College Event Hub

Description

College Event Hub is a comprehensive web-based platform developed using the MERN stack to streamline both intra- and inter-college event management. It serves as a unified space where administrators can effortlessly create, organize, and analyse events, while students can explore upcoming activities, register online, and provide feedback after participation. By integrating features such as event creation, online registration, role-based dashboards, email-based password recovery, and a structured feedback and analytics system, the platform eliminates manual overhead and enhances transparency. Overall, College Event Hub enables institutions to manage events more efficiently and offers students a smooth, engaging, and user-friendly experience.

Purpose

The primary purpose of College Event Hub is to transform how colleges plan, manage, and execute events while enhancing the overall student experience. The platform eliminates traditional manual processes and brings administrators, organizers, and students together on a unified digital system.

Objectives

- **Centralised Event Management**
  A single platform to manage all cultural, technical, sports, and academic events efficiently.
- **Hassle-Free Online Registration**
  Students can register for events instantly without paperwork, queues, or manual confirmation.
- **Streamlined Admin Workflow**
  Administrators can create events, monitor participation, approve registrations, manage student data, and access analytics all from one dashboard.
- **Enhanced Student Engagement**
  Students can explore upcoming events, view details, track registration status, and stay updated with important notifications.
- **Feedback-Driven Quality Improvement**
  Students share ratings and comments after events, enabling organizers to continuously improve event quality based on real user insights.
- **Secure Role-Based Access**
  Distinct permissions for Admins, Organizers, and Students ensure secure and controlled access, powered by JWT authentication.

Getting Started

Prerequisites

Before setting up the project, ensure the following are installed:

- **Node.js (LTS) and npm** – For running backend and frontend.
- **MongoDB** – Local instance or Atlas cluster.
- **Visual Studio Code** or any IDE.
- **SMTP Email Account** – e.g., Gmail with App Password for sending OTP and notifications.

Installation

Follow these steps to set up the project:

Clone the Repository git clone
https://github.com/springboardmentor0316/Batch4_CollegeEventHub_Team4 .git
cd college-event-hub

Install Dependencies
npm install

Running the Application

Start the Backend

Navigate to the backend directory:

cd backend

Run the backend server:

npm start

Start the Frontend

Navigate to the frontend directory:

cd frontend

Run the development server:

npm start

Accessing the Application

Student / Admin UI (Frontend):

http://localhost:3000

Backend API Base URL:
http://localhost:5000/api

Environment Setup

To run the project seamlessly, you need to set up environment variables for both the backend and frontend.
Create a .env file in the root directory of the backend project and add the required configurations as shown below.

Required Environment Variables

Variable Name	Description	Example Value
PORT	Port number where backend server runs	1000
MONGO_URI	Connection string for the main database	mongodb://localhost:27017/eventhub
JWT_SECRET	Secret key used for signing JWT tokens	supersecretkey123
JWT_EXPIRES_IN	JWT token expiration time	7d
CLIENT_URL	URL of frontend client	http://localhost:3000
EMAIL_HOST	SMTP server host name	smtp.gmail.com
EMAIL_PORT	SMTP service port	587
EMAIL_USER	Email ID for sending OTP/reset emails	your-email@gmail.com
EMAIL_PASS	App password or SMTP password	your-email-app-password

Example .env Configuration

Below is an example .env file configuration for **College Event Hub**:

MongoDB Configuration

```
PORT=1000
MONGO_URI=mongodb://localhost:27017/eventhub
```

JWT Configuration

```
JWT_SECRET=supersecretkey123
JWT_EXPIRES_IN=7d
```

Frontend URL

```
CLIENT_URL=http://localhost:3000
```

SMTP Configuration

```
EMAIL_HOST=smtp.gmail.com
EMAIL_PORT=587
EMAIL_USER=your-email@gmail.com
EMAIL_PASS=your-email-app-password
```

Steps to Set Up the Environment

- Create a .env file in the root directory of your backend project.
- Copy the example configuration above and paste it into the .env file.
- Replace placeholder values (like your-email@gmail.com and your-email-app-password) with your actual credentials.
- Make sure MongoDB is installed and running on your machine.
- Ensure the frontend REACT_APP_API_URL (if used) points to your backend URL.

Additional Notes

- For cloud-hosted MongoDB (e.g., MongoDB Atlas), replace:
- with your actual cluster connection string.
- If using a third-party email service, update:
    - EMAIL_HOST
    - EMAIL_PORT
    - EMAIL_USER
    - EMAIL_PASS
- Keep the .env file secure and **never** commit it to version control.
  Add it to .gitignore to prevent accidental uploads.

Folder Structure

```
college-event-hub/
|
├── backend/                 # Backend (Node.js + Express)
|   ├── node_modules/          # Backend dependencies
|   ├── config/              # Database configuration, environment setup
|   |   └── db.js            # MongoDB connection file
|   ├── controllers/           # Backend controllers (auth, events, feedback)
|   |   ├── authController.js
|   |   ├── eventController.js
|   |   ├── registrationController.js
|   |   ├── statsController.js
|   |   └── feedbackController.js
|   ├── middleware/            # Authentication & Role-based access middleware
|   |   ├── auth.js
|   |   └── role.js
|   ├── models/              # Mongoose database models
|   |   ├── User.js
|   |   ├── Event.js
|   |   ├── Registration.js
|   |   └── Feedback.js
```

```
│   ├── routes/                  # API route definitions
│   │   ├── auth.js
│   │   ├── event.js
│   │   ├── registrations.js
│   │   ├── stats.js
│   │   └── feedback.js
│   ├── utils/                  # Helper functions (email, token, etc.)
│   │   └── mailer.js
│   ├── server.js                # Backend entry point
│   ├── .env                    # Environment variables for backend
│   ├── package.json              # Backend scripts & dependencies
│   └── package-lock.json          # Dependency lock file
│
├── frontend/                    # Frontend (React.js)
│   ├── node_modules/              # Frontend dependencies
│   ├── public/                  # Static files (HTML, images, icons)
│   ├── src/
│   │   ├── components/            # Reusable React components
│   │   │   ├── Navbar.jsx
│   │   │   ├── StudentLayout.jsx
│   │   │   ├── AdminLayout.jsx
│   │   │   ├── FeedbackModal.jsx
│   │   │   └── ErrorBoundary.jsx
│   │   ├── pages/               # Main application screens/pages
│   │   │   ├── Login.jsx
│   │   │   ├── Signup.jsx
│   │   │   ├── ForgotPassword.jsx
│   │   │   ├── OtpVerification.jsx
│   │   │   ├── ResetPassword.jsx
│   │   │   ├── LandingPage.jsx
│   │   │   ├── AdminDashboard.jsx
│   │   │   ├── CreateEvent.jsx
│   │   │   ├── AdminEvents.jsx
│   │   │   ├── ManageStudents.jsx
│   │   │   ├── AdminAnalytics.jsx
│   │   │   ├── StudentDashboard.jsx
│   │   │   ├── AllEvents.jsx
│   │   │   ├── EventDetails.jsx
│   │   │   ├── PastEvents.jsx
│   │   │   └── MyRegistrations.jsx
│   │   ├── utils/               # API helper functions
│   │   │   └── api.js
│   │   ├── App.js               # Main React root component
│   │   ├── App.css               # Global styles
│   │   └── index.js             # React entry point
│   ├── package.json              # Frontend scripts & dependencies
│   └── package-lock.json          # Dependency lock file
```

```
│
├── README.md                   # Project documentation
└── .gitignore                  # Git ignore file
```

