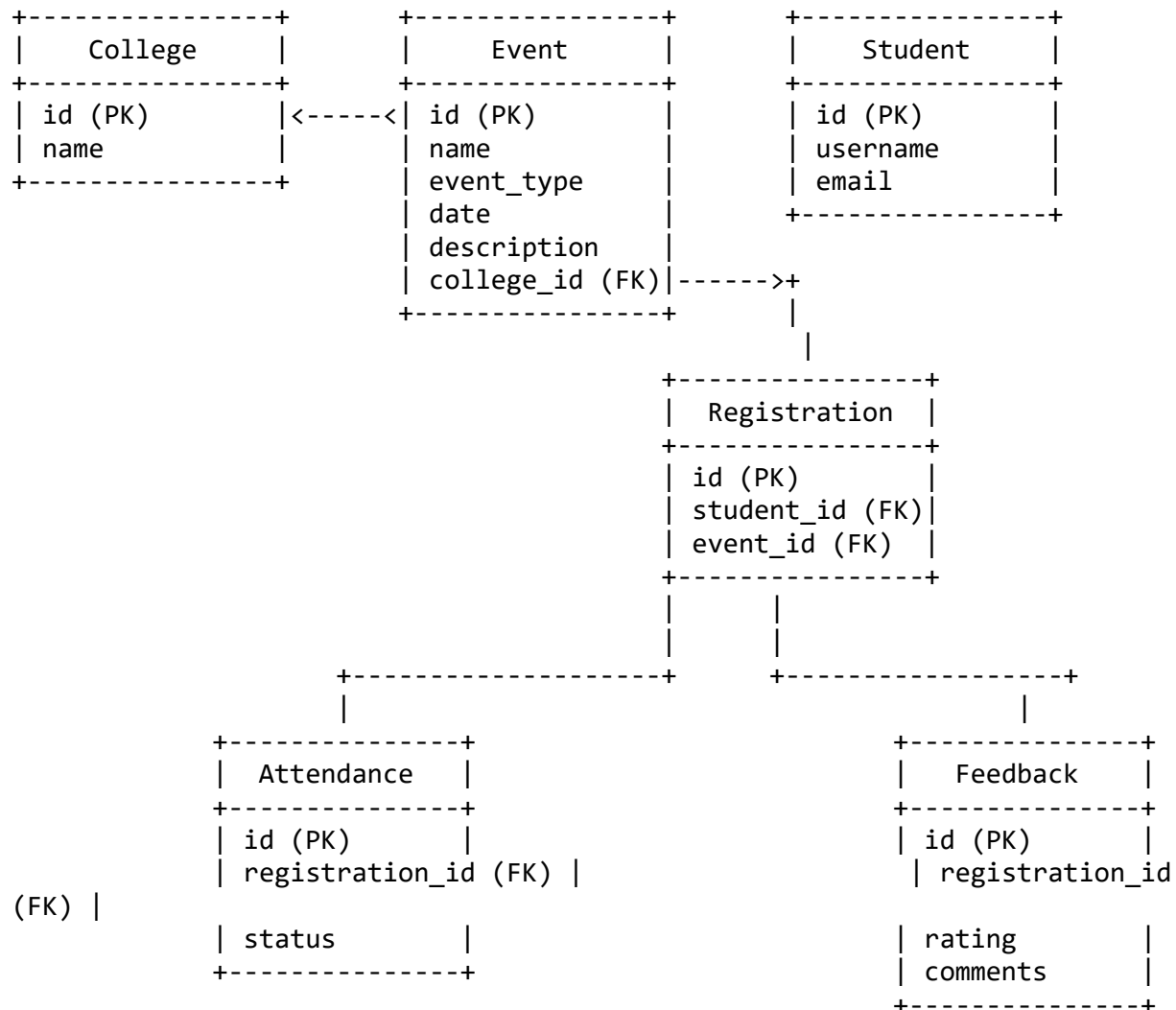
## 2. Design Document

### 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)

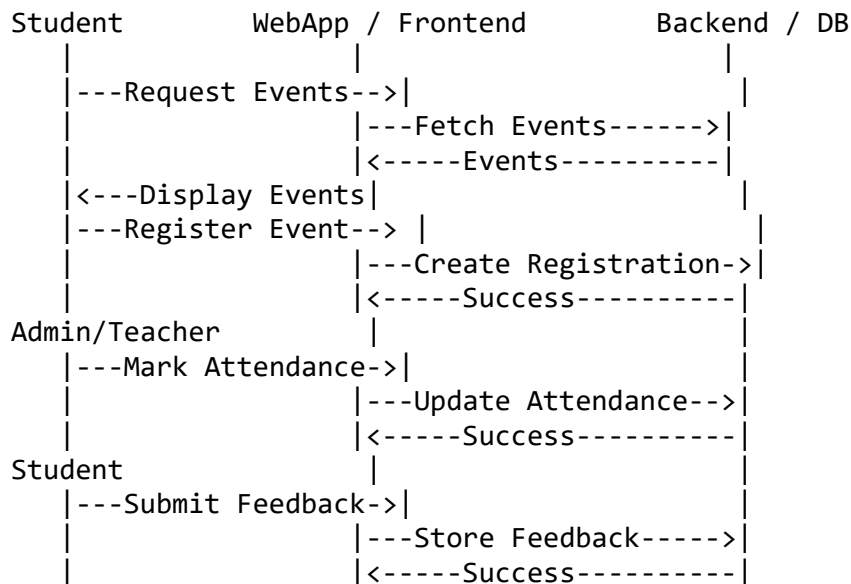


## API Design

- 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)



