Project Title- Campus Event Management Platform

Submission For:
Webknot Technologies – Campus Drive Assignment

Submitted By:

- Name: Jithendra N
- Email: jithendranandakumar@gmail.com
- Contact: 9482260381
- University: REVA University

This project is about a prototype implementation of Campus Event Management Platform, it is basically a web application/ platform that allows organizers, colleges and universities to create and host events and lets students to register the same.

The platform consists of two main feature **Admin panel** and **Student dashboard**. It also consists of other important technical features that makes it easier for organizers and students to easily create and participate in events and thereby accelerating and improved event experiences.

Technical Features:

Admin Features

1. Event Creation – Admins can create new events with details (title, code, type, capacity, start & end date/time).
2. Event Status Management – Ability to cancel or reactivate events.
3. Event Listing Dashboard – View all events with details like status, type, capacity, and schedule.
4. Reports Dashboard – Generate reports such as:
    - Event popularity (registrations per event).
    - Student participation (number of events registered/attended).
    - Top 3 most active students

Student Features

1. Event Browsing – Students can view all active events with details.
2. Event Registration – One-click registration for events (with duplicate prevention & capacity check).
3. Unregister from Events – Students can opt-out from registered events.
4. Attendance Marking – Students can mark attendance for events they participated in (updates to "Attended").
5. Feedback System – Students can submit ratings and comments for individual events.
6. Student Dashboard – Centralized view of:
    - Registered events.
    - Attendance marking option.
    - Feedback submission.

Platform-Wide Features

1. Authentication (Prototype level) – Basic login page with role selection (Admin/Student).
2. Navigation System – Integrated Navbar + Footer across all pages.
3. Responsive UI – Built using React + React Bootstrap, ensuring mobile-friendly layout.
4. Backend APIs (Node.js + Express) – REST APIs for events, students, registrations, attendance, feedback, and reports.
5. Database (MongoDB Atlas) – All data is stored in the cloud with proper schema models for events, students, registrations, attendance, and feedback.
6. Error Handling & Validations – Prevents:
    - Duplicate registrations.
    - Multiple feedback submissions.
    - Attendance duplication.

Tech Stack & Database Design

I have used MERN Stack to build this project.

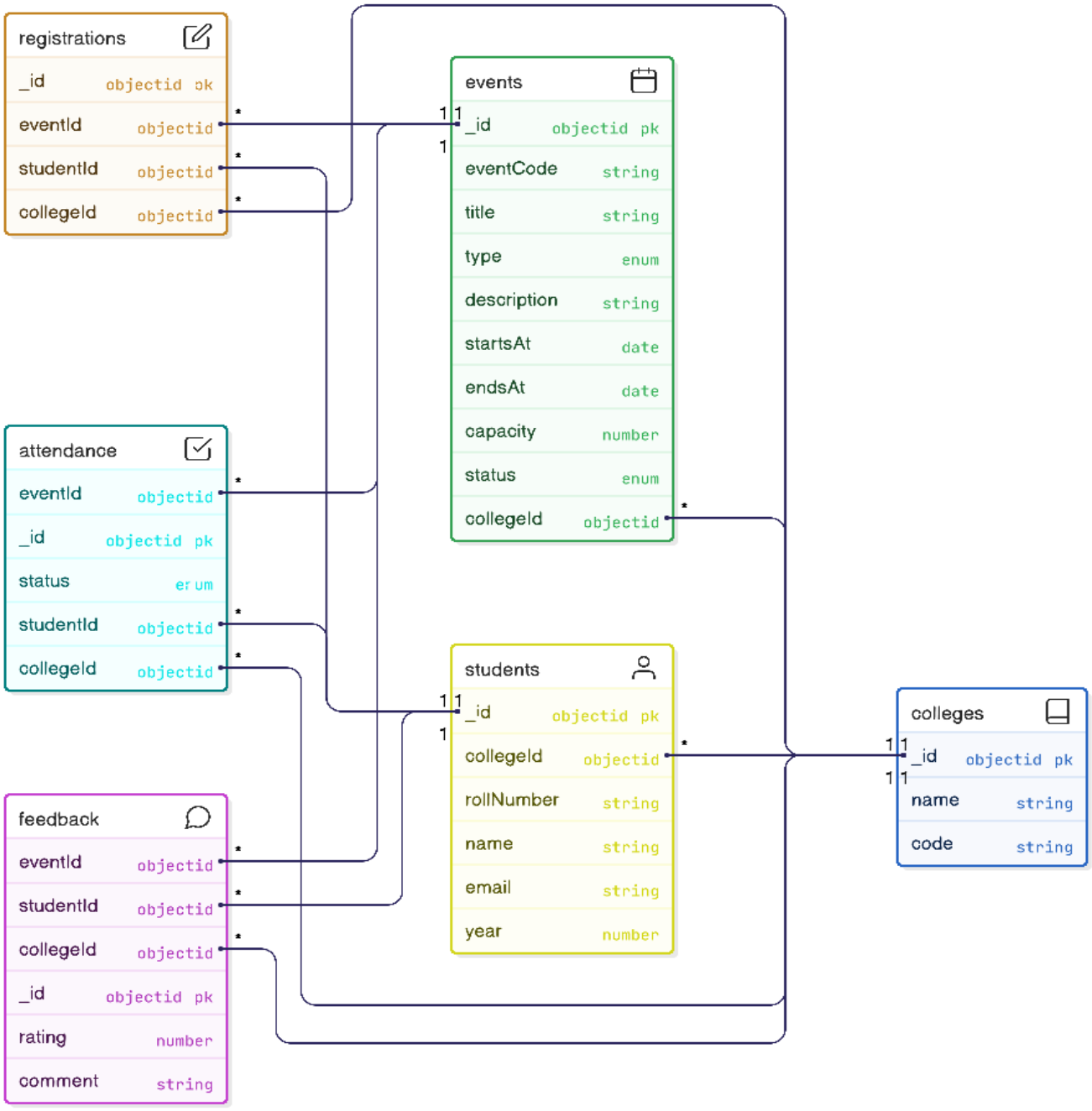
MongoDB- Database

Express+ NodeJS - Backend

ReactJS- Frontend

The reason I chose MongoDB instead of MySql/ PostgreSql is because of its flexible schema. MongoDB's **document-based schema** allows us to store variable fields without altering database schemas which you would need to do in SQL. Also MongoDB is highly scalable, for a data for 50 colleges with 200 students each it would easy to scale using MongoDB.