Key Features

- **User Authentication & Authorization**
    - JWT-based login and signup.
    - Role support: ADMIN, STUDENT, ORGANIZER.
    - Protected routes on backend & restricted pages on frontend.
- **OTP-Based Password Reset**
    - Students and admins can recover accounts using email OTP.
    - OTP sent through SMTP (Gmail or other SMTP provider).
- **Admin Dashboard**
    - Quick stats: total events, registrations, students.
    - Recent events list.
    - Access to events management, students, and analytics.
- **Event Management**
    - Create, edit, delete events.
    - Fields: title, date, time, location, category, college, eligibility, prizes, entry fee, poster.
    - Status management (approved / pending / rejected).
- **Student Experience**
    - Browse all events.
    - Filter by category, college, or date.
    - Register for events and view status.
    - View past events and provide feedback.
- **Registration & Slot Management**
    - Students register online.
    - Admin can track number of registrations and manage them through dedicated views.
    - Cancel registration functionality for students (as per code).
- **Feedback System**
    - Students who attended past events can rate them (1–5 stars) and add comments.
    - One feedback per user per event (update allowed).
    - Average rating and review count shown in Event Details.
    - Full feedback list visible to users.
- **Analytics**
    - Basic admin statistics: counts of events by status, registrations, user stats.
    - Can be extended later for advanced charts.

Project Components

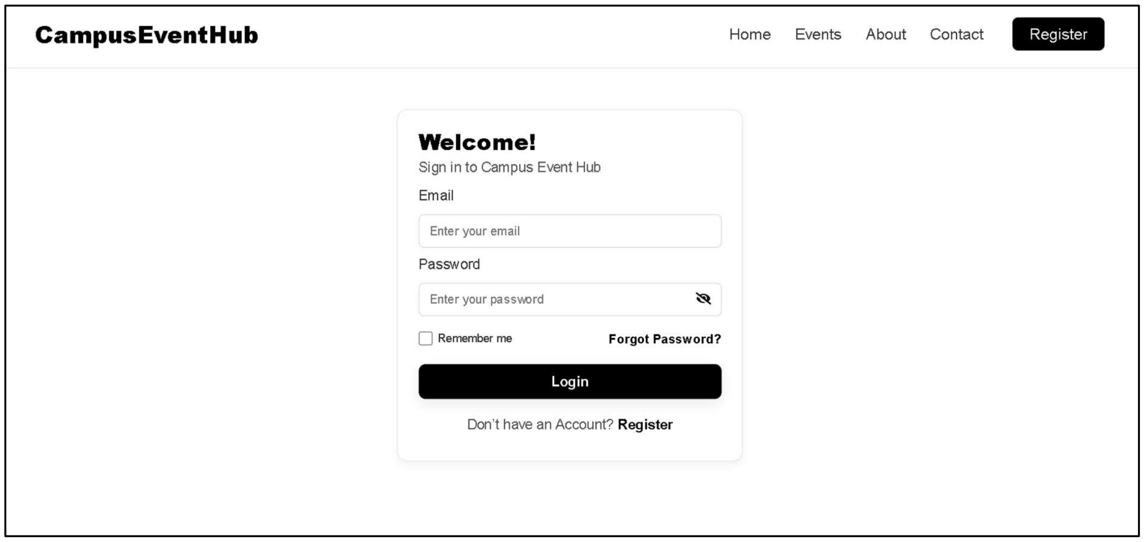
Authentication Components :

<u>Login Component</u>

Description:
The Login Component handles secure authentication for both students and administrators. It provides a smooth and interactive login experience while ensuring proper validation and security checks.

Key Features:

- **Password Visibility Toggle**
  Allows users to show or hide their password while typing for a better user experience.
- **Role-Based Access Control**
  After successful authentication, users are redirected to their respective dashboards (Student Dashboard or Admin Dashboard) based on their assigned role.
- **Form Validation**
  Validates user input such as email format and required fields before sending the request to the backend.
- **Error Handling**
  Displays clear and user-friendly error messages for incorrect credentials, missing fields, or server issues.

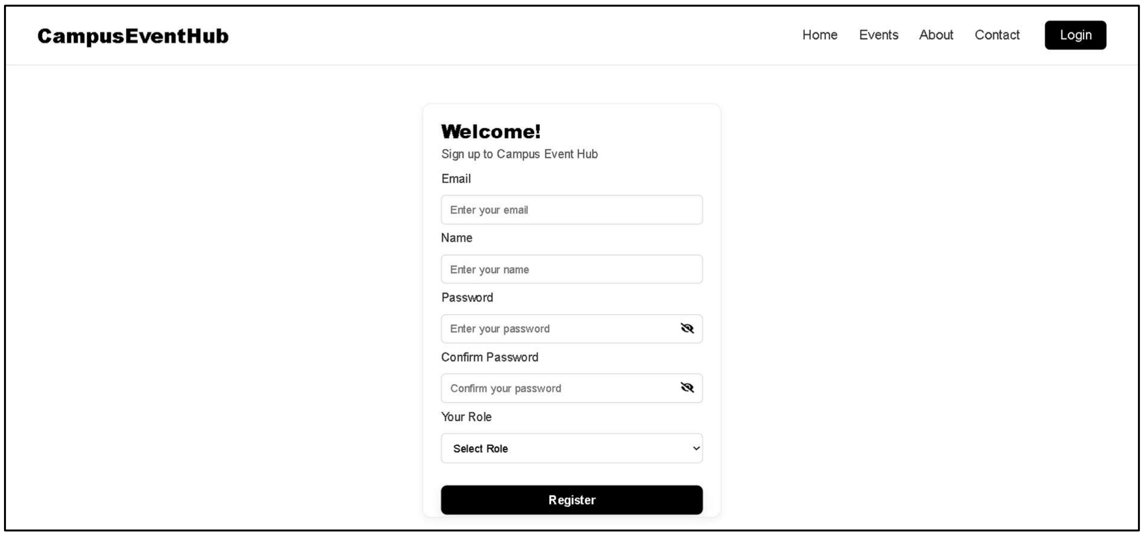


<u>Signup Page</u>

Description:
The Signup Page enables new users—both students and administrators—to create an account on the platform.

Key Features:

- **User Registration**
  Allows new users to sign up by providing essential information such as:
  - Full Name
  - Email Address
  - College / Institution
  - Department / Year (optional and based on role)
  - Password
  - User Role (Student or Admin)
- **Input Validation**
  Ensures all required fields are filled and validates email format before account creation.
- **Secure Account Creation**
  Passwords are encrypted using bcrypt before being stored in the database.

