**Charter Document**

**Event Announcer Application – MITAOE**

# Purpose

* To create a centralized platform for all MITAOE student clubs and departments to announce and manage their events.
* Allows students to browse, register, and get notified about upcoming events in one place.
* Supports event creation, updates, and filtering by category, club, or date.
* Includes a dedicated module for Nakshatra, the college’s major cultural and sports festival.
* Enhances student engagement, simplifies event management, and promotes a digital campus experience.

# Objective

To streamline the event management and registration process for MITAOE by:

* Providing a unified portal for all clubs to post upcoming events
* Allowing students to discover and register for club events or large-scale college events (like Nakshatra)
* Supporting category-wise filtering (e.g., cultural, sports, tech)
* Sending notifications and reminders
* Tracking user registrations and participation history
* Offering a user-friendly interface

# Demand / Opportunity

**Institutional Demand**

Currently, event announcements in MITAOE are decentralized (e.g., posters, social media, WhatsApp groups). There is a strong need for a unified platform where:

* Clubs can manage and promote their events
* Students can easily view and register for events
* College admins can track engagement and participation

This project supports digital transformation and streamlines event operations campus-wide.

# Business Requirement:

* **User Authentication:** Secure login/registration using email/password.
* **Event Management (CRUD):** Organizers can create, read, update, and delete events.
* **Categorization & Tagging:** Events are organized under categories and searchable tags.
* **Save/Favourite Events:** Users can bookmark events to revisit or get reminders.
* **Event Registration:** Users can register for an event and receive confirmation.
* **Real-Time Notifications:** Users get notified about newly added or updated events.
* **Search & Filters:** Efficient search and filter options by location, date, category, etc.
* **Responsive Interface:** Fully functional and optimized UI for all device types.

## **Technical Requirement**

### ● User Authentication

○ UI for secure email/password login and registration  
○ API routes for student and club admin sign-up, login, logout, and session management  
○ Passwords securely hashed and stored in the backend using encryption techniques .  
○ Session-based authentication system for protected routes and role management (Student, Club Admin, Super Admin)

### ● Event Management (CRUD)

○ API endpoints to create, read, update, and delete events by authorized club admins  
○ Event fields include title, description, date, time, venue, category, poster, and registration link  
○ UI for club admins to manage their event listings through a dashboard  
○ Validation for event details before submission

### ●Event Registration System

○ UI for students to register for any event with a single click  
○ Backend logic to track registered users for each event  
○ Email confirmation using Node mailer sent to students after successful registration  
○ Dashboard to view upcoming and past registered events

### ● Favourites and Reminders

○ Students can bookmark/save favourite events for easy access  
○ API endpoints to add, retrieve, and remove favourite events  
○ Notifications or reminders sent for saved events before the event date

### ● Nakshatra Management Module

○ Dedicated module to showcase cultural and sports events under Nakshatra  
○ Allow registration to multiple competitions from one place  
○ Filtering by category (e.g., Singing, Football, Dance, Cricket, etc.)  
○ Real-time updates and announcements for changes or result

### ● Search and Filtering

○ Backend logic for searching events by title, category, club, or date  
○ Frontend filters to allow users to refine events by category, club, or registration status  
○ Sorting options based on upcoming date or popularity

### ● Real-Time Notifications ○ Notifications display on UI dynamically without needing a page refresh

### ● Data Storage

○ Secure and scalable **MongoDB** database for storing user profiles, event details, registration lists, and favourites  
○ Collection design to handle different user roles, event categories, and Nakshatra data

## **Technological Requirement**

### ● Frontend

○ **React.js** – To build an interactive, component-based UI  
○ **Tailwind CSS** – For modern, responsive, and easily customizable design

### ● Backend

○ **Node.js + Express.js** – RESTful APIs for managing user authentication, events, and registrations  
○ **Node mailer** – To send event registration confirmation emails and club communication

### **OAuth 2.0** - Let users log in to your app without creating a new account using login with google.

### ● Database

○ **MongoDB** – NoSQL database to store user info, event details, registration history, and club-specific data

### ● Cloud Storage

○ **AWS S3** – To store event posters, banners, and any media files securely and efficiently

### ● Hosting

○ **Render** or **Railway** – For deploying and hosting the full-stack application

### ● Dev & Testing Tools

○ **Visual Studio Code** – Code editor for project development  
○ **GIT & GitHub** – Version control and team collaboration  
○ **Postman** – For testing REST APIs and validating request/response cycles  
○ **MongoDB Atlas** – GUI for database visualization, inspection, and manual queries

