Implementation Document

for

EventriX

Version 1.0

Prepared by

Group 3:		**Group Name: Tech Troopers**
Parmar Priyen Indravadan	**230735**	**parmari23@iitk.ac.in**
Divyansh Bansal	**230381**	**divyanshb23@iitk.ac.in**
Jayant Mitawa	**230504**	**jayantm23@iitk.ac.in**
Shaik Jameel Ur Rahaman	**230951**	**shaikja23@iitk.ac.in**
Pranav Krishna	**230771**	**pranavkris23@iitk.ac.in**
Misar Shlok Sunil	**230653**	**smshlok23@iitk.ac.in**
Mohit Parihar	**230660**	**mohitp23@iitk.ac.in**
Majji Sharmila	**230620**	**msharmila23@iitk.ac.in**
Hemanth Kumar Ampili	**230128**	**hemanthka23@iitk.ac.in**
Chaitanya Vishnoi	**230316**	**chaitanyav23@iitk.ac.in**

Course: CS253

Mentor TA: Hemant Mohanlal Khatri

Date: 28th March 2025

Index

CONTENTS II

REVISIONS III

1 IMPLEMENTATION DETAILS 4

2 CODEBASE 8

3 COMPLETENESS 10

APPENDIX A - GROUP LOG 11

Revisions

Version	Primary Author(s)	Description of Version	Date Completed
1.0.0	All Group Members	First Draft	28/03/2025

1 Implementation Details

Introduction

EventriX is a web-based platform that simplifies event organization and participation within our institution. It allows students to browse and register for events, while clubs can create, update, and manage events effortlessly.

To make the platform interactive, visually appealing, and highly efficient, we used React.js for the frontend. React.js enables the creation of dynamic, fast, and smooth web applications. Additionally, we used Tailwind CSS to ensure a clean and responsive design, making the platform accessible on various devices.

Frontend Development with React.js

Why We Chose React.js?

React.js is a JavaScript library that helps in building modern, efficient, and reusable user interfaces. Unlike traditional websites that reload every time a user interacts with them, React enables us to update only specific parts of the page without refreshing everything. This makes the application faster and more responsive.

We chose React.js because:

- **It creates a single-page application (SPA)** : Users can navigate different sections without reloading the entire page.
- **It improves performance** : React updates only the changed data, making the UI smooth and efficient.
- **It supports fast development** : Development becomes quicker due to React's modular approach.
- **Structuring the Frontend with Components** : In React.js, we divided the website into small, independent components. Each component handles a specific part of the user interface and can be reused throughout the application. Each of these components interacts with the backend using **API calls** and updates dynamically whenever a user performs an action.

Making the UI Beautiful and Responsive with Tailwind CSS

To design the frontend, we used Tailwind CSS, a utility-first framework that provides pre-built styling classes. Instead of writing long CSS files, we could use ready-made classes to style components directly within the HTML structure, which significantly reduced development time.

How Tailwind CSS Helped in Our Project

- **Flexible Layouts with Grid and Flexbox** : We used Tailwind's built-in grid system and flexbox utilities to design an organized, adaptive layout.
- **Responsive Design for All Devices** : The UI automatically adjusts for mobile, tablet, and desktop screens without needing extra code.
- **Pre-Designed Utility Classes** : Tailwind provides ready-to-use classes for margins, padding, colors, and typography, making the UI visually appealing.
- **Custom Theming & Styling** : We could easily define consistent colors, fonts, and spacing, giving the platform a unique, professional look.

Navigation & Page Routing with React Router

Since the EventriX has multiple pages (Home, Events, Clubs, Dashboard), we needed a smooth way for users to move between these sections without reloading the entire website.

For this, we used React Router, a powerful library that helps in managing page navigation within React applications. Instead of traditional links that cause a full-page refresh, React Router changes only the necessary parts of the page, ensuring a seamless experience.

React Router ensures that when a user moves between these pages, only the required section loads dynamically, keeping the interface fast and smooth.

Performance Optimization in the Frontend

To ensure a fast and efficient user experience, we optimized the frontend using several techniques:

- **Lazy Loading** – Instead of loading everything at once, we only load content when needed, reducing page load time.
- **Code Splitting** – We divided large JavaScript files into smaller chunks so that the website loads faster.
- **State Management with useState & Context API** – Instead of making multiple requests to the backend, we store some user data locally, making the app faster.