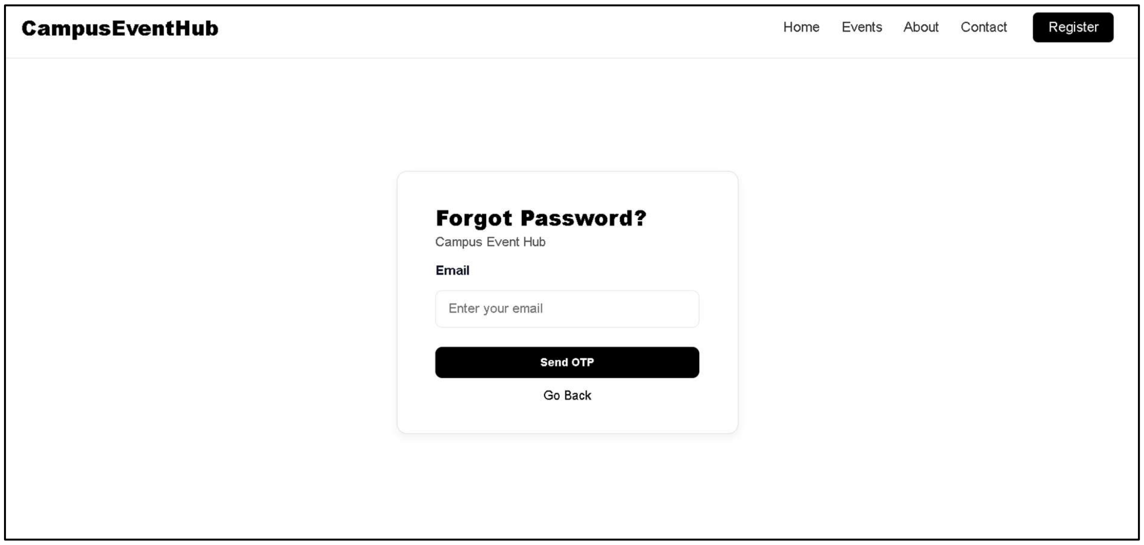


Forgot Password / OTP / Reset Password Flow

The password recovery system consists of a secure three-step OTP-based workflow powered by Nodemailer and the authentication controller.

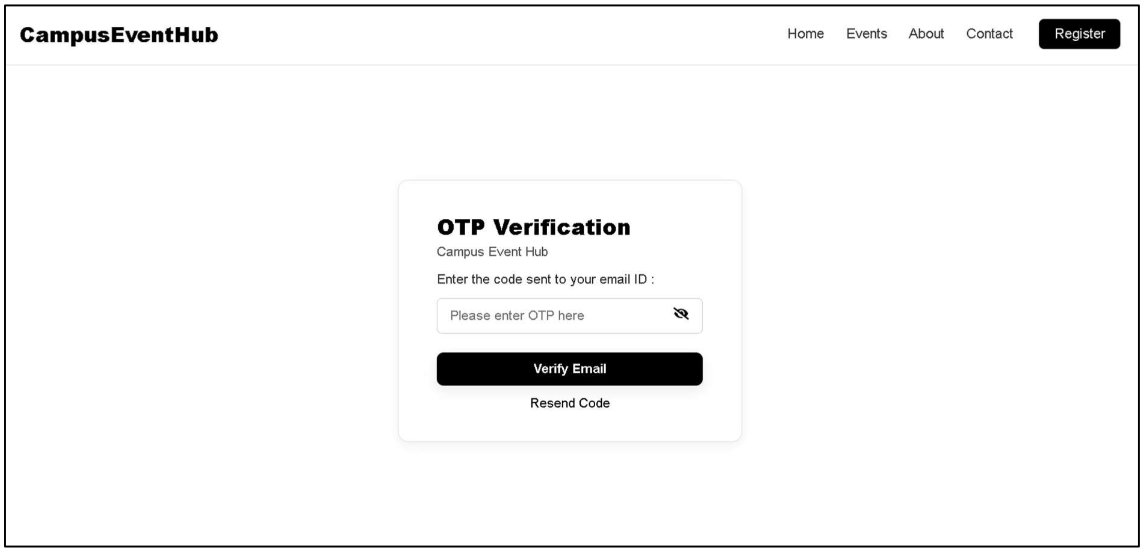
1. Forgot Password

- Users enter their registered email address.
- Backend triggers **POST /api/auth/send-otp**, which generates and sends a One-Time Password (OTP) to the user's email.
- OTP is stored securely in memory (temporary storage with auto-expiry).



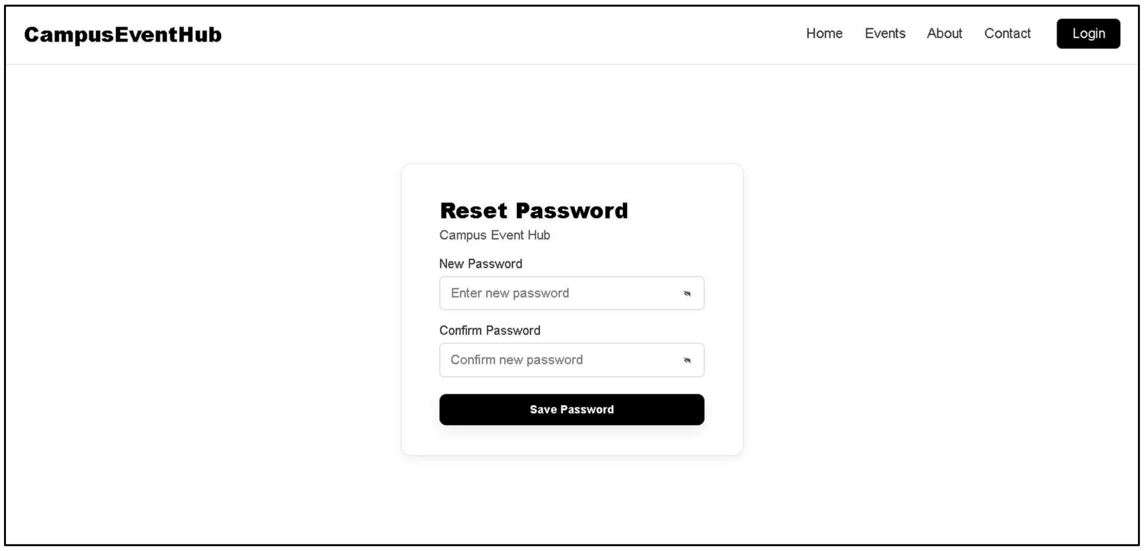
2. OTP Verification

- Users submit the OTP received in their email.
- Backend verifies the code using **POST /api/auth/verify-otp**.
- Only valid and non-expired OTPs are accepted.



3. Reset Password

- After OTP verification, users are allowed to set a new password.
- New password is updated securely in the database using **POST /api/auth/reset-password**.
- Password hashing ensures user security.