# Stakeholder:

|  |  |  |
| --- | --- | --- |
| **Role** | **Name(s)** | **Count** |
| Developers | Aditi, Shravani, Sayali | 3 |
| DB Designer | Sayali | 1 |
| Testers | Pooja, Jayesh Raut, Chetan | 3 |
| Project Management | Aditi | 1 |
| Documentation | Shravani (Creator), Sayali (Reviewer) | 2 |
| Cloud Service | AWS | 1 |
| Hosting Provider | Render | 1 |
| Security Reviewer | Sayali | 1 |
| Clubs & Event Admins | All MITAOE clubs | - |
| Students (Users) | All MITAOE students | - |

**Resources Needed:**

* **Documentation & Development Tools**

React.js – For building the frontend user interface  
 Tailwind CSS – For styling and responsive UI design  
 Express.js – For creating backend REST APIs  
 MongoDB – For storing user, event, and registration data  
 Nodemailer – For sending event registration confirmations and email alerts  
 Postman – For API testing and validation  
 Git & GitHub – For version control and team collaboration  
 Visual Studio Code – Code editor for frontend and backend development

* **Cloud Account**

AWS – For storing event images/posters using S3 bucket

* **Hosting**

Render (or Railway) – For deploying and hosting the full-stack application

* **Human Resource**

|  |  |  |
| --- | --- | --- |
| **Role** | **Count** | **Name(s)** |
| UI/UX Designer | 1 | Sayali Deshmukh |
| Frontend Developer | 2 | UI Development – Shravani, API Integration – Sayali |
| Backend Developer | 1 | Sayali |
| DB Designer | 1 | Shravani |
| Project Management | 1 | Aditi |
| Documentation | 2 | Creator – Shravani , Reviewer – Sayali Deshmukh |
| Tester | 2 | Pooja, Jayesh Raut, Chetan |

# PESTEL Analysis:

* **Political:**
  + No direct government constraints for a college-level internal platform.
  + Requires alignment with MITAOE’s institutional IT and data usage policies.
* **Economic:**
  + Utilizes a low-cost, open-source tech stack (MERN) suitable for student projects.
  + Affordable hosting and cloud services (Render, AWS S3) keep operational costs minimal.
* **Social:**
  + Encourages student engagement by centralizing event discovery and registration.
  + Enhances the visibility and reach of club events and college fests like Nakshatra.
* **Technological:**
  + Built with scalable technologies (React, Node.js, MongoDB, Socket.io).
  + Mobile-responsive UI ensures accessibility across devices.
* **Environmental:**
  + Cloud hosting reduces the need for on-premise infrastructure and energy use.
  + Promotes a paperless culture by digitizing event promotion and registration.
* **Legal:**
  + Must ensure secure storage of student data and protect user credentials.
  + User consent is needed for notifications, and basic data privacy practices must be followed.

