Back End Engineering-I
Project ID-(G9-B-PID11)

Project Report

Semester-IV (Batch-2023)

Event Scheduling System

Supervised By:
Dr. Prabhjot Singh Manocha

Submitted By:
KaranbirSingh,2310990708(G-9)
Kartik Jindal, 2310990709(G-9)
Nahar, 2310990744(G-9)
Pranav Goyal, 2310991426(G-9)

Department of Computer Science and Engineering
Chitkara University Institute of Engineering & Technology,
Chitkara University, Punjab

Abstract

Occasio is a web-based application developed to streamline the process of organizing, managing, and attending events. In today's fast-paced digital environment, both event hosts and participants face challenges such as scattered communication, inefficient planning tools, and limited engagement. This project addresses these concerns by providing an integrated platform that allows users to create, update, and delete events while enabling attendees to explore, search, and interact with events easily.

The application is built using a full-stack JavaScript architecture, incorporating Node.js, Express.js, MongoDB, and Handlebars (HBS), ensuring a robust and scalable backend. The frontend is developed with HTML, CSS, and JavaScript, with Bootstrap and Font Awesome enhancing the user interface and experience. The platform supports essential functionalities such as adding event-specific details (title, date, time, location, and description), and allows attendees to filter, like, and comment on events. This interactivity boosts user engagement and provides meaningful feedback to event organizers.

One of the standout features of the application is its responsive design, allowing seamless usability across both desktop and mobile devices. It also includes a modular backend architecture, structured with RESTful APIs, making it scalable and easy to maintain. The use of MongoDB ensures efficient data retrieval and storage, supporting the real-time reflection of changes in the event listings.

In essence, Occasio offers an intuitive and user-friendly interface tailored to the needs of both event hosts and participants. It solves real-world problems related to event coordination and participation by simplifying communication and improving accessibility. With potential future enhancements such as authentication features, calendar integration, and real-time notifications, the app sets a solid foundation for a comprehensive and smart event management system. This project not only demonstrates practical implementation of modern web development tools but also contributes meaningfully to enhancing digital event experiences.

This project showcases the effective integration of modern web technologies to solve a common real-life problem. By focusing on usability, responsiveness, and modular design, Occasio not only meets current user needs but also lays the groundwork for future scalability and feature expansion. It reflects a strong understanding of full-stack development and practical application of software engineering principles.

Table of contents

Table 1.0

1. Introduction

1.1 Background

In today's digitally driven world, events are not just limited to large-scale conferences or concerts; they span across educational seminars, office meetings, startup launches, webinars, social campaigns, and casual community gatherings. With the growing number of events and the increasing dependency on digital tools, the demand for efficient, real-time, and centralized event management systems has never been higher.

As events become more dynamic and user expectations increase, there is a clear need for a platform that provides not just event visibility but also real-time updates, user interaction, and reliable access across devices. Users—both organizers and attendees—expect simple and intuitive platforms that work seamlessly without requiring them to jump between tools or platforms. Furthermore, in many educational institutions and organizations, there is no dedicated system for managing internal events, causing reliance on external or makeshift solutions.

This project addresses that gap by building a web-based Event Scheduling App that brings together the essential features needed by both event hosts and participants. The application enables hosts to create, edit, and delete events easily, while users can browse, search, and engage with those events through features like comments and likes. The app is developed using a full-stack JavaScript approach, with Node.js and Express.js powering the backend, MongoDB for database management, and Handlebars for server-side templating. This stack ensures flexibility, scalability, and efficient performance.

The background of this project also lies in the educational objective of learning full-stack development and applying it to solve real-world problems. By designing and developing this app, the project team not only delivers a functional system but also strengthens their knowledge of software architecture, UI/UX principles, and backend API design. Occasio serves as a strong foundation for both academic exploration and practical deployment in real use-cases.

1.2 Objectives

The primary objective of Occasio is to develop a centralized and user-friendly platform that simplifies the process of event management for both organizers and attendees. In today’s fast-paced environment, managing event logistics such as scheduling, communication, and updates can be challenging without a dedicated system. This app seeks to bridge that gap by providing a seamless and efficient tool that enhances coordination, communication, and user engagement.

One of the main goals is to empower event hosts with the ability to create, edit, and delete event listings with ease. Hosts can add vital event details such as title, description, date, time, and location, thereby ensuring that participants have access to all necessary information. The system also enables hosts to update or cancel events when needed, with changes reflecting in real time for users. This significantly reduces confusion and enhances clarity in event communication.

Another key objective is to improve the experience of attendees by providing them with a clean, responsive dashboard where they can explore upcoming events. With integrated search, filter, and sort features, users can quickly find events relevant to their interests based on parameters such as name, date, or type. The intuitive interface ensures easy navigation and encourages user participation by making the process of discovering and following events enjoyable.

Additionally, the application focuses on fostering user interaction and feedback. Features such as a comment section and a “like” system have been included to facilitate engagement. These features allow users to share opinions, ask questions, and gauge the popularity of events, which can influence their decision to attend.

From a technical perspective, the project aims to implement a modular and scalable architecture using the MERN stack components (MongoDB, Express.js, Node.js) along with Handlebars (HBS) for templating. The use of RESTful APIs ensures efficient communication between the front-end and back-end, while the optimized MongoDB database supports fast data retrieval and robust storage capabilities. The app is also designed to be responsive, supporting both desktop and mobile devices.

1.3 Significance

In the modern digital age, efficient and accessible event management tools play a critical role in facilitating communication, coordination, and engagement among communities, institutions, and organizations. The significance of Occasio lies in its ability to transform the traditional and often fragmented process of event planning into a cohesive, interactive, and streamlined digital experience. This project addresses the growing need for a centralized platform that not only simplifies the organizational workload of event hosts but also enhances the overall user experience for attendees.