Database Schema.



Restful API’s

Admin / Events APIs

- POST /api/events → Create a new event (Admin only).
- GET /api/events → Get all events (with optional filters by college, type, status).
- PATCH /api/events/:id/cancel → Cancel an event.
- PATCH /api/events/:id/activate → Reactivate a cancelled event.
- DELETE /api/events/:id → Permanently delete an event.

Student APIs

- POST /api/students → Register a new student.
- GET /api/students → Fetch all students (for reports/admin).
- GET /api/students/:id → Get student profile details.

Registration APIs

- POST /api/registrations → Student registers for an event.
- GET /api/registrations → Fetch all registrations (filter by studentId/collegeId).
- DELETE /api/registrations/:id → Student unregisters from an event.

Attendance APIs

- POST /api/attendance → Mark attendance for a student at an event.
- GET /api/attendance → Fetch attendance records.

Feedback APIs

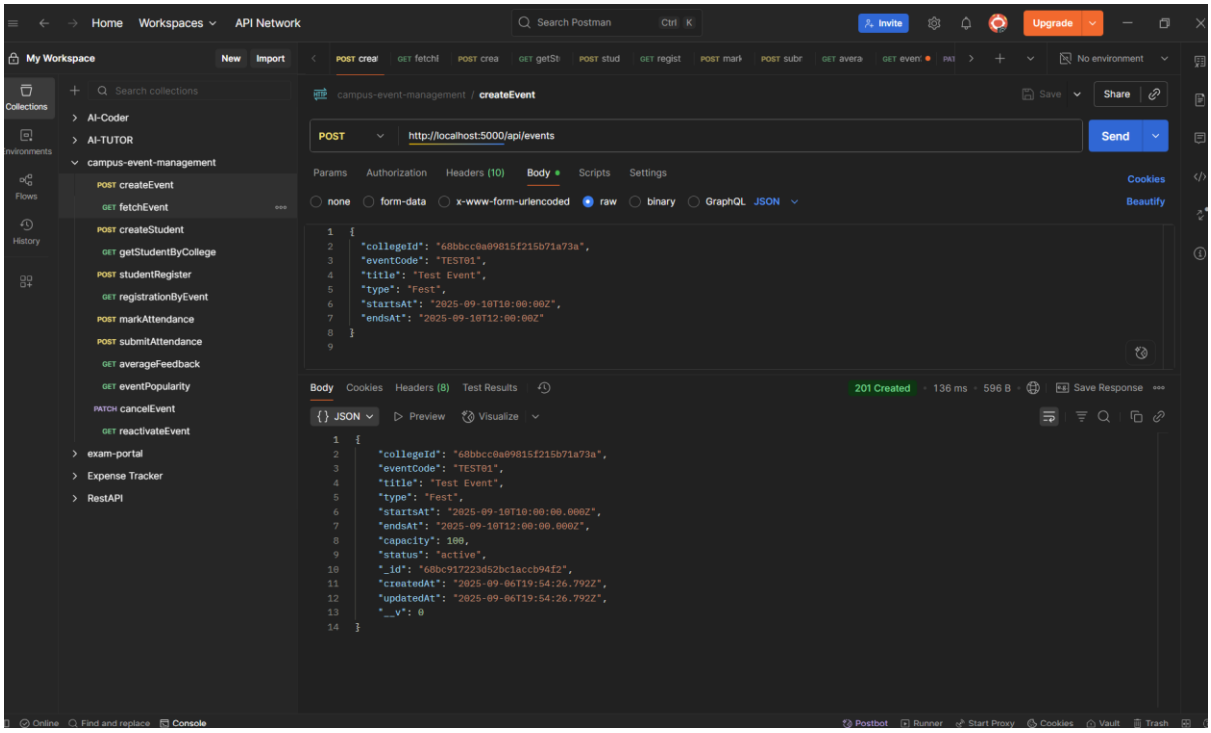
- POST /api/feedback → Submit feedback (rating + comments) for an event.
- GET /api/feedback/summary/:eventId → Get average rating + count of feedback for an event.

Reports APIs (Admin Analytics)