CampusEventHub

Home Events About Contact Login

Reset Password

Campus Event Hub

New Password

Enter new password

Confirm Password

Confirm new password

Save Password

Dashboards :

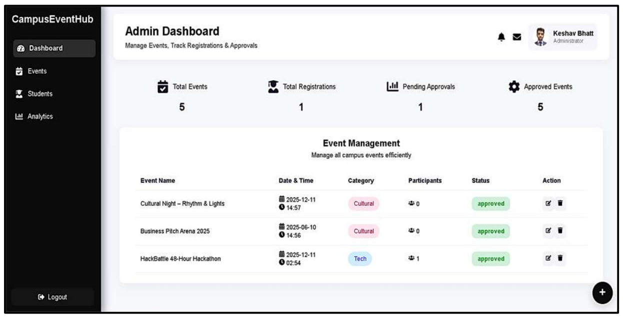
Admin Dashboard

Description:

The Admin Dashboard provides a centralized control panel for college administrators to monitor activities and manage events efficiently.

Key Features:

- **Overall System Overview**
  Displays total events, total registrations, active users, and other key statistics at a glance.
- **Quick Metrics Cards**
  Visual cards showing important numbers such as approved events, pending registrations, and total participants.
- **Centralized Control Panel**
  Helps administrators streamline workflows, manage users, and track event performance in real time.


CampusEventHub

Dashboard
Events
Students
Analytics
Logout

Admin Dashboard

Manage Events, Track Registrations & Approvals

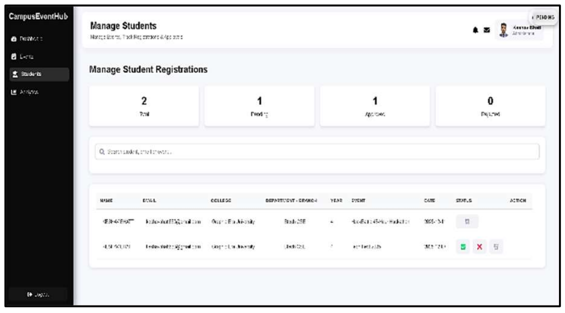
Keshav Bhatt
Administrator

Total Events	Total Registrations	Pending Approvals	Approved Events
5	1	1	5

Event Management

Manage all campus events efficiently

Event Name	Date & Time	Category	Participants	Status	Action
Cultural Night – Rhythm & Lights	2025-12-11 14:57	Cultural	0	approved	
Business Pitch Arena 2025	2025-06-10 14:56	Cultural	0	approved	
HackBattle 48-Hour Hackathon	2025-12-11 02:54	Tech	1	approved	


CampusEventHub

Students

Manage Students

Manage Student Registrations

2	1	1	0
Total	Pending	Approved	Rejected

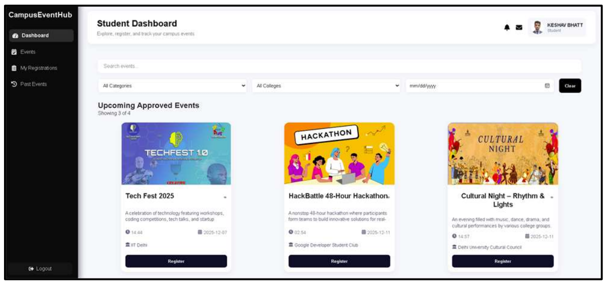
NAME	EMAIL	COLLEGE	DEPARTMENT / BRANCH	YEAR	EVENT	DATE	STATUS	ACTION
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]		
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]		

Student Dashboard

Description:
The Student Dashboard is designed to provide students with a personalized and user-friendly space to explore and track events.

Key Features:

- **Upcoming Events Overview**
  Displays a list of upcoming events based on date, helping students stay informed and plan ahead.
- **My Registrations Section**
  Shows events the student has registered for along with their approval status (Pending, Approved, Rejected).
- **New Event Highlights**
  Recently added or trending events are showcased to improve student engagement and participation.



Student Dashboard
Explore, register, and track your campus events

Search events...

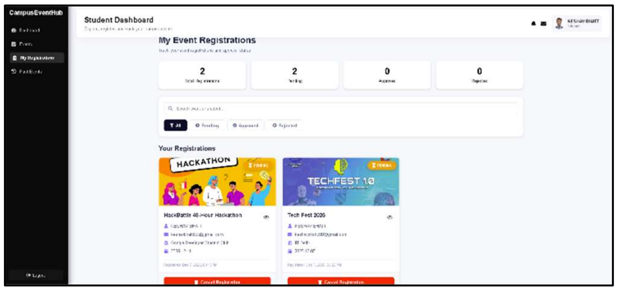
All Categories | All Colleges | mm/dd/yyyy

Upcoming Approved Events
Showing 3 of 4

Tech Fest 2025
A celebration of technology featuring workshops, coding competitions, tech talks, and startup
14:44 | 2025-12-07
IIT Delhi

HackBattle 48-Hour Hackathon
A nonstop 48-hour hackathon where participants form teams to build innovative solutions for real-
02:54 | 2025-12-11
Google Developer Student Club

Cultural Night – Rhythm & Lights
An evening filled with music, dance, drama, and cultural performances by various college groups.
14:57 | 2025-12-11
Delhi University Cultural Council



Student Dashboard

My Event Registrations

2 Total Registrations | 2 Pending | 0 Approved | 0 Rejected

Your Registrations

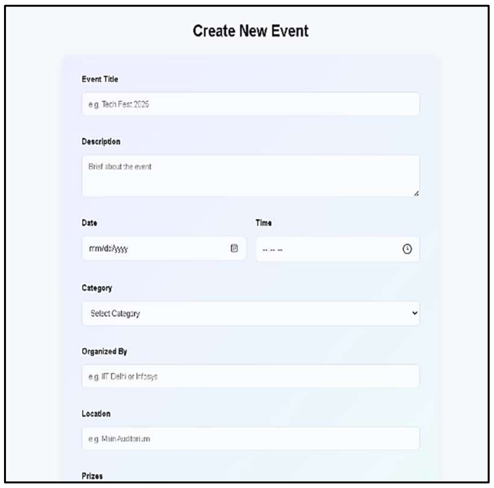
HackBattle 48-Hour Hackathon

Tech Fest 2025

Event Management Components

Create Event

- Form to create a new event (admin/organizer).
- Fields: title, description, category, date, time, location, entry fee, eligibility, prizes, poster.
- Validation for essential fields.
- Calls POST /api/events.