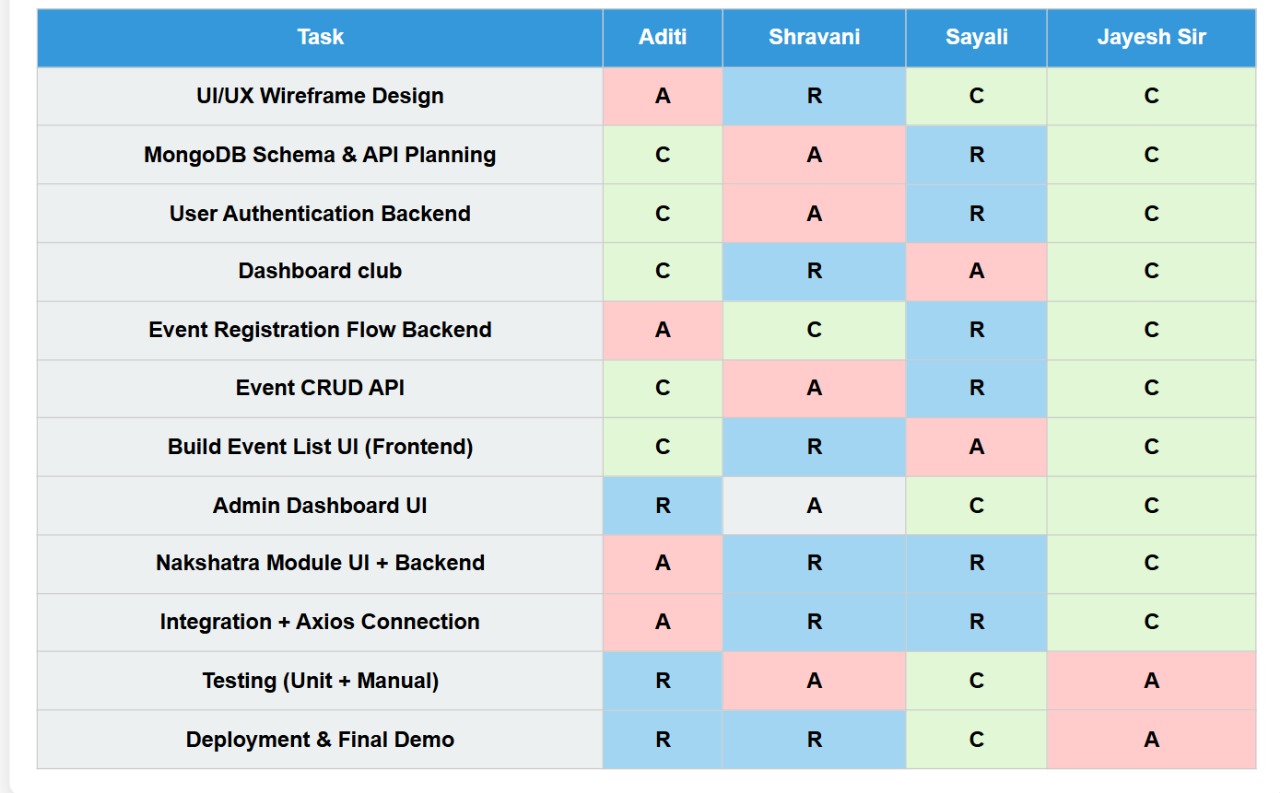
**Risk Analysis:**

|  |  |  |
| --- | --- | --- |
| **Risk** | **Description** | **Mitigation** |
| **Low Adoption** | • Students may not know about the platform.  • They may continue using posters/WhatsApp. | • Promote via college campaigns.  • Introduce it during club orientations. |
| **Data Misuse** | • Organizers may access data from other clubs.  • Risk of unintentional data exposure. | • Implement role-based access.  • Restrict access per user role. |
| **Feature Overload** | • Too many features can confuse users.  • MVP may get delayed or buggy. | • Focus on essential features first.  • Plan extras for later versions. |
| **Server Overload** | • App may lag or crash during large events.  • Increased simultaneous usage can cause downtime. | • Use pagination & caching.  • Optimize backend queries. |
| **Security Vulnerabilities** | • Risk of unauthorized admin access.  • Sensitive data could be compromised. | • Use session and input validation. • Maintain secure logs and headers. |

**Timeline / Milestone:**

|  |  |  |
| --- | --- | --- |
| **Phase** | **Tasks (Detailed)** | **Timeline** |
| **Requirement & Planning** | • Identify all stakeholders (students, club admins, super admins).  • Gather feature requirements (event posting, registration, filtering, Nakshatra module, etc.).  • Create user flow diagrams and design wireframes for key screens (event list, event creation, registration, dashboard). | **Week 1** |
| **Backend Development** | • Set up Express.js project structure.  • Develop RESTful APIs for event CRUD, user registration/login, and student-event registration.  • Implement session-based authentication and role-based access (student, club admin).  • Integrate Node mailer for confirmation emails. | **Week 2** |
| **DB & API Design** | • Design database schema for users, events, registrations, and favourites.  • Establish entity relationships (e.g., user ↔ event, club ↔ event).  • Define API endpoints for user auth, event management, and registration workflows. | **Week 3–4** |
| **Frontend Development** | • Build UI components using React.js and Tailwind CSS (event list, registration form, admin dashboard, event creation/editing form).  • Add responsive design to support mobile and desktop users.  • Add filtering, search, and category selection features. | **Week 5–6** |
| **Integration & Testing** | • Connect frontend with backend APIs using Axios/fetch.  • Test complete workflows: event creation, registration, favourites, Nakshatra module.  • Perform unit testing and manual testing across different roles and scenarios.  • Fix bugs and improve UX. | **Week 7** |
| **Final Deployment** | • Deploy the full-stack app on Render.  • Configure environment variables (API keys, DB connection).  • Set up AWS S3 for image uploads (event posters).  • Conduct live testing and final review before presentation/submission. | **Week 8** |

**RACI Chart:**



* **R = Responsible (executes the task)**
* **A = Accountable (owns the outcome)**
* **C = Consulted (input provider)**