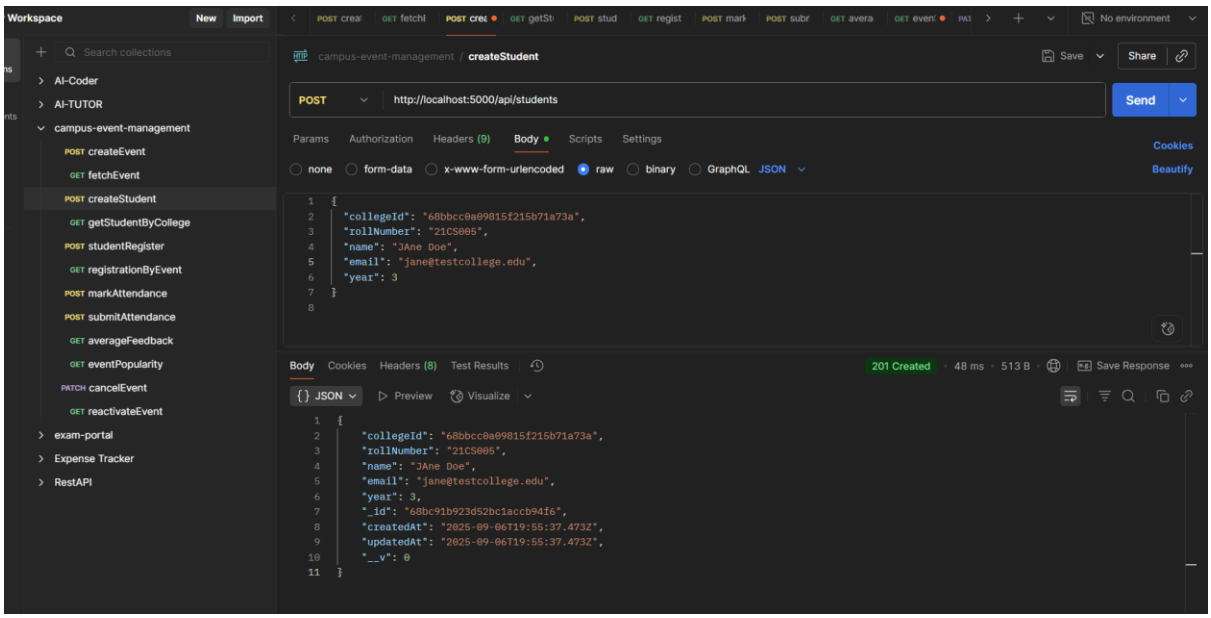
- GET /api/reports/events/popularity → Report of events with registration counts.
- GET /api/reports/students/participation → Report of student participation (events registered).
- GET /api/reports/students/top → Top 3 most active students (highest registrations).