Create New Event

Event Title
e.g. Tech Fest 2025

Description
Brief about the event

Date
mm/dd/yyyy

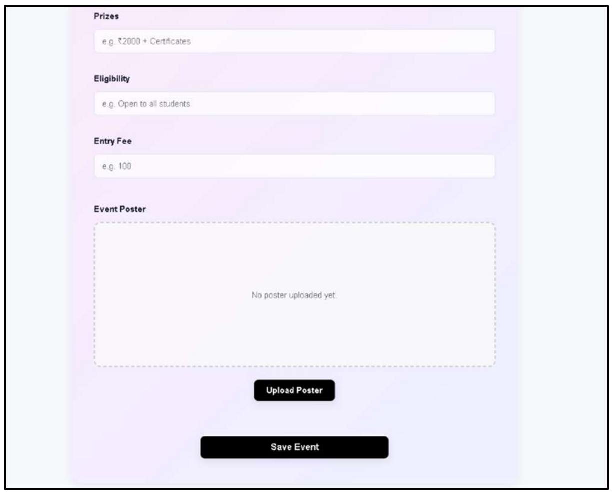
Time
-- : -- --

Category
Select Category

Organized By
e.g. IIT Delhi or Infosys

Location
e.g. Main Auditorium

Prizes



Prizes
e.g. ₹2000 + Certificates

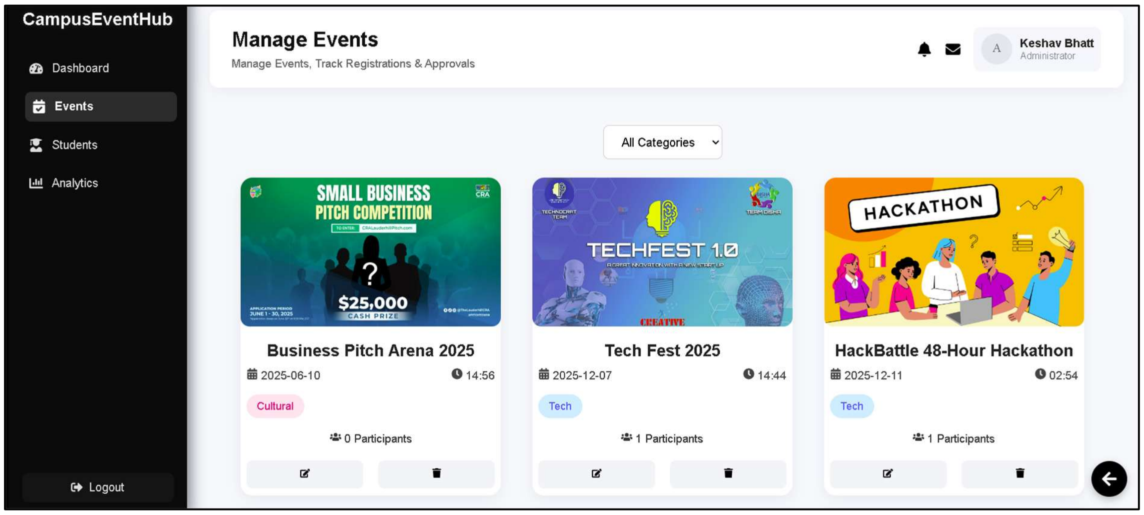
Eligibility
e.g. Open to all students

Entry Fee
e.g. 100

Event Poster
No poster uploaded yet

Admin Events List

- Shows all events created.
- Actions to edit or delete events.
- Can filter by status.



Manage Events

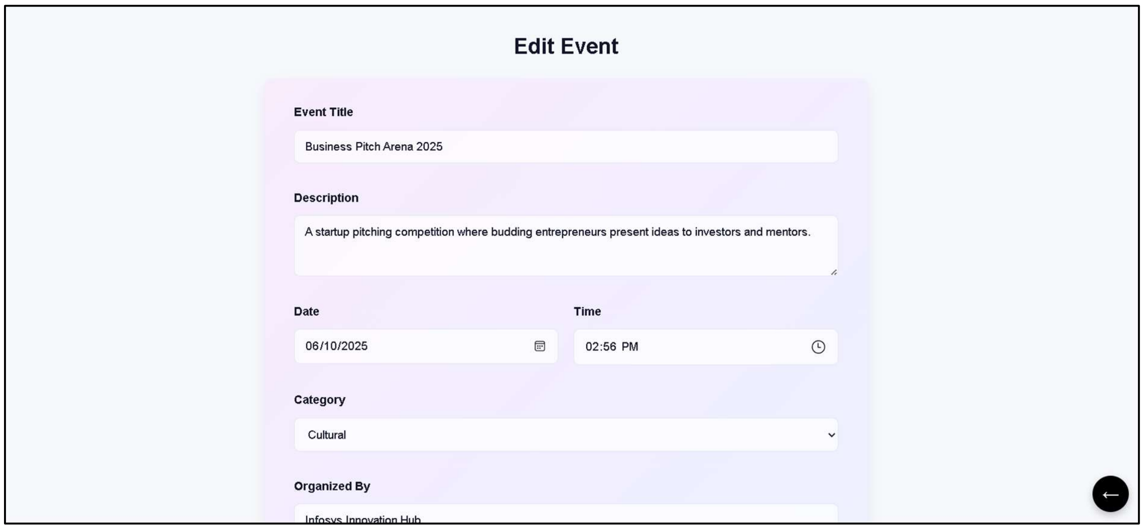
Manage Events, Track Registrations & Approvals

Event	Date	Time	Category	Participants
Business Pitch Arena 2025	2025-06-10	14:56	Cultural	0 Participants
Tech Fest 2025	2025-12-07	14:44	Tech	1 Participants
HackBattle 48-Hour Hackathon	2025-12-11	02:54	Tech	1 Participants

Event Details

Detailed page for a single event:

- Description, date, time, location, category, eligibility, prizes, entry fee.
- Poster as background.
- Total registrations count.
- Rating display (average stars and number of reviews).
- Student Feedback section with list of comments.
- Register Now and Write Feedback buttons.



Edit Event

Event Title: Business Pitch Arena 2025

Description: A startup pitching competition where budding entrepreneurs present ideas to investors and mentors.

Date: 06/10/2025

Time: 02:56 PM

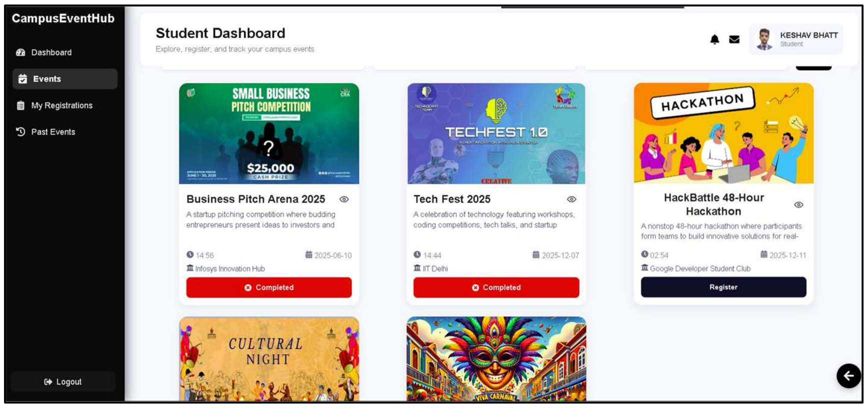
Category: Cultural

Organized By: Infosys Innovation Hub

<u>Student Event Components</u>

All Events

- Grid of events with poster, title, date/time, college.
- Filters for:
    - Category
    - College
    - Date
    - Registered / Upcoming / Past



CampusEventHub

Student Dashboard

Explore, register, and track your campus events

Business Pitch Arena 2025

A startup pitching competition where budding entrepreneurs present ideas to investors and

14:56 2025-06-10

Infosys Innovation Hub

Tech Fest 2025

A celebration of technology featuring workshops, coding competitions, tech talks, and startup