Event organizers frequently face the challenge of handling multiple aspects of event coordination, such as managing participant lists, updating schedules, and distributing key information. Without a reliable platform, these tasks can lead to confusion, missed deadlines, and poor attendance. Occasio is significant because it provides hosts with a comprehensive toolkit to manage their events efficiently. By enabling features such as real-time updates, easy editing of event details, and structured data input, the app reduces the cognitive load on hosts and ensures a smooth planning process.

From the perspective of attendees, accessing timely and accurate information about events is crucial. Traditional methods like email chains, posters, or word-of-mouth announcements often fail to keep participants updated on changes in schedule, venue, or other details. This app addresses that gap by offering a user-friendly dashboard where users can browse, filter, and follow events of interest. The addition of interactive components such as a comments section and likes allows users to engage with the content, share opinions, and make informed decisions about which events to attend.

Moreover, the app’s significance extends to its technical contributions. By utilizing a full-stack JavaScript architecture—including Node.js, Express.js, MongoDB, and Handlebars—the project offers a modular and scalable backend system that is both efficient and adaptable. This modern technology stack ensures that the platform remains responsive, secure, and capable of handling growing user demands. Features like RESTful APIs, optimized database queries, and clean UI design reflect best practices in contemporary web development and make the system suitable for future enhancements.

The app also plays a pivotal role in promoting user engagement and community participation. Social features such as likes and comments not only drive interaction but also help in building a transparent environment where users can assess the popularity and credibility of events.

For example, a highly liked event with positive user feedback in the comments may encourage others to attend, thereby increasing participation and fostering a sense of community. This dynamic layer of interaction transforms the app from a simple scheduling tool into an inclusive platform for social connection and feedback.

Another area where the app proves its significance is its accessibility. Designed to be fully responsive, the application caters to both desktop and mobile users, ensuring a consistent and intuitive experience across devices. This inclusivity enhances usability for a wider audience, including students, professionals, and community members who may access the app from different platforms and environments.

In addition, the app's extensible design allows for future integration with services such as user authentication, personalized dashboards, and calendar syncing (e.g., Google Calendar). These potential enhancements demonstrate that the app is not just a standalone tool but a foundational system capable of evolving into a more comprehensive event management ecosystem.

In conclusion, Occasio holds substantial significance both in its practical utility and technical robustness. It solves real-world problems related to event management and user engagement while embodying principles of good software engineering. Whether used in educational institutions, corporate environments, or public community groups, the app provides a reliable, interactive, and modern solution to the challenges of event organization and participation.

2. Problem Definition and Requirements

2.1 Problem Statement

Event organization, especially in academic, corporate, or community settings, often involves multiple steps—ranging from scheduling to updating attendees on last-minute changes. In the absence of a centralized digital platform, hosts and attendees face several challenges that lead to inefficiencies and communication breakdowns.

Hosts struggle with maintaining consistency in event information, manually informing participants of updates, and handling multiple event records simultaneously. On the other hand, attendees often miss out on important updates due to a lack of real-time notifications or accessible information. This gap results in low engagement, scheduling conflicts, or event cancellations due to poor coordination.

Moreover, current tools used for event communication such as email threads, messaging apps, or social media are not purpose-built for structured event management. These platforms lack features like filtering, commenting, or performance analytics, which are essential for ensuring event success and high user satisfaction.

To address these issues, Occasio was conceptualized and developed. The app provides a comprehensive platform where hosts can create, update, and delete events, while users can explore, interact, and stay informed about events in real time. It bridges the communication gap, enhances engagement, and streamlines the entire event lifecycle with a modern, intuitive, and responsive interface.

2.2 Software Requirements/ Hardware Requirements/ Data sets

Software Requirements

Occasio uses a modern web development stack that ensures scalability, modularity, and responsiveness. The key software components include:

- Frontend:
  - HTML5, CSS3, JavaScript for structure and interactivity
  - Bootstrap and Font Awesome for responsive design and UI enhancements
- Backend:
  - Node.js with Express.js for server-side development
  - Handlebars (HBS) as the templating engine
  - MongoDB for storing event and user data
  - Mongoose for database modelling
  - RESTful APIs for client-server communication

Hardware Requirements

The application is lightweight and requires only basic hardware for development or deployment:

- Minimum:
  - Processor: Intel i3 or equivalent
  - RAM: 4 GB
  - Disk Space: 500 MB
  - Stable internet connection
- Recommended:
  - Processor: Intel i5 or higher
  - RAM: 8 GB
  - Disk Space: 1 GB
  - Display: 13" or higher with 1366×768 resolution

Data Handling

The app does not rely on pre-existing datasets. All data—such as event details, user comments, and likes—is dynamically created and managed by users. Data is stored in MongoDB and accessed using internal APIs.

3. Proposed Design / Methodology

3.1 Overview

The proposed design of Occasio follows a modular, scalable, and maintainable structure using a full-stack JavaScript approach. The goal of the design is to separate concerns between the frontend, backend, and database, allowing for easy feature expansion and independent debugging. The architecture relies on the MVC (Model-View-Controller) pattern, which promotes code clarity and logical separation of functionality.