All these API’s were implemented and tested using **Postman**

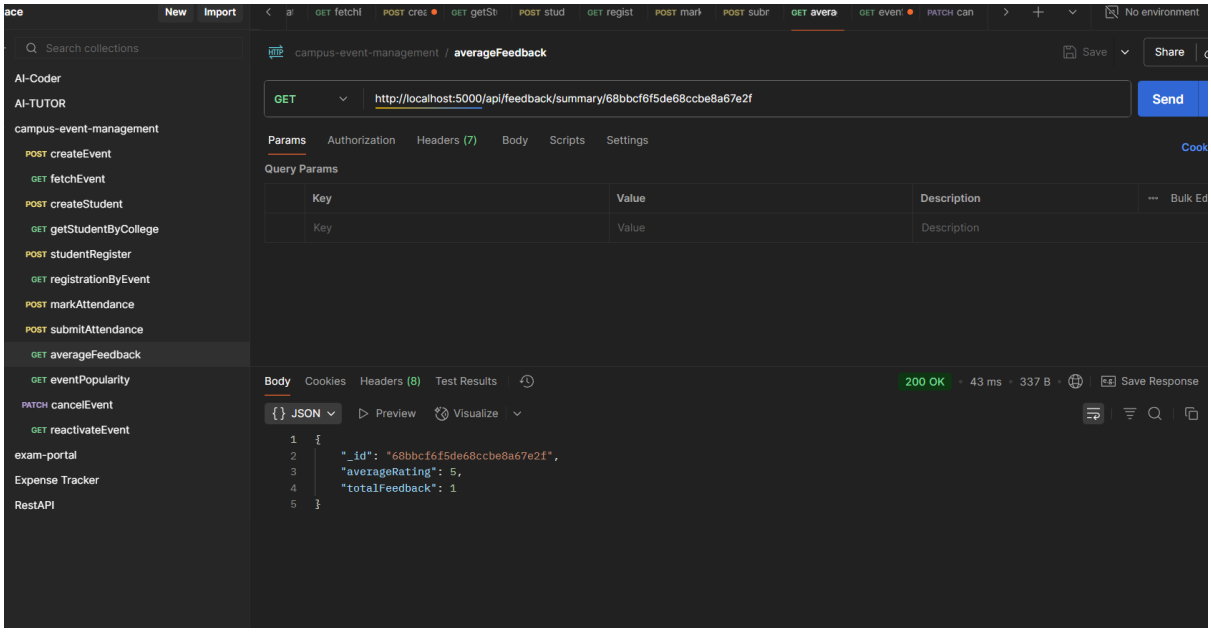
Creating Event



Creating student

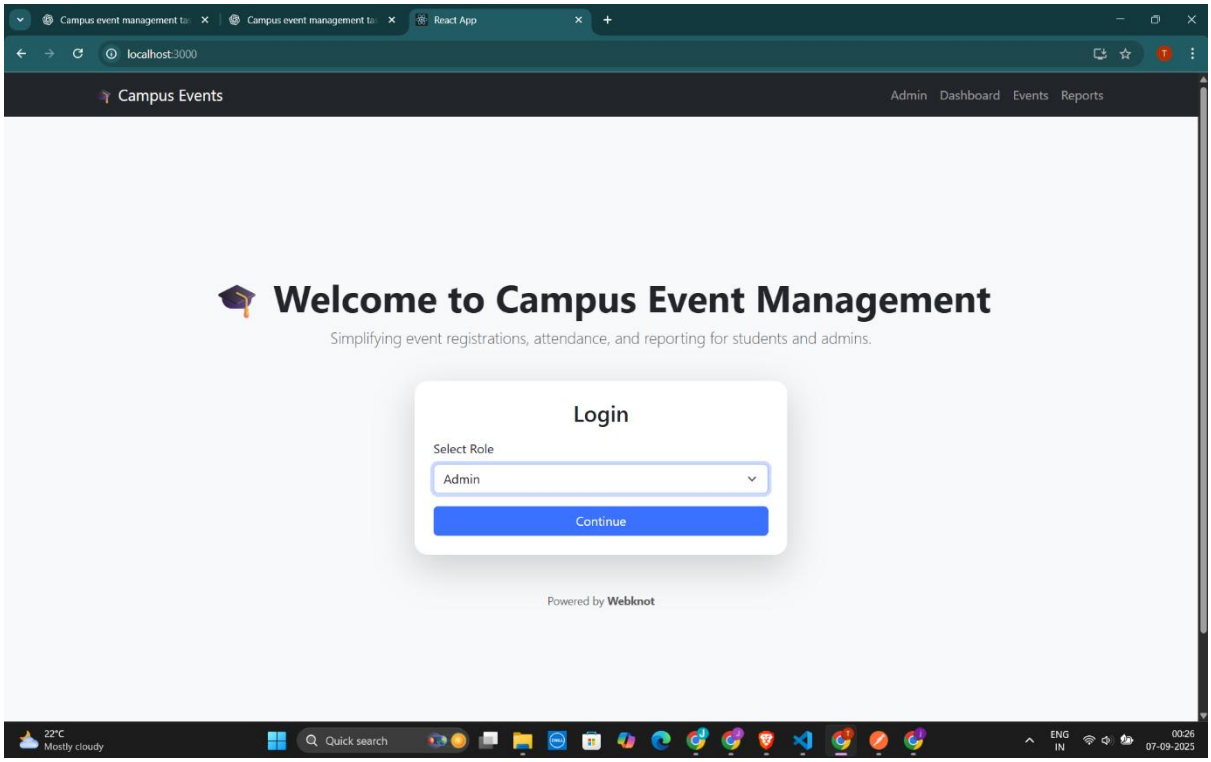