## Assumptions & Edge Cases

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

---

## 3. 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: Admin Dashboard Table -->
<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 tbody");
  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;`

---

Some screenshots of chat with Ai:

Link: <https://chatgpt.com/share/68bcd71-e64c-8013-80c8-7e4e7ac2dc55>

The screenshot shows the ChatGPT web interface. The left sidebar contains a list of chats, with 'Improve event system UI' selected. The main area displays two code snippets. The first snippet, titled '2. Student Route', is a Python Flask route that handles attendance requests. It checks if a request already exists for a given student and event, and if so, it flashes a warning and redirects to the student dashboard. If not, it creates a new record in the database and redirects to the student dashboard. The second snippet, titled '3. Admin Approves Attendance', is a Python Flask route that displays all pending requests to an administrator. The interface includes a 'Copy code' button for each snippet and a 'Share' button at the top right.

```
requested_at = db.Column(db.DateTime, default=datetime.utcnow)
```

### 2. Student Route

```
python

# ----- Student Attendance Request -----
@app.route('/student/request_attendance', methods=['POST'])
def student_request_attendance():
    student_id = request.form['student_id']
    event_id = request.form['event_id']

    # Check if already requested
    existing = AttendanceRequest.query.filter_by(student_id=student_id, event_id=event_id).first()
    if existing:
        flash("You already requested attendance for this event.", "warning")
        return redirect(url_for('student_dashboard'))

    request_entry = AttendanceRequest(student_id=student_id, event_id=event_id)
    db.session.add(request_entry)
    db.session.commit()

    flash("Attendance request submitted! Waiting for admin approval.", "success")
    return redirect(url_for('student_dashboard'))
```

### 3. Admin Approves Attendance

Admin sees all pending requests.

```
python

# ----- Admin View Pending Requests -----
@app.route('/admin/attendance_requests')
def admin_attendance_requests():
    if 'admin_id' not in session:
```

The screenshot shows the ChatGPT web interface. The left sidebar contains a list of chats, with 'Improve event system UI' selected. The main area displays a code snippet for an attendance form and a suggested flow for attendance requests. The code snippet, titled '@app.route("/attendance\_form")', is a Python Flask route that renders the 'attendance.html' template. The suggested flow, titled 'Suggested Flow', describes the process of submitting an attendance request and the database structure for the 'AttendanceRequest' table. The interface includes a 'Copy code' button for the code snippet and a 'Share' button at the top right.

```
@app.route("/attendance_form")
def attendance_form():
    return render_template("attendance.html")
```

this i want to separate for admin and student .where as in admin side he see the the student name and is student id and check box in frontend.if student is request for attendences admin should able give attendeces then only it save in data base otherwise it should not save anything in data base .next from student side based on the event name and id he should be able to request for attendeces

Got it 🙌 — you want two separate attendance flows:

1. Student side → student requests attendance for a particular event.
2. Admin side → admin sees pending attendance requests (student name + ID + event), and only after the admin approves/marks it, the record should be saved/updated in the database.

### ✓ Suggested Flow

#### 1. Student Request Attendance (does not mark directly)

- Student submits a form with `event_id` and `student_id`.
- This creates a record in a new table: `AttendanceRequest` (pending state).

```
python

class AttendanceRequest(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    student_id = db.Column(db.Integer, db.ForeignKey('student_id'), nullable=False)
    event_id = db.Column(db.Integer, db.ForeignKey('event_id'), nullable=False)
    status = db.Column(db.String(20), default="Pending") # Pending / Approved / Rejected
    requested_at = db.Column(db.DateTime, default=datetime.utcnow)
```

#### 2. Student Route

```
python
```

+ Ask anything

ChatGPT can make mistakes. Check important info. See Cookie Preferences.