The app is built with the following components:

- Frontend: HTML5, CSS3, Bootstrap, JavaScript
- Backend: Node.js, Express.js
- Templating: Handlebars (HBS)
- Database: MongoDB with Mongoose
- Routing & APIs: RESTful endpoints
- Optional Enhancements: User authentication, Google Calendar integration

3.2 System Architecture

The app uses a client-server architecture. The system is composed of:

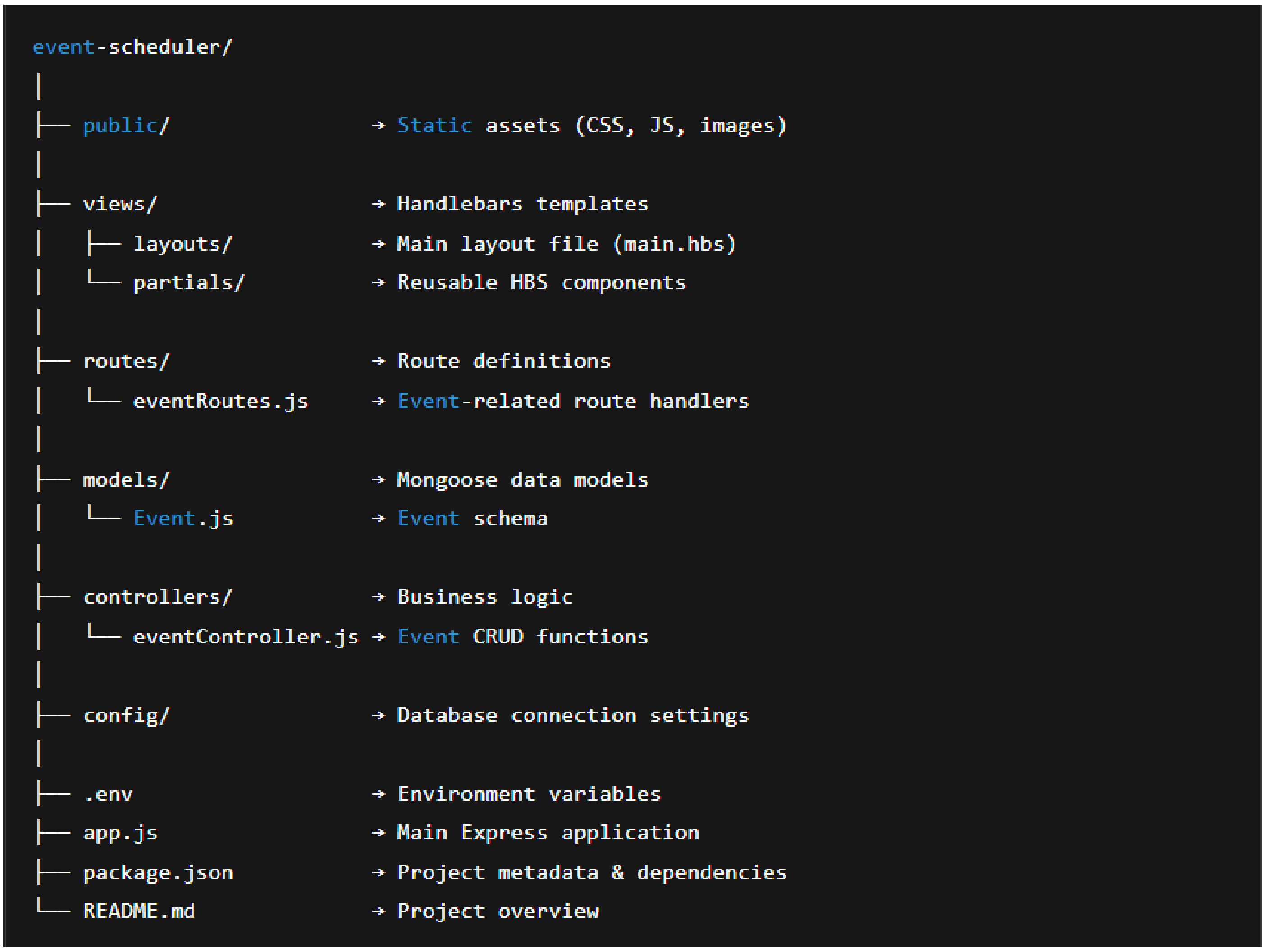
- Client Layer (Frontend): Responsible for rendering views, capturing user inputs, and sending requests to the server.
- Server Layer (Backend): Handles routing, request processing, and business logic using Express.js.
- Database Layer: MongoDB stores all event-related data such as titles, descriptions, dates, likes, and comments.

Data Flow:

1. A user accesses the app via a browser.
2. The frontend requests event data or submits form data via HTTP requests.
3. Express.js routes the request to the appropriate controller.
4. The controller interacts with the MongoDB database using Mongoose models.
5. Data is processed and passed to the frontend via the Handlebars templating engine for rendering.

3.3 File Structure

The project follows a clean directory layout for maintainability and scalability:



```
event-scheduler/
│
├── public/                 → Static assets (CSS, JS, images)
│
├── views/                  → Handlebars templates
│   ├── layouts/            → Main layout file (main.hbs)
│   └── partials/           → Reusable HBS components
│
├── routes/                 → Route definitions
│   └── eventRoutes.js      → Event-related route handlers
│
├── models/                 → Mongoose data models
│   └── Event.js            → Event schema
│
├── controllers/            → Business logic
│   └── eventController.js → Event CRUD functions
│
├── config/                 → Database connection settings
│
├── .env                    → Environment variables
├── app.js                  → Main Express application
├── package.json            → Project metadata & dependencies
└── README.md               → Project overview
```

Figure 1.0

3.4 Methodology

The development process was divided into the following stages:

1. Requirement Analysis

- Identified the key problems and limitations of existing event management systems.
- Defined the functional and non-functional requirements.

2. Design

- Applied MVC architecture to separate data, logic, and presentation.
- Used wireframes and flow diagrams to design the user interface and system workflow.