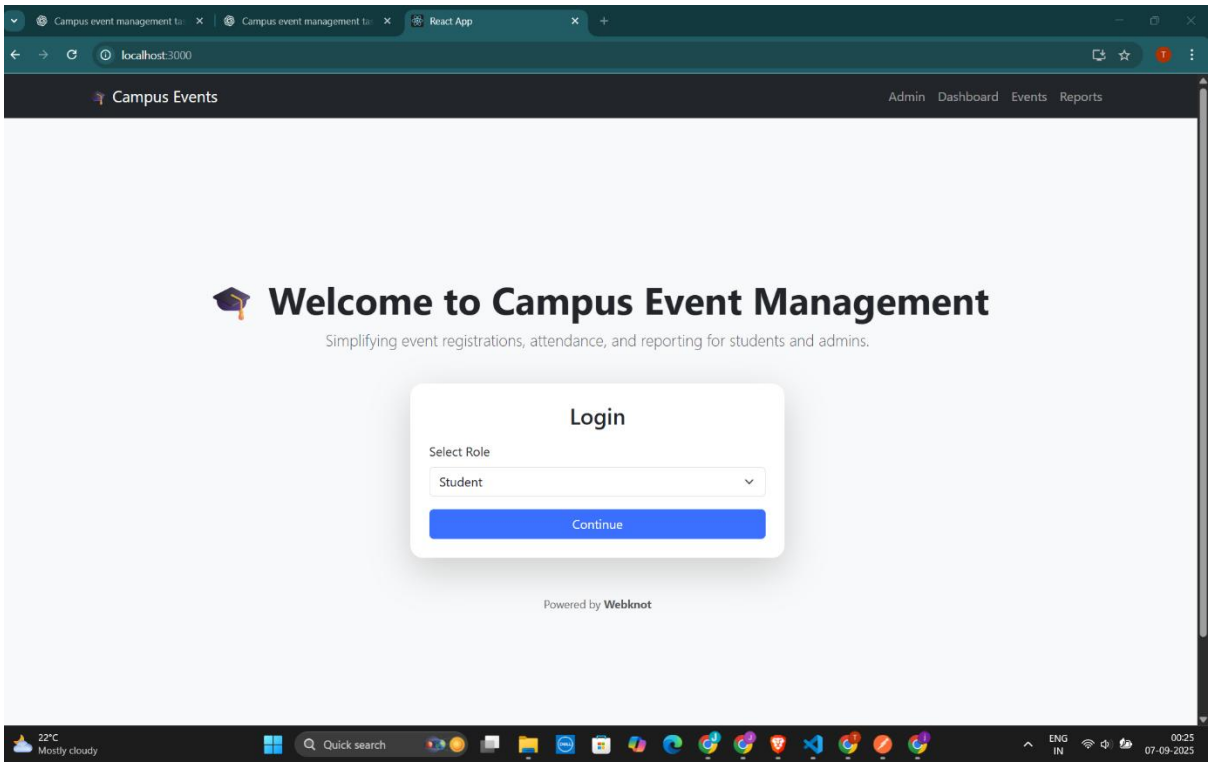
Average Feedback for Event



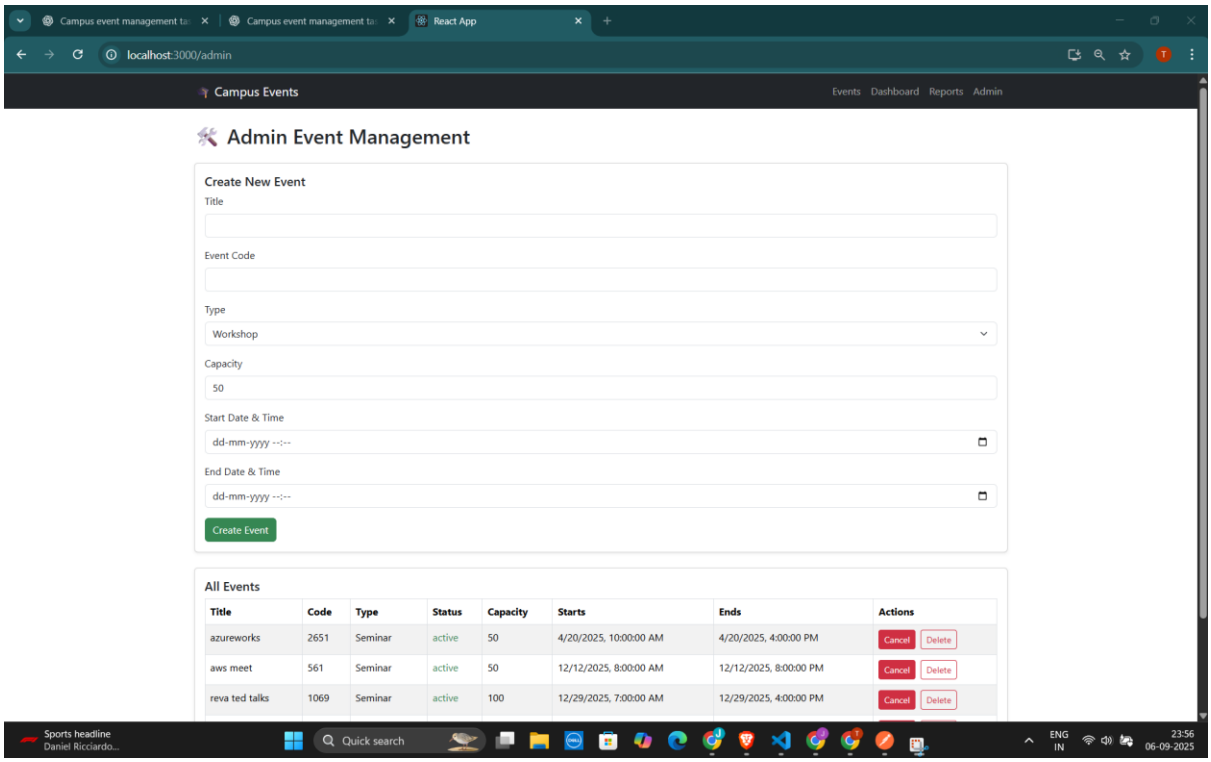
Workflows

1.Login Page- Can select login as Admin / Students.