14:44 2025-12-07

IIT Delhi

HackBattle 48-Hour Hackathon

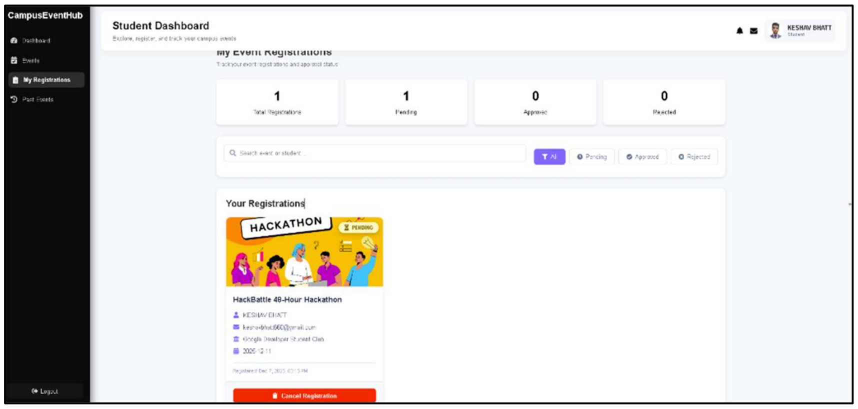
A nonstop 48-hour hackathon where participants form teams to build innovative solutions for real-

02:54 2025-12-11

Google Developer Student Club

My Registrations

- Shows events the current student has registered for.
- Status badges: pending, approved, rejected (based on stored status).



CampusEventHub

Student Dashboard

Explore, register, and track your campus events

My Event Registrations

Track your event registrations and approval status

1 Total Registrations

1 Pending

0 Approved

0 Rejected

Your Registrations

HackBattle 48-Hour Hackathon

KESHAV BHATT

keshavbhatt660@gmail.com

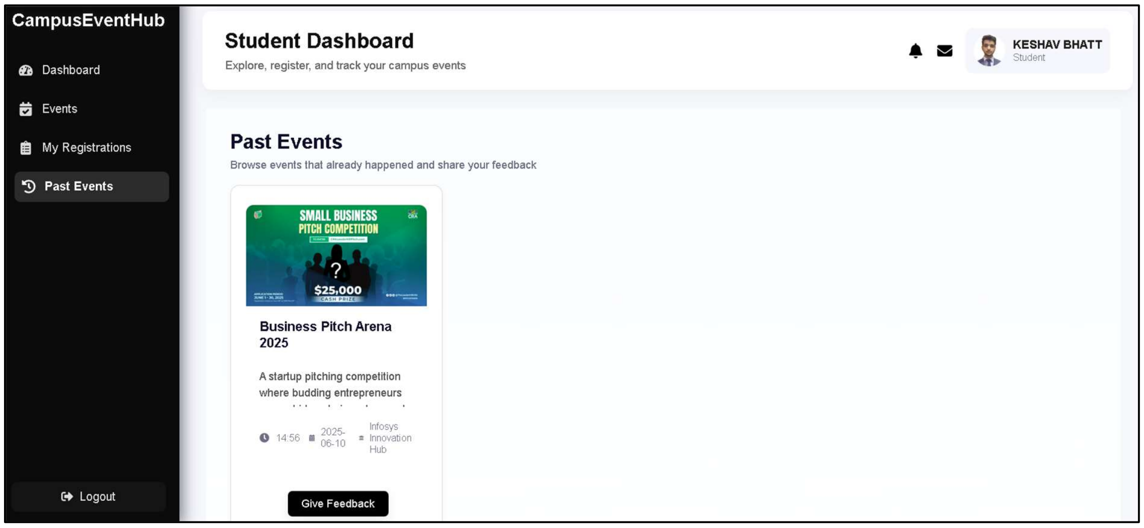
Google Developer Student Club

2025-12-11

Registered: Dec 7, 2025, 02:15 PM

Past Events

- Shows events whose date is before today.
- Each card allows:
    - View Details
    - Open **FeedbackModal** to submit feedback for that event.


CampusEventHub

Dashboard

Events

My Registrations

Past Events

Logout

Student Dashboard

Explore, register, and track your campus events

KESHAV BHATT

Student

Past Events

Browse events that already happened and share your feedback

Business Pitch Arena 2025

A startup pitching competition where budding entrepreneurs

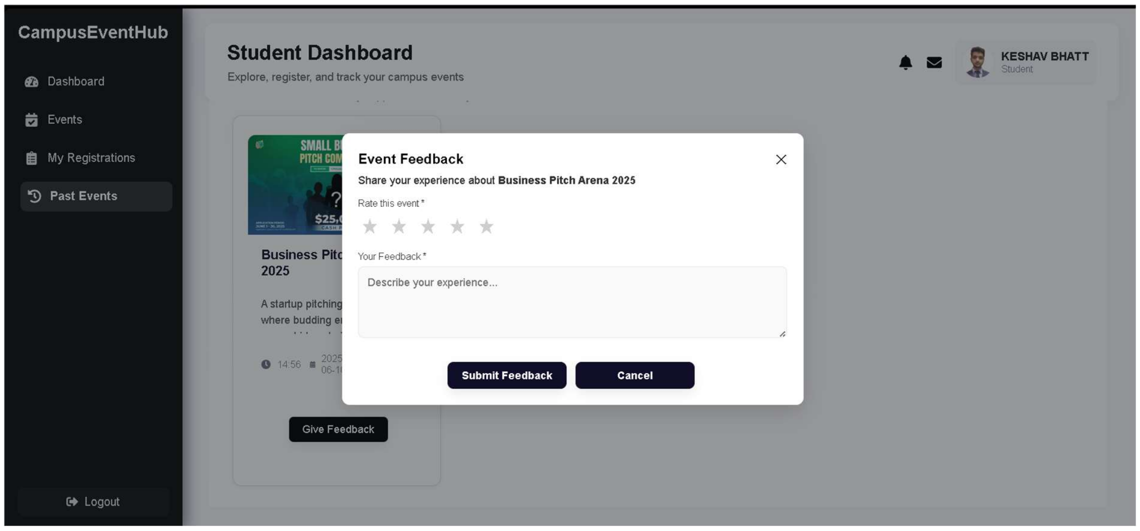
14:56 2025-06-10 Infosys Innovation Hub

Give Feedback

Feedback Components

FeedbackModal

- Reusable modal for submitting feedback:
  - Star rating (1–5).
  - Text comment.
- Uses Axios helper to call:
  - POST /api/feedback/:eventId
- After submit:
  - Shows toast notification.
  - Notifies parent via onSubmitted callback.


CampusEventHub

Dashboard

Events

My Registrations

Past Events

Logout

Student Dashboard

Explore, register, and track your campus events

KESHAV BHATT

Student

Event Feedback

Share your experience about **Business Pitch Arena 2025**

Rate this event *

Your Feedback *

Describe your experience...