3. Implementation

- Developed a responsive frontend using Bootstrap.
- Built backend logic in Node.js with Express.js, implementing routes and middleware.
- Connected to MongoDB for data storage and retrieval using Mongoose.

4. Testing

- Manually tested CRUD operations and UI responsiveness across browsers.
- Verified real-time updates and mobile compatibility.

5. Deployment

- Prepared the system for future hosting on platforms like Render or Heroku.
- Configured environment variables for secure database access.

3.5 Key Modules

1. Event Management

- Hosts can create, edit, and delete events.
- Each event includes title, description, date, time, and location.

2. Search and Filter

- Users can search for events by name or filter by date or category.
- Results update dynamically based on input.

3. User Interaction

- Users can add comments to events.
- Like functionality tracks event popularity.

4. Dynamic Rendering

- Views are rendered dynamically using Handlebars based on database content.

3.6 Future Enhancements

- User Authentication: Allow users to register/login and manage their events.
- Calendar Integration: Sync events with external calendars like Google Calendar.
- Notifications: Enable email or SMS alerts for upcoming events.
- Admin Panel: Add moderation tools and event analytics.

4. Results

The development and implementation of Occasio led to the successful realization of all intended functionalities. The system was rigorously tested for both usability and performance across different devices and platforms. The application achieved its objective of offering a centralized, interactive, and responsive solution for managing and exploring events, catering to both event organizers and attendees.

4.1 Functional Outcomes

The app fulfils the following core functionalities:

- Event Creation and Management: Hosts can create, update, and delete events seamlessly. The event form supports key fields such as title, description, date, time, and location. Updates made to events are reflected in real time on the user interface, ensuring that attendees always have access to the most current information.

- User Interaction and Engagement: Users are able to interact with events through the comment section, enabling discussion or queries. The like feature provides a way to gauge the popularity of each event, encouraging user engagement.

- Search and Filtering: A working search functionality allows users to find specific events using keywords. Filtering options enable narrowing down events by date or category, thereby improving accessibility and discoverability.

- Responsive UI Design: The user interface was tested across multiple screen sizes and devices. The layout adapts well from desktop to mobile, with clean navigation and responsive buttons. This ensures an inclusive experience for all users regardless of device.

- Data Integrity and Storage: Event data, comments, and likes are securely stored in the MongoDB database. Data validation and schema management via Mongoose ensure consistency and reliability.

- Template-Based Rendering: The use of Handlebars as the templating engine allows for dynamic content rendering based on the data retrieved from the backend. This supports modular development and enhances scalability.

4.2 Backend Functionality and Data Management

The backend of Occasio plays a crucial role in ensuring data consistency, business logic enforcement, and efficient communication between the user interface and the database. It is developed using Node.js and Express.js, forming a fast and scalable server-side environment.

RESTful API endpoints were designed for all key actions—creating, reading, updating, and deleting (CRUD) event data. These endpoints follow proper naming conventions and HTTP methods to standardize backend interactions. For example, GET requests fetch events, POST requests create new entries, PUT/PATCH modify existing data, and DELETE requests remove entries.

Each request is carefully validated through middleware to prevent empty or malformed submissions. This includes ensuring valid dates, non-empty fields, and correct formatting. Backend controllers handle responses efficiently by using try-catch blocks and error handlers that return appropriate HTTP status codes and messages.

All event data, comments, and likes are stored in MongoDB, a NoSQL database that is known for its flexibility and speed. With the help of Mongoose, a schema-based Object Data Modeling (ODM) library, the app maintains structure in its otherwise flexible data storage system. Collections are organized into documents representing events, and within each document, nested arrays store comments and likes.

A major advantage of MongoDB is its support for high-performance queries and indexing. Events are indexed based on date and title to optimize search and filter operations. The database also supports asynchronous operations, reducing server response time and improving user experience.

4.3 Frontend Design and Templating

The frontend was developed with a focus on usability, aesthetics, and accessibility. A combination of HTML5, CSS3, and JavaScript was used for structure and interactivity, while Bootstrap enhanced the responsiveness of the design. Layout components were selected to provide a clean and structured look across all pages.

The interface is dynamic and built using Handlebars (HBS), a templating engine that integrates seamlessly with Express.js. Handlebars allows server-rendered dynamic views that populate data fetched from the database in real time. The use of partials and layouts enables code reuse, reducing redundancy and simplifying maintenance.

For example, the same event card template is reused to display different sets of events—upcoming, popular, or filtered—while the layout template defines the consistent page structure (navbar, footer, and content container).

Form inputs, buttons, and navigation elements are clearly styled and placed for intuitive user interaction. Event details are presented in card format with bold headers, clearly visible timestamps, and logically arranged metadata. Alerts and toasts are also used to notify users when actions like event creation or deletion are successful.

All pages are optimized for mobile view, with collapsible menus, adjustable grid layouts, and touch-friendly buttons. Accessibility was also considered—elements have proper labels, and the interface supports keyboard navigation and screen readers to a basic extent.

4.4 User Interaction Modules

User engagement was a key area of focus during the development of this application. The inclusion of interactive features such as comments and likes not only enriches the user experience but also encourages participation and feedback.

- Comments: The comment system enables users to ask questions, share opinions, or post reviews about events. Comments are stored in the database and displayed under each event. This feature helps users make informed decisions before attending.

- Likes: The like button provides a quick way for users to express interest in an event. A count of total likes is displayed alongside each event, acting as a measure of popularity and community interest.

These features were implemented with security and performance in mind. Inputs are sanitized to prevent XSS attacks, and backend logic ensures that duplicate or spam comments are minimized. Data is updated instantly and re-rendered using Handlebars, maintaining a seamless user flow.