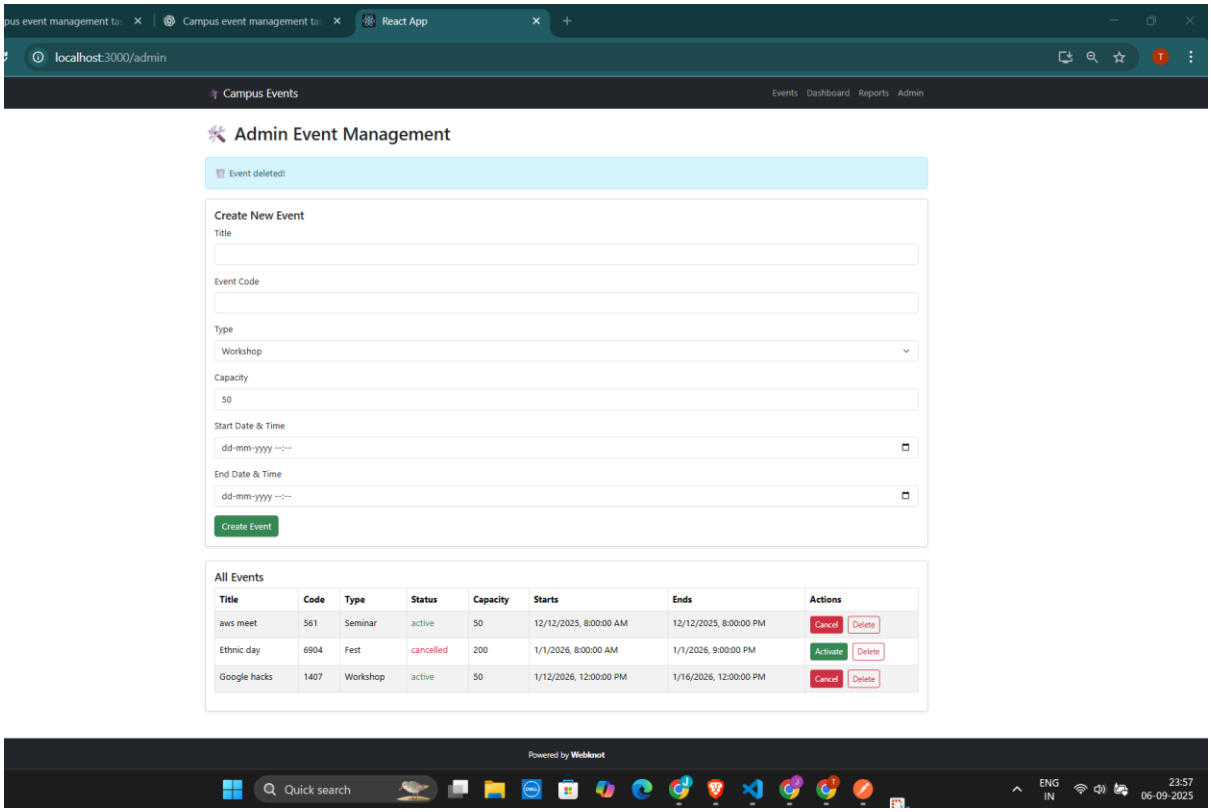




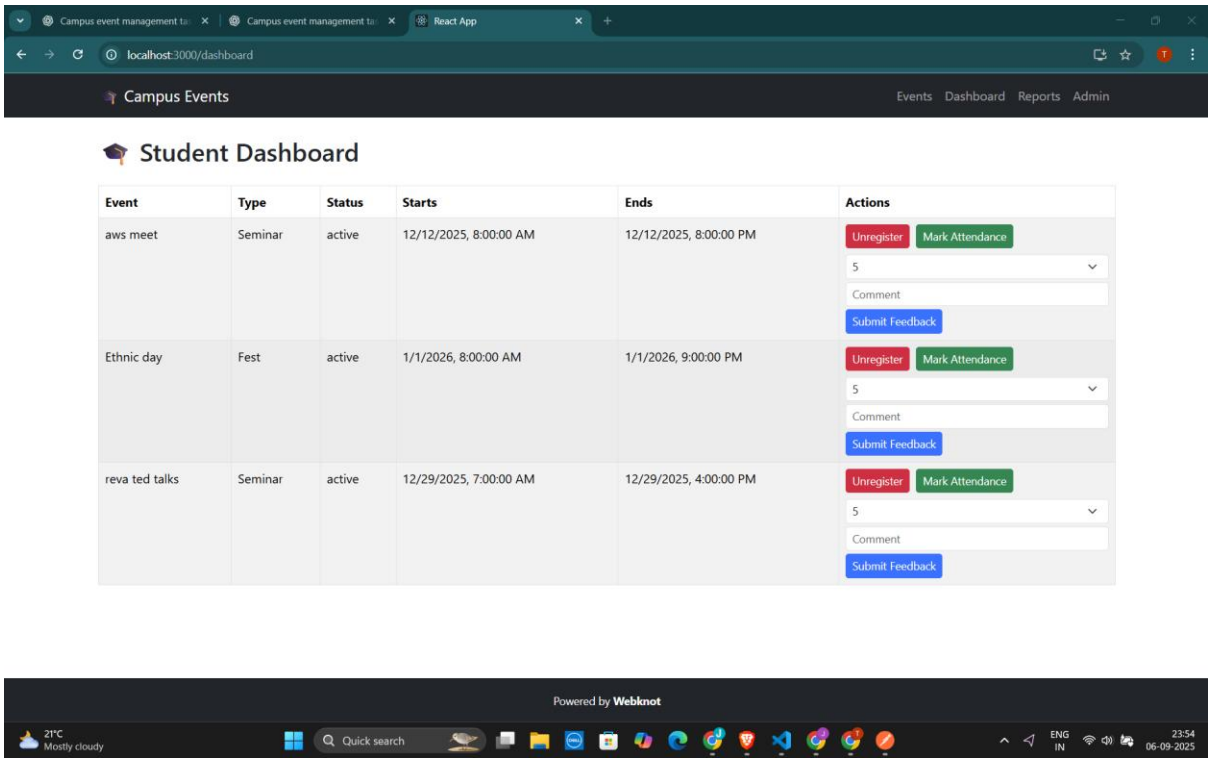
2.On selecting Admin Role, Admin Page loads.



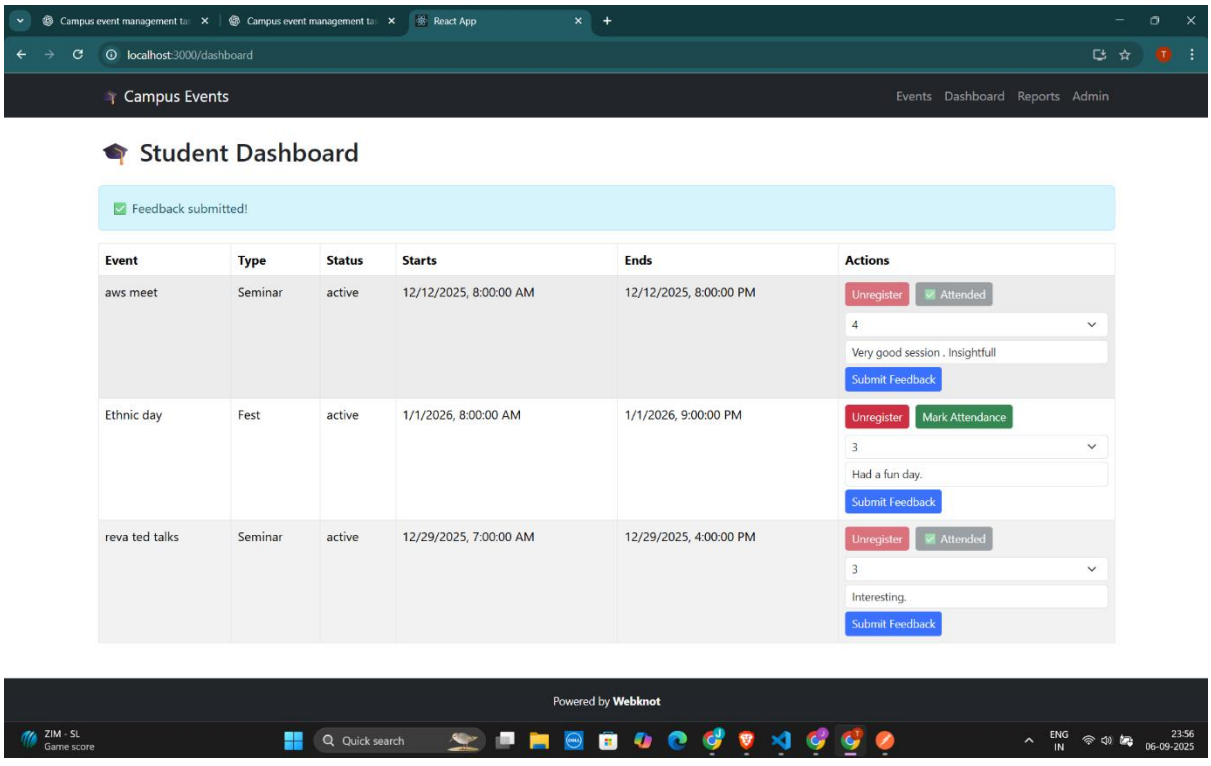
Admins have the access to Cancel or Activate events or even Delete the events.



