**1. Project Overview**

This is a Campus Event management portal where Admins can host their events through this and the students can enrol themselves to the event. It can also be useful in tracking the attendance and the duration of event.  
It provides a platform for:

* Students → to register for events, mark attendance, and submit feedback.
* Organizers → to create events, track registrations, and analyze reports.

**2. Objectives**

* Simplify event management through automation.
* Ensure secure login and access control for students.
* Provide real-time attendance and feedback tracking.
* Generate analytical reports for event evaluation.
* Collect feedback from each events
* Encourage the students to take part in more cultural events and tech events to upskill themselves

**3. Technology Stack**

**Backend**

* **Framework**: Django REST Framework
* **Database**: MySQL
* **Authentication**: JWT (SimpleJWT)
* **Language**: Python 3

**Frontend**

* **Framework**: React.js
* **Styling**: Bootstrap 5
* **API Calls**: Axios

**4. System Architecture**

**🔹 High-Level Architecture**

[Frontend: React + Bootstrap] <-- Axios REST calls -->

[Backend: Django + DRF + JWT] <-- ORM -->

[Database: MySQL]

* Frontend consumes REST APIs for all operations (login, events, registrations, attendance, feedback).
* Django has a corsheaders module which is essential in accepting the cross-origin-requests.
* Backend handles authentication, business logic, and persistence.
* Simple rest frameworks views is helpful in helping the user login
* Database stores structured data for colleges, students, events, etc.
* The Django orm used that is Object relational mapping is uselful in creating the rows and columns in databases

**5. Modules & Features**

**5.1 Student Module**

* Register as a new student (with password & confirm password).
* Login with email & password.
* Register for available events.
* Mark attendance (only if logged in).
* Submit event feedback.

**5.2 Event Module**

* Create and manage events (workshop, seminar, hackathon, fest, etc.).
* Track event registrations.
* Calculate attendance and feedback statistics.

**5.3 Reports Module**

* Registrations per event.
* Attendance percentage.
* Average feedback per event.
* Top active students (by attendance).

**6. Database Design (ERD)**

**Entities**

* **College**: id, name
* **Student**: id, student\_id, name, email, password, college\_id
* **Event**: id, title, description, event\_type, start\_time, end\_time, capacity, college\_id
* **Registration**: id, student\_id, event\_id, registered\_at
* **Attendance**: id, registration\_id, checked\_in\_at
* **Feedback**: id, registration\_id, rating, comment, submitted\_at

**Relationships**

* A **College** has many **Students** and **Events**.
* A **Student** can register for multiple **Events**.
* Each **Registration** can have one **Attendance** and one **Feedback**.

**7. API Design**

**Authentication**

* POST /api/login/ → Student login (JWT tokens).

**Students**

* POST /api/students/ → Create student.

**Events**

* GET /api/events/ → List events.
* POST /api/events/<id>/register/ → Register student to event.

**Attendance**

* POST /api/attendances/ → Mark attendance.

**Feedback**

* POST /api/feedbacks/ → Submit feedback.

**Reports**

* /api/reports/registrations\_per\_event/
* /api/reports/attendance\_percentage/
* /api/reports/average\_feedback/
* /api/reports/top\_active\_students/

**8. UI Design**

* **Navbar**: Home, Register, Feedback, Login, Logout.
* **Home Page**: Event list with IDs, type badges, and register/attendance buttons.
* **Register Page**: Student registration + Event registration.
* **Login Page**: Student login form.
* **Feedback Page**: Submit rating & comment.
* **Reports Page** (for admin): Charts with attendance and feedback stats.

**9. Security**

* JWT tokens stored in localStorage.
* Only authenticated users can:
  + Register for events.
  + Mark attendance.
  + Submit feedback.
* Password confirmation at registration.
* Future scope: Hashing passwords with Django make\_password.

**10. Future Enhancements**

* Role-based access (Student vs Admin).
* Event seat limit validation.
* Notifications for upcoming events.
* Integration with college ERP.

**11. Deviation from AI suggestions**

* While creating models it didn’t specify the id for some classes where there was an issue.
* Since I had done migrations I had to change the database and run migrations again.
* In the frontend operations where the proper urls has to be used.
* Failed in properly designing and decoding the jwt through axios.
* Serializers were improper as many fields were not present.
* Login was not basic to TokenObtainPairView.
* Faced many issues while doing frontend where many corrections were to be made.