Submit Feedback

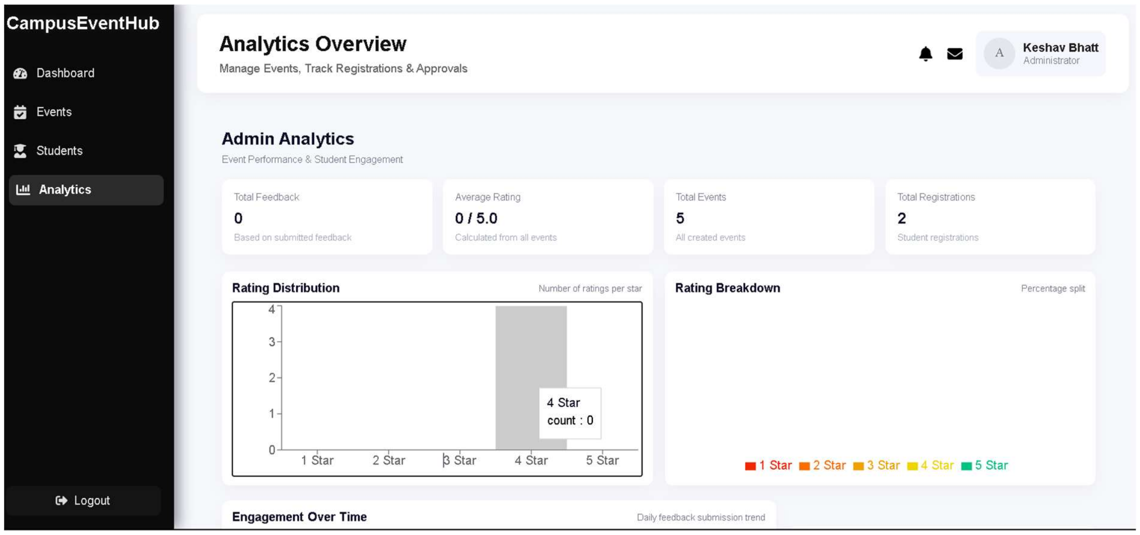
Cancel

Give Feedback

EventDetails Feedback Section

- Displays:

- Average rating (averageRating) and total reviews (totalFeedbacks) retrieved from backend.
- List of feedback entries showing:
  - User name
  - Rating stars
  - Comment
  - Date



Analytics Overview

Manage Events, Track Registrations & Approvals

Admin Analytics

Event Performance & Student Engagement

Total Feedback	Average Rating	Total Events	Total Registrations
0	0 / 5.0	5	2
Based on submitted feedback	Calculated from all events	All created events	Student registrations

API Components

This section documents all backend REST APIs implemented in **College Event Hub**. The backend is built using **Node.js (Express)**, **MongoDB (Mongoose)** and uses **JWT Authentication**.

Base URL for backend (development):

http://localhost:5000/api

User Authentication & Management APIs :

<u>User Registration</u>

Route:

POST /api/auth/signup

Description:

Registers a new user (Student / Admin / Organizer).

Input Example:

```
{
  "name": "John Doe",
  "email": "john.doe@example.com",
  "password": "SecurePass123",
  "role": "STUDENT"
}
```
Success Response – 200 OK
```
{
  "message": "Signup successful",
  "user": {
    "name": "John Doe",
    "email": "john.doe@example.com",
    "role": "STUDENT"
  }
}
```
Error – 400 Bad Request
```
{
  "error": "User already exists"
}
```

User Login

Route:

POST /api/auth/login

Description:

Authenticates the user and returns a JWT token.

Input:
```
{
  "email": "john@college.edu",
  "password": "pass123"
}
```
Success – 200 OK
```
{
  "message": "Login successful",
  "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
  "user": {
    "id": "664c12ab",
    "name": "John Doe",
    "role": "STUDENT"
  }
}
```
Error – 401 Unauthorized
```
{
  "error": "Invalid credentials"
}
```

Request Password Reset (OTP)

Route:

POST /api/auth/send-otp

Description:

Sends an OTP to the registered email.

Input:
```
{
  "email": "john@college.edu"
}
```
Success – 200 OK

```
{
  "message": "OTP sent to your email"
}
```

Verify OTP

Route:

POST /api/auth/verify-otp

Description:

Validates the OTP entered by the user.

Input:

```
{
  "email": "john@college.edu",
  "otp": "123456"
}
```

Success – 200 OK

```
{
  "message": "OTP verified successfully"
}
```

Reset Password

Route:

POST /api/auth/reset-password

Input:

```
{
  "email": "john@college.edu",
  "newPassword": "NewPass123"
}
```

Success – 200 OK

```
{
  "message": "Password reset successful"
}
```

Get Current User Profile

Route:

GET /api/auth/me

Headers:

Authorization: Bearer <token>

Response:

```
{
  "user": {
    "id": "64fc1b23",
    "name": "John Doe",
    "email": "john@college.edu",
    "role": "STUDENT",
    "isVerified": true
  }
}
```

Events Management APIs :

Get All Events

Route:
GET /api/events
Description:
Fetch all approved events.
Example Response:
```
{
 "events": [
  {
   "_id": "evt1",
   "name": "AI Workshop",
   "category": "Technical",
   "college": "ABC College",
   "date": "2025-01-20",
   "time": "10:00 AM"
  }
 ]
}
```

Get Event by ID
Route:
GET /api/events/:id
Response:
```
{
 "event": {
  "_id": "evt1",
  "name": "AI Workshop",
  "description": "Hands-on session",
  "date": "2025-01-20",
  "time": "10:00 AM",
  "location": "Auditorium"
 },
 "registrations": 52,
 "rating": {
  "averageRating": 4.5,
  "totalFeedbacks": 17
 }
}
```

Create Event (Admin / Organizer)
Route:
POST /api/events
Input:
```
{
 "name": "Hackathon",
 "description": "24-hour coding contest",
 "date": "2025-05-01",
 "time": "09:00",
 "category": "Technical",
 "college": "XYZ Engineering College",
 "location": "Seminar Hall",
 "prizes": "10K Cash",
 "entryFee": 100
}
```
Response:

```
{
  "message": "Event created successfully",
  "eventId": "evt12345"
}
```

Update Event

Route:

PUT /api/events/:id

Response:

```
{
  "message": "Event updated successfully"
}
```

Delete Event

Route:

DELETE /api/events/:id

Response:

```
{
  "message": "Event deleted successfully"
}
```

Registration APIs :

Register for an Event

Route:

POST /api/events/:id/register

Input:

```
{
  "registrationDetails": {
    "fullName": "John",
    "email": "john@example.com",
    "preferredTimeSlot": "Morning"
  }
}
```

Response:

```
{
  "message": "Registration submitted",
  "status": "pending"
}
```

Get Event Registrations (Admin)

Route:

GET /api/registrations/:eventId

Response Example:

```
{
  "registrations": [
    {
      "user": {
        "name": "Student User",
        "email": "student@college.edu"
      },
      "status": "approved"
    }
  ]
}
```

Feedback APIs :

Submit Feedback

Route:

POST /api/feedback/:eventId

Only approved participants can submit feedback.