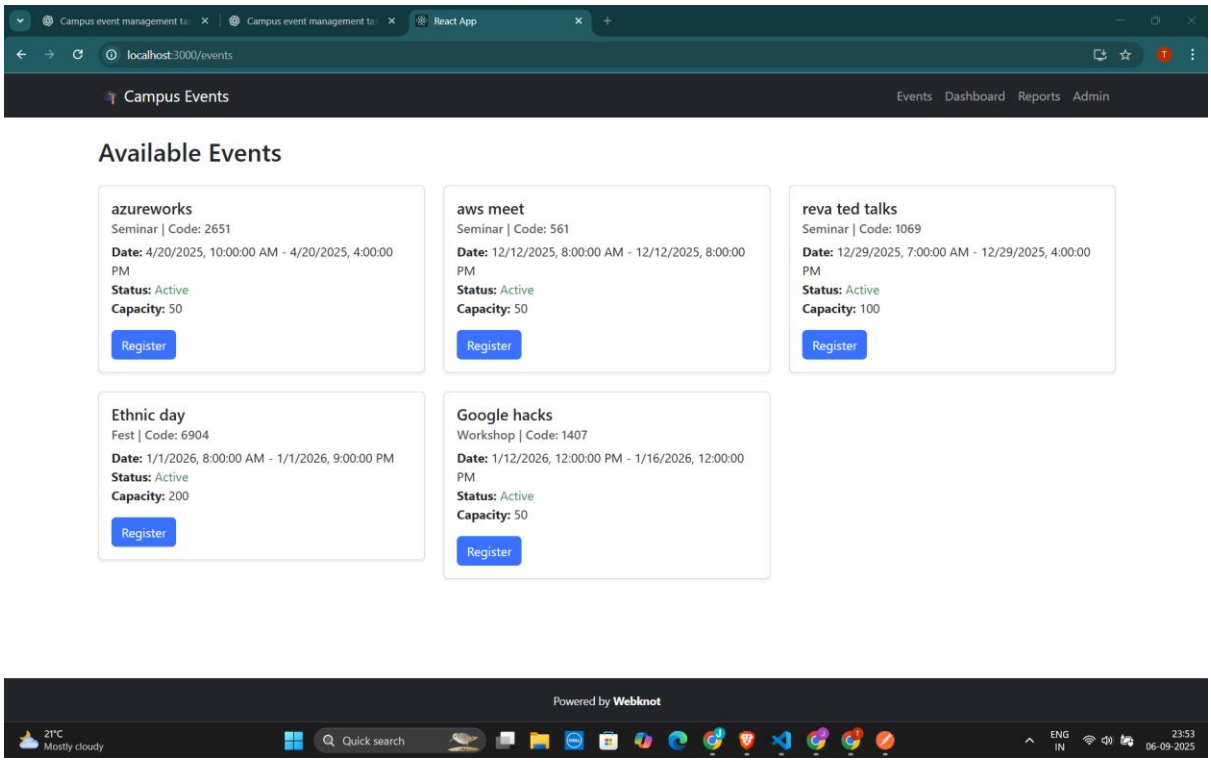
3.On Selecting Student Role in Login Page, Student Dashboard loads.



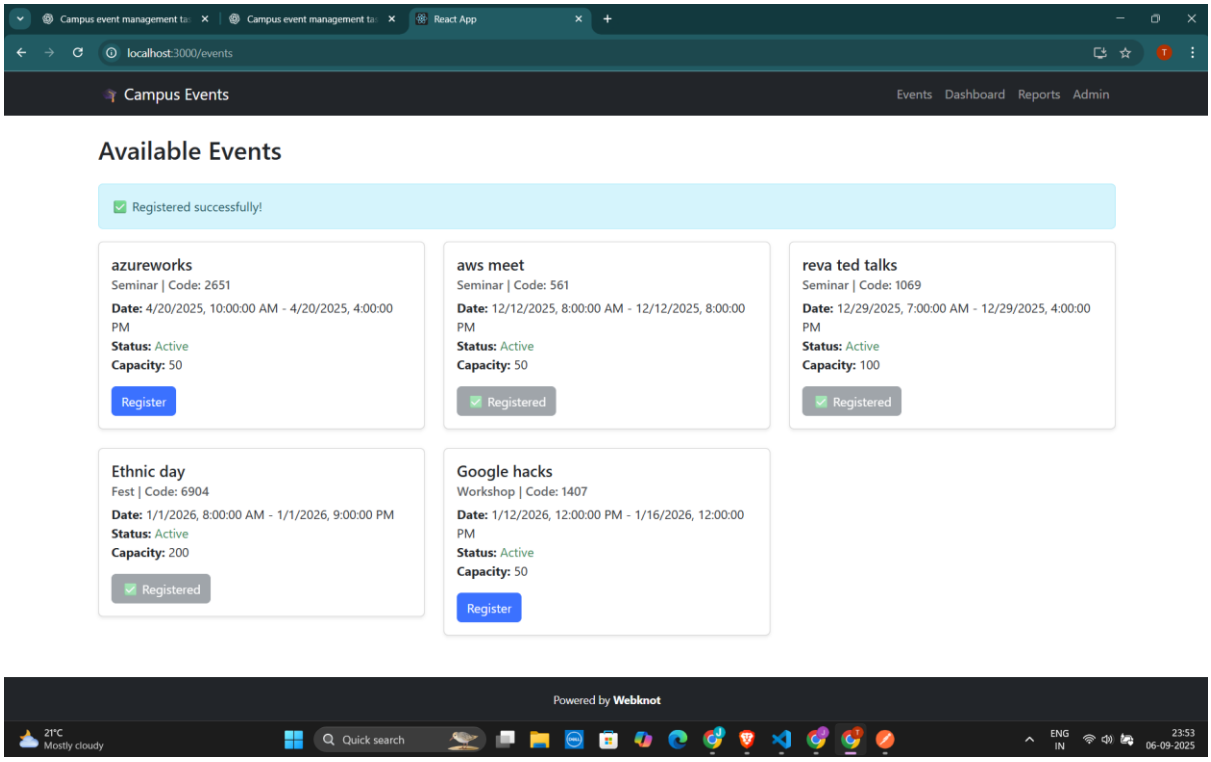
Student Dashboard allows students to see their registered events, it lets them mark their attendance, give feedback and comment. It also allows students to unregister from the event if they want to.