Conclusion

The frontend of the EventriX is designed to provide a seamless, interactive, and user-friendly experience. By leveraging React.js, Tailwind CSS, and React Router, we created a modern, dynamic, and responsive platform that enhances event organization and participation.

Backend Development with Node.js and Express.js

Why We Chose Node.js and Express.js

Node.js is a JavaScript runtime environment that allows us to build a fast and scalable backend using a non-blocking, event-driven architecture. To simplify backend development, we used Express.js, a lightweight framework for handling HTTP requests, routing, and middleware integration. Express.js makes it easy to create a modular and structured backend, ensuring clean code and efficient API handling.

Structuring the Backend with Routes and Controllers

To maintain a well-organized backend, we followed a Model-View-Controller (MVC) architecture. This separation of concerns made the codebase easy to manage, scale, and debug.

Key Backend Components in EventriX

- **Routes:** Define API endpoints for handling requests from the frontend.
- **Controllers:** Process incoming requests, interact with the database, and send responses.
- **Models:** Define database schemas and structures for different entities.
- **Middleware:** Handles authentication, validation, and request logging.

Each API request is first processed by **routes**, then passed to **controllers**, which interact with **models** to retrieve or store data in the database.

User Authentication and Role-Based Access Control

To secure EventriX, we implemented **JWT-based authentication**. Every user (student or club) must log in to access certain features, and their role determines their permissions.

How Authentication Works in EventriX

- A user registers or logs in using their email and password.
- The backend verifies the credentials and generates a **JWT token**.

Password Hashing for users and admins

- Securely store user passwords by hashing before saving.
- **Algorithms**: **Bcrypt** (Recommended, includes salting).
- **Process :**Hash password before storing in MongoDB.

Hash and compare during login for authentication.

We also implemented **role-based access control (RBAC)** to ensure:

- **Students** can only view and register for events.
- **Admins** can create, update, and delete events.

Database Management with MongoDB and Cloudinary

Why We Chose MongoDB

MongoDB is a NoSQL database that allows us to store data in a flexible JSON-like format (documents). Since events and user details can have dynamic attributes, a schema-less database like MongoDB provides better adaptability compared to relational databases. Additionally, MongoDB works seamlessly with Mongoose, an ODM (Object Data Modeling) library that simplifies database interactions using JavaScript.

Why do we Choose Cloudinary?

Cloudinary is used in EventriX for efficient image storage, management, and retrieval. It enables seamless uploading, optimization, and delivery of event-related images, ensuring fast loading times and high-quality media display across the platform.

Designing the Database Schema

We structured the database to store different types of data efficiently. Each entity (Users, Events, Clubs, Admins) is stored in separate collections for better organization. Each collection is optimized for fast lookups, ensuring quick event retrieval and seamless user interactions.

Optimizing Backend Performance

To ensure EventriX remains fast and scalable, we implemented several backend optimizations:

- **Error Handling & Logging** – Used middleware to catch and log errors systematically.

Conclusion

This implementation ensures that EventriX can support a growing user base while maintaining fast and reliable event management features. Future enhancements may include real-time updates using WebSockets and automated email notifications for event reminders.

2 Codebase

GitHub Repository

🔗 **EventriX GitHub Repository**

About the Web App

The EventriX web application follows a MERN stack architecture, where the front-end and back-end operate as separate units.

Front-end

The front-end is built using **React.js** and is responsible for delivering a seamless and dynamic user experience.

Back-end

The back-end is implemented using **Node.js, Express.js, and MongoDB**, providing RESTful APIs to manage event data, user authentication, and club functionalities.

Codebase Navigation

The repository is organized into distinct sections, ensuring modularity and maintainability:

Front-end Code (/frontend/)

Located in the **frontend/** directory, this contains all UI-related components and logic.

- **src/**: The core directory containing all React components, styles, and state management logic.
- **components/**: Reusable UI components such as buttons, forms, modals, all pages components.
- **assets/**: Contains static files, images, and stylesheets.
- **public/**: Holds the base HTML template and meta configurations for the app and images used in the website.