Input:

```
{
  "rating": 5,
  "comment": "Amazing event!"
}
```

Response:

```
{
  "message": "Feedback submitted",
  "stats": {
    "averageRating": 4.7,
    "totalFeedbacks": 20
  }
}
```

Get All Feedback for an Event

Route:

GET /api/feedback/:eventId

Response:

```
{
  "feedbacks": [
    {
      "user": "John Doe",
      "rating": 5,
      "comment": "Well organized!",
      "createdAt": "2025-02-02"
    }
  ],
  "stats": {
    "averageRating": 4.6,
    "totalFeedbacks": 15
  }
}
```

User APIs :

Get User Profile

Route:

GET /api/auth/me

Response:

```
{
  "user": {
    "id": "user-1",
    "name": "John Doe",
    "email": "john@college.edu",
    "role": "STUDENT"
  }
}
```

Get User Registrations

Route:

GET /api/user/:id/registrations

Response:

```
{
  "registrations": [
    {
      "event": "Tech Symposium 2024",
      "status": "approved"
    }
  ]
}
```

Error Handling and Troubleshooting

Effective error handling and troubleshooting are essential for ensuring the smooth and reliable operation of the College Event Hub platform.
This section outlines the most common issues users or developers may face, along with clear steps to diagnose and resolve them.

Gmail SMTP Not Working

Error Description:
SMTP is used to send verification emails, OTP codes, and password reset messages.
If SMTP fails, users will not receive essential emails.

Troubleshooting Steps:
• **Enable App Passwords:** If using Gmail with Two-Factor Authentication (2FA), generate a Gmail App Password.
• **Verify SMTP Configuration:**
– Server: smtp.gmail.com (or the configured SMTP host)
– Port: 587 (TLS)
– Secure: true
• **Check Login Credentials:** Ensure EMAIL_USER and EMAIL_PASS in .env are correct.
• **Review Backend Logs:** Look for Nodemailer errors such as "Invalid login", "Connection timeout", etc.

Port Conflicts

Error Description:
The Node.js server may fail to start if the configured port (e.g., 3000/5000/1000) is already in use by another process.

Troubleshooting Steps:
• **Identify the Process Using the Port:**
– Linux/macOS: `lsof -i :3000`
– Windows: `netstat -ano | findstr :3000`
• **Terminate the Conflicting Process:** Kill the process using the displayed PID.
• **Or Change the Port:** Update the port value in the `.env` file (e.g., use 8080, 5001).

Database Connection Failures

Error Description:
The backend may fail to connect to MongoDB due to invalid credentials, service downtime, or network restrictions.

Troubleshooting Steps:
• **Verify Connection String:** Confirm that `MONGO_URI` in the `.env` file is correct.
• **Ensure MongoDB is Running:**
– Local MongoDB must be running
– MongoDB Atlas cluster must be online
• **Check User Permissions:** Ensure the database user has proper read/write access.
• **Whitelist IP (Atlas Only):** Add your system IP to the MongoDB Atlas Network Access list.

API Request Failures

Error Description:
API calls may fail due to incorrect routes, missing authentication, or server errors.

Troubleshooting Steps:
• **Verify Endpoint URLs:** Ensure the frontend hits the correct API routes
(e.g., `/api/auth/login`, `/api/events/:id/register`).
• **Check Authentication:** Include the JWT token in the request header:
`Authorization: Bearer <token>`
• **Interpret HTTP Error Codes:**
– **401 Unauthorized** → Invalid or expired token
– **404 Not Found** → Wrong route or missing resource
– **500 Server Error** → Inspect backend logs

File Upload Failures

Error Description:
Users may be unable to upload event posters or profile images.

Troubleshooting Steps:
- **File Size Limit:** Check upload size defined in middleware or Express settings.
- **Allowed File Types:** Ensure only valid formats (.jpg, .png, .jpeg, .pdf) are accepted.
- **Storage Availability:** Verify sufficient local or cloud storage space.

Authentication Problems

Error Description:
Users may be unexpectedly logged out or unable to log in.

Troubleshooting Steps:
- **Token Expiration:** JWT tokens automatically expire — prompt the user to log in again.
- **Correct Credentials:** Ensure valid email/password combination.
- **Check Local Storage or Cookies:** Tokens must be saved and sent with every request.

General Troubleshooting Tips

- **Clear Browser Cache:** Try refreshing the page or using incognito mode.
- **Restart Servers:** Restart both backend and frontend after code changes.
- **Use Postman / Insomnia:** Test APIs independently to confirm correct responses.
- **Check Logs:** Review both frontend console logs and backend server logs for detailed error messages.

Glossary or FAQ

Glossary :

- **JWT (JSON Web Token):** Token used for login authentication and secure API access.
- **Registration:** Student signing up for an event; stored with status (pending/approved/rejected).
- **Feedback:** Rating (1–5 stars) and comment submitted by a student for an event.
- **Event Status:** Indicates whether an event is upcoming, completed, or ongoing.
- **Registration Status:** Shows approval state for a student’s event registration.
- **OTP:** One-time code used during password reset verification.
- **Admin Dashboard:** Area for admins to manage events, registrations, feedback, and analytics.
- **Student Dashboard:** Area for students to browse events, register, and give feedback.

FAQs:

Why am I not receiving email notifications?

- Check spam/junk folders.
- Verify your email in profile settings.
- Ensure SMTP/App Password is correctly configured by the admin.

How do I reset my password?

• Click **Forgot Password** → Enter email → Verify OTP → Set new password.

Why can't I register for an event?

• Registrations may be closed.
• Seats may be full (if limited).
• Your registration may already be submitted or rejected.

How do I submit feedback?

• Go to **Past Events** → Select event → Click Write Feedback.

Why was my registration rejected?

• Event seats were filled.
• Eligibility criteria not met.
• Admin disabled or rejected the request.

How do I update my profile?

• Go to **Profile → Edit Profile → Save Changes**.

Why is my registration still pending?

• Admin has not approved the request yet.

The server isn't starting — what should I do?

• Check if another process is using the same port.
• Update the PORT in .env or stop the conflicting process.

Future Enhancements

To further expand the capabilities of **College Event Hub**, several advanced features can be integrated in future versions. These additions will improve usability, automation, and overall event management efficiency.

1. Advanced Analytics Dashboard

- Interactive charts for tracking event performance
- Insights on registration trends, student participation, and feedback patterns
- Predictive analytics for estimating turnout and identifying popular categories

2. Real-Time Notifications & Alerts

- Push notifications for registration updates, event reminders, and approvals
- In-app alerts for schedule changes, announcements, or new events
- Optional email and SMS notification support

3. QR-Based Attendance System

- Auto-generated QR codes for each event
- Scan-based check-in to track attendance instantly
- Reduce manual entry and eliminate fraudulent attendance

4. Integrated Payment System

- Online payments for paid events using Razorpay / Stripe
- Auto-generated receipts and payment history
- Support for discounts, coupons, and early-bird pricing

5. Dedicated Mobile Application

- Android and iOS app for quick access
- Offline caching for viewing events without internet
- Push notification support for instant updates

6. Enhanced Organizer Role

- Separate dashboard for event organizers with controlled permissions
- Ability to manage specific events without full admin access
- Tools for verifying attendees, viewing registrations, and collecting feedback