In future iterations, these interaction modules can be expanded with real-time WebSocket updates, comment threading, and user tagging to promote deeper engagement.

4.5 Search, Filter, and Sorting Mechanisms

To enhance discoverability and user convenience, the app includes a robust search and filter system. Users can quickly locate events using keywords in titles or descriptions. Filters allow narrowing down events based on date, location, or category, making it easy for users to focus on relevant results.

Search and filter requests are handled via route parameters and backend queries. The search field accepts partial strings and returns fuzzy matches using regular expressions in MongoDB. Sorting functionality ranks results by event date or popularity (number of likes), and this can be extended to include sorting by user ratings or number of comments.

This feature proved especially helpful in cases where multiple events were posted, allowing users to prioritize events based on personal preferences or urgency.

4.6 System Testing and Performance

System testing was carried out in stages, including unit testing of individual components, integration testing across modules, and user testing in real-world browsing scenarios. The application was tested with sample data for:

- Form validation (e.g., empty fields, invalid dates)
- CRUD operation consistency
- Comment and like behaviour
- Search and filter accuracy
- UI responsiveness on various screen sizes

From a performance perspective, MongoDB handled multiple read/write operations efficiently, and Express.js managed concurrent requests without bottlenecks. Load times remained minimal even with high-frequency data updates, thanks to asynchronous routing and database optimization.

4.7 Cross-Platform Compatibility

To ensure broad usability, the app was tested on various operating systems (Windows, macOS, Linux) and browsers (Chrome, Firefox, Edge). It rendered consistently across platforms, with no visual or functional discrepancies observed.

The responsive layout adapted to different device sizes, including smartphones and tablets. The use of Bootstrap grid systems and media queries ensured that the layout, buttons, and text scaled appropriately. This makes the app usable in academic, organizational, or public settings where users may access it from a range of devices.

4.8 Observations and Summary

The project successfully met its design and performance goals. Some noteworthy observations include:

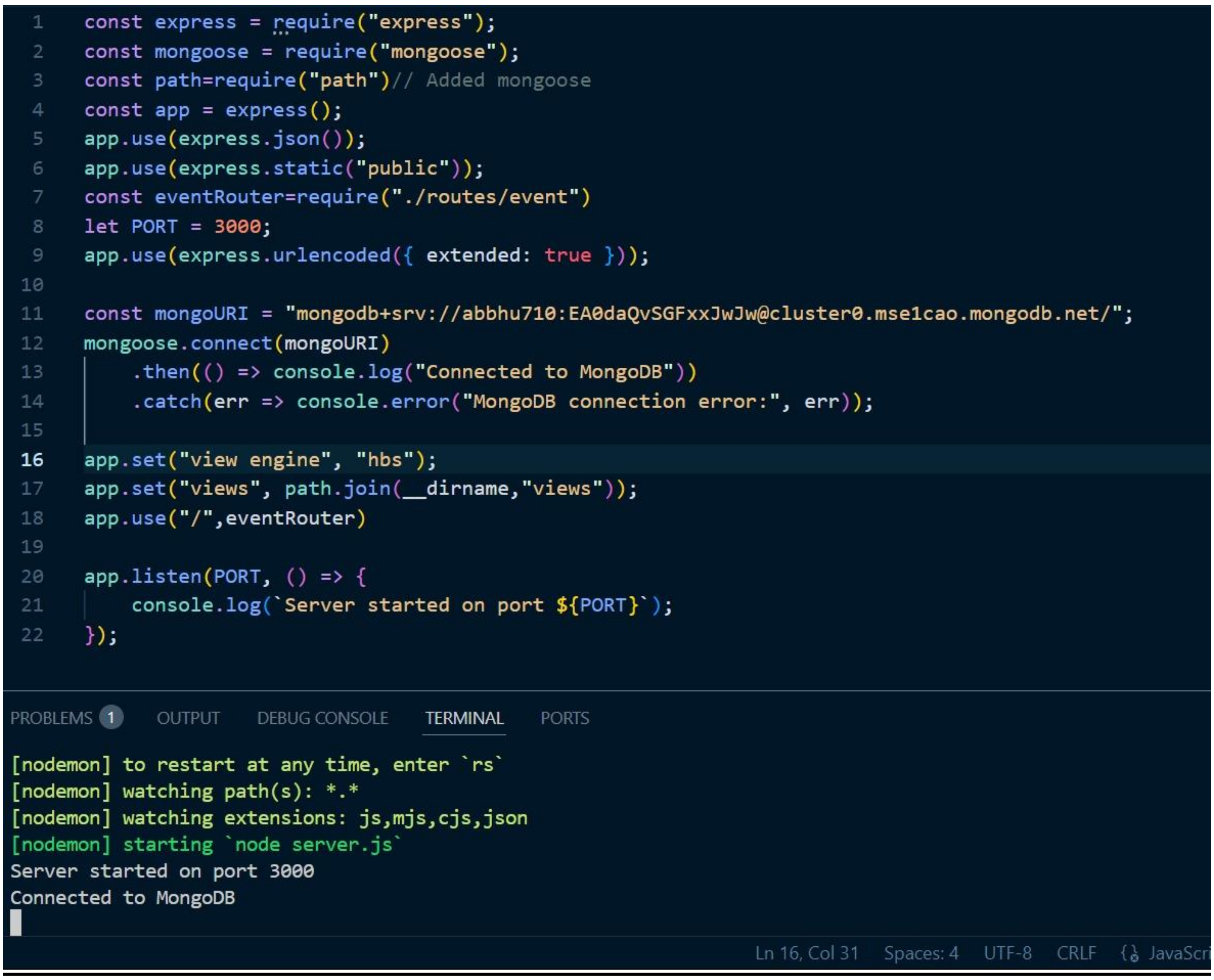
- The real-time update feature allowed hosts to edit events even after publishing, with immediate effect.
- The templating system enhanced maintainability and modularity.
- User interaction modules encouraged feedback and built a sense of community.
- The app’s speed and low error rate contributed to a smooth user experience.