Event-Management Webknot Technology Webknot Campus Dr Improve event sy Page not found - Git Downloads Top Private University Sign in to your account Recent - Google Drive

chatgpt.com/c/68bbffc8-0244-8324-a429-a58971860c4

ChatGPT

New chat  
Search chats  
Library  
Sora  
GPTs  
image generator  
Projects  
Chats  
New chat  
Python 3.13 compatibility fix  
Fix template not found  
Improve event system UI  
Push code to GitHub  
Module not found error  
Flask 3.0 fix  
Resume changes for frontend  
Resume suggestions for KPMG  
Database field explanation  
Decode shorthand  
Code for document chunking  
C++ OR bit problem  
Test case selection guide  
Av Thrineth Av Thrineth

### 3. Admin Approves Attendance

Admin sees all pending requests.

```
python
# ----- Admin View Pending Requests -----
@app.route('/admin/attendance_requests')
def admin_attendance_requests():
    if 'admin_id' not in session:
        flash("Please login first.", "warning")
        return redirect(url_for('admin_login_form'))

    requests = AttendanceRequest.query.filter_by(status="Pending").all()
    return render_template('admin/attendance_requests.html', requests=requests)
```

### 4. Admin Approves / Rejects

```
python
@app.route('/admin/approve_attendance/<int:request_id>', methods=[ 'POST' ])
def approve_attendance(request_id):
    req = AttendanceRequest.query.get_or_404(request_id)

    # Approve -> Create real Attendance record
    attendance = Attendance(
        registration_id=req.student_id, # or registration mapping
        status="Present"
    )
    db.session.add(attendance)

    req.status = "Approved"
    db.session.commit()
    flash("Attendance approved and saved!", "success")
    return redirect(url_for('admin_attendance_requests'))
```

+ Ask anything

ChatGPT can make mistakes. Check important info. See [Cookie Preferences](#).

Event-Management Webknot Technology Webknot Campus Dr Improve event sy Page not found - Git Downloads Top Private University Sign in to your account Recent - Google Drive

chatgpt.com/c/68bbffc8-0244-8324-a429-a58971860c4

ChatGPT

New chat  
Search chats  
Library  
Sora  
GPTs  
image generator  
Projects  
Chats  
New chat  
Python 3.13 compatibility fix  
Fix template not found  
Improve event system UI  
Push code to GitHub  
Module not found error  
Flask 3.0 fix  
Resume changes for frontend  
Resume suggestions for KPMG  
Database field explanation  
Decode shorthand  
Code for document chunking  
C++ OR bit problem  
Test case selection guide  
Av Thrineth Av Thrineth

```
req = AttendanceRequest.query.get_or_404(request_id)
req.status = "Rejected"
db.session.commit()
flash("Attendance request rejected.", "danger")
return redirect(url_for('admin_attendance_requests'))
```

### Frontend

#### Student (request form)

student/attendance\_request.html

```
html
<form method="POST" action="{{ url_for('student_request_attendance') }}">
  Student ID: <input type="number" name="student_id" required><br>
  Event ID: <input type="number" name="event_id" required><br>
  <button type="submit">Request Attendance</button>
</form>
```

#### Admin (approve requests)

admin/attendance\_requests.html

```
html
<h2>Pending Attendance Requests</h2>
<table border="1">
  <tr>
    <th>Request ID</th>
    <th>Student ID</th>
    <th>Event ID</th>
    <th>Action</th>
  </tr>
  <tr>
    <td>{% for req in requests %}</td>
  </tr>
```

+ Ask anything

ChatGPT can make mistakes. Check important info. See [Cookie Preferences](#).

Event-Management

Webknot Technology

Webknot Campus Dr

Improve event sy

Page not found · GitH

Downloads

Top Private University

Sign in to your accou

Recent - Google Drive

+

-

o

x

chatgpt.com/c/68bbffc8-0244-8324-a429-a58971860c4

ChatGPT

New chat

Search chats

Library

Sora

GPTs

image generator

Projects

Chats

New chat

Python 3.13 compatibility fix

Fix template not found

Improve event system UI

Push code to GitHub

Module not found error

Flask 3.0 fix

Resume changes for frontend

Resume suggestions for KPMG

Database field explanation

Decode shorthand

Code for document chunking

C++ OR bit problem

Test case selection guide

Av Thrinesh Av Thrinesh

Admin (approve requests)

admin/attendance\_requests.html

html

Copy code

```
<h2>Pending Attendance Requests</h2>
<table border="1">
  <tr>
    <th>Request ID</th>
    <th>Student ID</th>
    <th>Event ID</th>
    <th>Action</th>
  </tr>
  {% for req in requests %}
  <tr>
    <td>{{ req.id }}</td>
    <td>{{ req.student_id }}</td>
    <td>{{ req.event_id }}</td>
    <td>
      <form method="POST" action="{{ url_for('approve_attendance', request_id=req.id) }}" style="display: inline-block; margin-right: 10px;">
        <button type="submit">Approve</button>
      </form>
      <form method="POST" action="{{ url_for('reject_attendance', request_id=req.id) }}" style="display: inline-block;">
        <button type="submit">Reject</button>
      </form>
    </td>
  </tr>
  {% endfor %}
</table>
```

This way:

- Student requests (nothing written in Attendance yet).
- Admin approves/rejects (only then saved into Attendance).

+ Ask anything

ChatGPT can make mistakes. Check important info. See [Cookie Preferences](#).