4.Events Page.

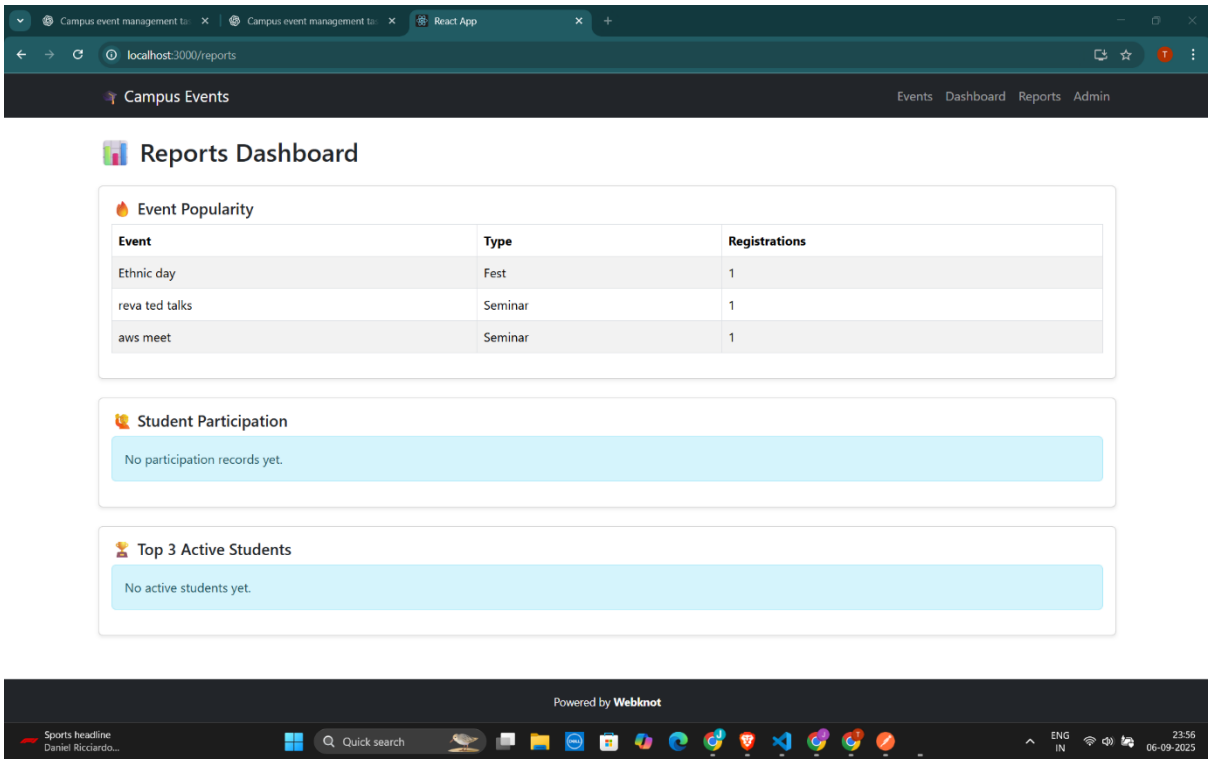


Students/Admin can see the events that are listed and can register to the event based on the capacity or status of the event.



5.Reports Dashboard.

It gives the insights and reports of the Events, Shows how popular the event is and gives the number of student participations and the top 3 active students in the community.



Assumptions

1. **Single College Context in Prototype**
   Although the assignment mentioned up to **50 colleges × 200 students**, for this prototype we assume a fixed collegeId (hardcoded in frontend). In real-world, authentication would dynamically determine collegeId.
2. **Unique Event Code per College**
   Each event within a college must have a unique eventCode.
3. **Hardcoded Student & College IDs**
   For simplicity, we used fixed student and college IDs in frontend. In real production, IDs would be tied to logged-in users via authentication.
4. **Feedback Once per Student per Event**
   A student can only submit feedback once for a given event.
5. **Attendance Allowed Only If Registered**
   A student cannot mark attendance for an event they are not registered for.

Edge Cases & How We Handled Them

1. **Duplicate Registration**
   - Case: A student tries to register for the same event twice.
   - Solution: API returns 409 Conflict (registration not created again).
2. **Event Capacity Full**
   - Case: More students try to register than allowed capacity.
   - Solution: Registration API rejects and shows "Event full" message.
3. **Cancelled Events**
   - Case: Event is cancelled by Admin.
   - Solution: Students see "Cancelled" status on both Events page & Dashboard, and cannot register or mark attendance.
4. **Attendance Marking Twice**
   - Case: A student tries to mark attendance multiple times.
   - Solution: Attendance API checks if already marked, rejects duplicate.

5. Feedback Multiple Submissions

- Case: A student submits feedback multiple times.
- Solution: Only one feedback per student per event allowed, duplicates are rejected.

6. Reports Showing Object IDs Instead of Names

- Problem: MongoDB returns _id references.
- Solution: Used Mongoose populate to fetch event titles and student names in reports.

7. Unregistering from Events

- Case: A student unregisters from an event.
- Solution: Registration entry is deleted, and student cannot mark attendance anymore.

8. UI Sync with Backend State

- Register → Button changes to "Registered".
- Cancel Event → Updates immediately on student/admin UI.
- Attendance → Changes button to "Attended".