While the current version covers core requirements, future iterations may incorporate:

- User authentication and role-based permissions
- Notification systems for reminders or changes
- Integration with external calendars (Google Calendar, Outlook)
- Admin dashboards with analytics and insights

In conclusion, Occasio proved to be a successful implementation of a practical and much-needed solution for managing, exploring, and engaging with events in a modern, intuitive, and scalable manner.

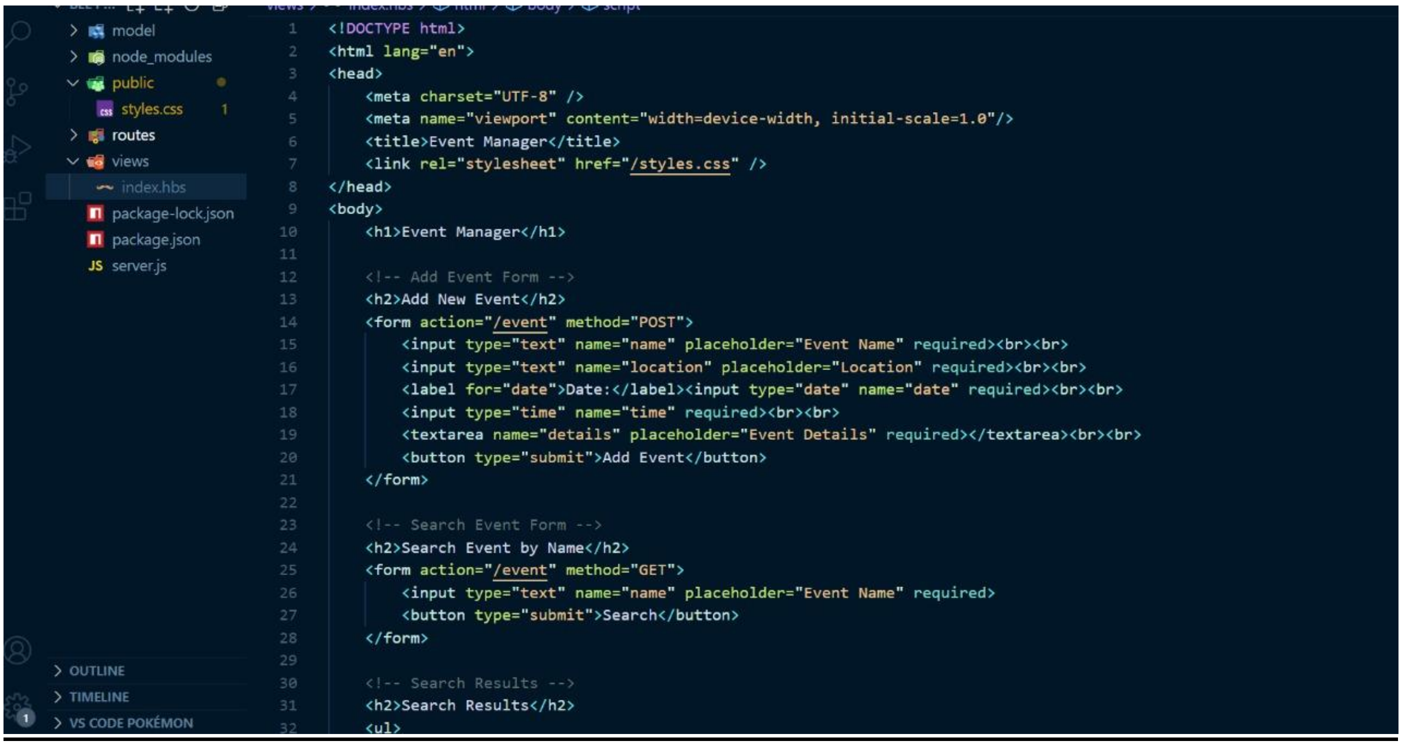
4.9 Project Screenshots



```
const express = require("express");
const mongoose = require("mongoose");
const path=require("path")// Added mongoose
const app = express();
app.use(express.json());
app.use(express.static("public"));
const eventRouter=require("./routes/event")
let PORT = 3000;
app.use(express.urlencoded({ extended: true }));

const mongoURI = "mongodb+srv://abbhu710:EA0daQvSGFxxJwJw@cluster0.mse1cao.mongodb.net/";
mongoose.connect(mongoURI)
    .then(() => console.log("Connected to MongoDB"))
    .catch(err => console.error("MongoDB connection error:", err));

app.set("view engine", "hbs");
app.set("views", path.join(__dirname,"views"));
app.use("/",eventRouter)

app.listen(PORT, () => {
    console.log(`Server started on port ${PORT}`);
});
```