Back-end Code (/backend/)

Located in the **backend/** directory, this handles API requests, authentication, and database operations.

- **routes/**: Defines API endpoints for events, users, and clubs.

- **controllers/**: Implements the core logic for processing API requests.
- **models/**: Mongoose schemas defining database structures for users, events, and clubs.
- **middlewares/**: Includes authentication and validation logic.
- **config/**: Database connection and environment variable configurations.
- **server.js** : We used it to run servers.
- **public/temp/:** We used it for the local storage of files.

Database Configuration (/database/)

- MongoDB is used as the primary database.
- Schema definitions for events, users, and clubs are stored in the models/ directory.
- The database connection is initialized in config/db.js.

Installation Requirements

Before running the project, ensure you have the following installed:

- Node.js (v14+ recommended)
- npm (Node Package Manager)
- MongoDB (Local or cloud-based instance)
- A modern web browser (Google Chrome, Firefox, etc.)

This structured approach ensures seamless navigation of the EventriX codebase while maintaining clarity and scalability.

3 Completeness

SRS Features	Status	Future Development Plans
Profile Login/Registration	Completed	OTP verification has been implemented to enhance security and prevent unauthorized access. In future updates, we may introduce multi-factor authentication (MFA) for more security.
Event Creation & Uploading	Completed	Clubs and admins can create and manage events, including editing event details after creation. Future enhancements could include bulk event creation and AI-based event recommendations.
Events Page (Viewing Events)	Completed	Users can browse and view all available events. In future iterations, personalized event suggestions based on user interests and past participation will be added.
Councils & Cells Pages	Completed	These pages function similarly to the Clubs Page, displaying relevant information and coordinator details. We plan to introduce filtering and sorting features for easier navigation.
Club Admin Dashboard	Completed	Clubs can update details, add and edit events. Future updates may include analytics and engagement insights for club admins to track event success and member activity.
Fests Page	Completed	The Fests Page will display fest details, schedules, and Team Heads' information.
Event Registration (Users)	In Progress	Users can register for events, and clubs can view participant details. In the future, event ticketing and QR-based check-in systems might be introduced.
Paid Events & Workshops	In Progress	Currently, event registration is free, but payment integration using platforms like Stripe or Razorpay will be implemented for paid events and workshops. Secure payment handling and refund policies will also be introduced.
Event Feedback & Reviews	In Progress	A review and rating system will be added to allow users to provide feedback on events. This will help clubs improve their events based on attendee opinions.
Notifications System	In Progress	Users will receive notifications for new events uploaded by subscribed clubs. Additionally, event-specific reminders will be implemented so attendees don't miss important updates.
Merchandise System	In Progress	A system will be introduced to allow clubs to list and sell merchandise related to their events. The purchasing process will be integrated with a secure payment system, and inventory management for clubs will be added.

Appendix A - Group Log

Sl. No.	Date	Timings	Venue	Description	Duration
1.	2 March	10:00 pm	Online Meet	Discussed about the frameworks we are going to use	1 hour
2.	4 March	9:30 pm	RM Building	Divided 10 people in 3 groups of frontend, backend and databasing	1 hour
3.	9 March	11:30 pm	Online Meet	Checked the progress of the project and discussed the problems that we were facing	1 hour
4.	18 March	9:30 pm	RM Building	Worked on finalising the different frontend components and made some design changes	2.5 hour
5.	20 March	10:00 pm	RM building	Discussed about how to proceed with the backend and databasing of components	1.5 hour
6.	22 March	10:00 pm	Online Meet	Checked the progress of the project and started work on backend along with some changes in frontend	2 hour
7.	24 March	10.30 pm	RM building	First draft of frontend was majorly done and did further work on backend and how databases will be integrated	3 hour
8.	25 March	10:00 pm	Online Meet	Discussed about the progress and also about how to integrate the different components and worked on events databasing	3 hour
9.	27th March	9:30 pm	RM building	Completed the routing for various components and pages and started work on the implementation document	3.5 hour
10	28th March	4:30 pm	KD Lab	Final discussion on the project ,made some final changes to the frontend and completed the implementation document	3 hour