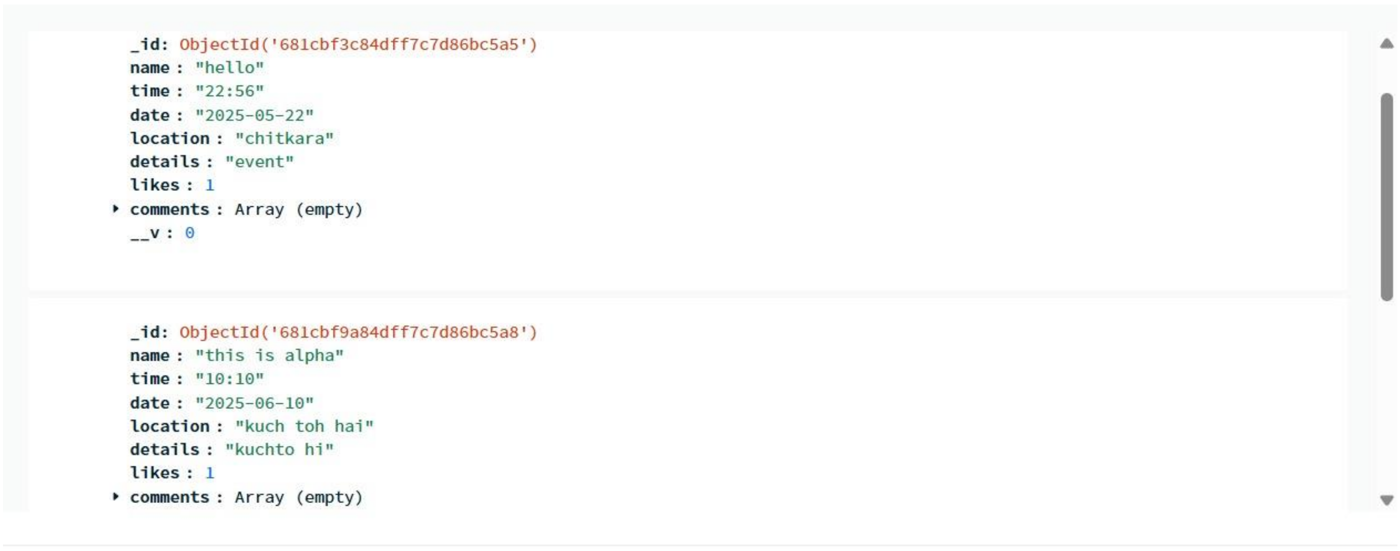
```
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node server.js`
Server started on port 3000
Connected to MongoDB
```

Figure 1.1



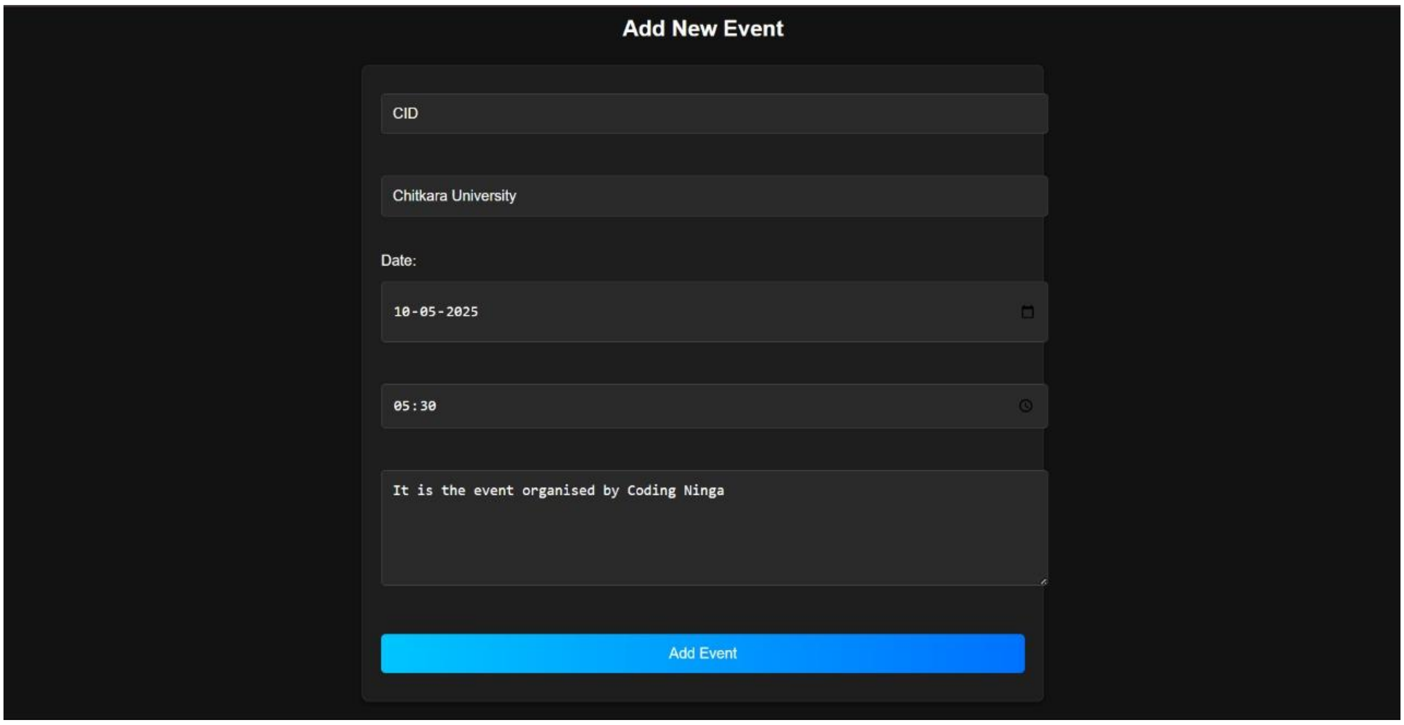
```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0"/>
    <title>Event Manager</title>
    <link rel="stylesheet" href="/styles.css" />
</head>
<body>
    <h1>Event Manager</h1>

    <!-- Add Event Form -->
    <h2>Add New Event</h2>
    <form action="/event" method="POST">
        <input type="text" name="name" placeholder="Event Name" required><br><br>
        <input type="text" name="location" placeholder="Location" required><br><br>
        <label for="date">Date:</label><input type="date" name="date" required><br><br>
        <input type="time" name="time" required><br><br>
        <textarea name="details" placeholder="Event Details" required></textarea><br><br>
        <button type="submit">Add Event</button>
    </form>

    <!-- Search Event Form -->
    <h2>Search Event by Name</h2>
    <form action="/event" method="GET">
        <input type="text" name="name" placeholder="Event Name" required>
        <button type="submit">Search</button>
    </form>

    <!-- Search Results -->
    <h2>Search Results</h2>
    <ul>
```

Figure 1.2



```
_id: ObjectId('681cbf3c84dff7c7d86bc5a5')
name : "hello"
time : "22:56"
date : "2025-05-22"
location : "chitkara"
details : "event"
likes : 1
comments : Array (empty)
__v : 0

_id: ObjectId('681cbf9a84dff7c7d86bc5a8')
name : "this is alpha"
time : "10:10"
date : "2025-06-10"
location : "kuch toh hai"
details : "kuchto hi"
likes : 1
comments : Array (empty)
```

Figure 1.3



CID

Chitkara University

10-05-2025

05:30

It is the event organised by Coding Ninga

Figure 1.4

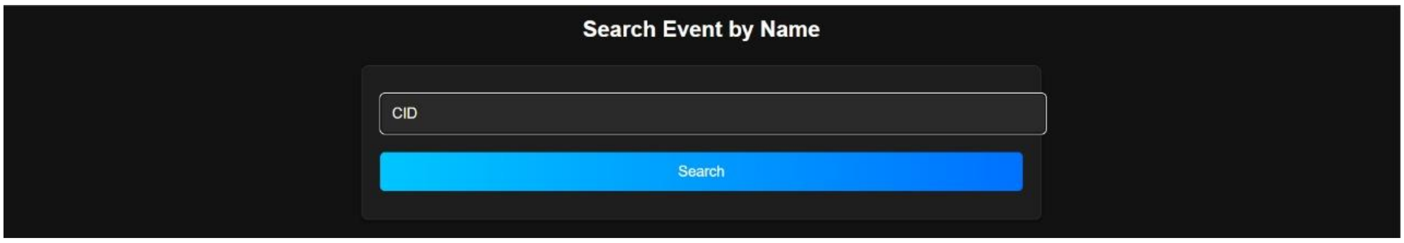
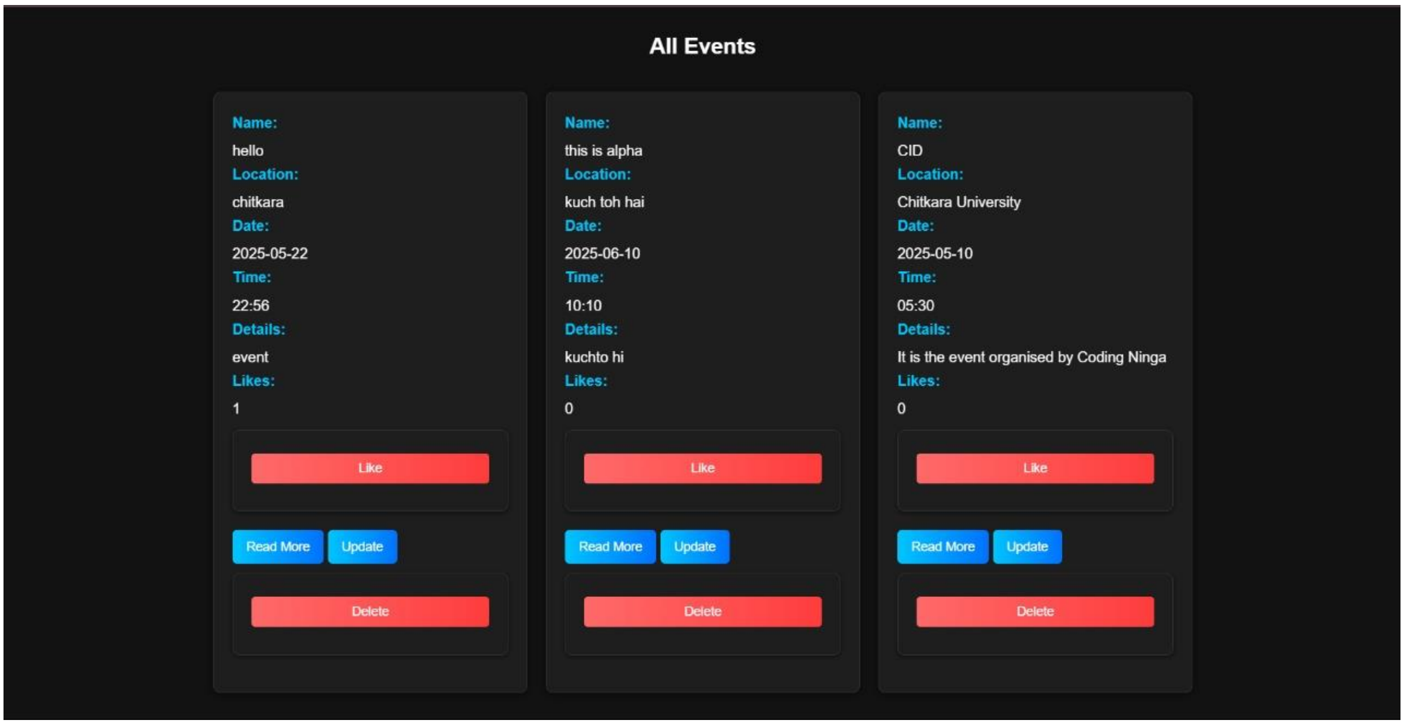


Figure 1.5



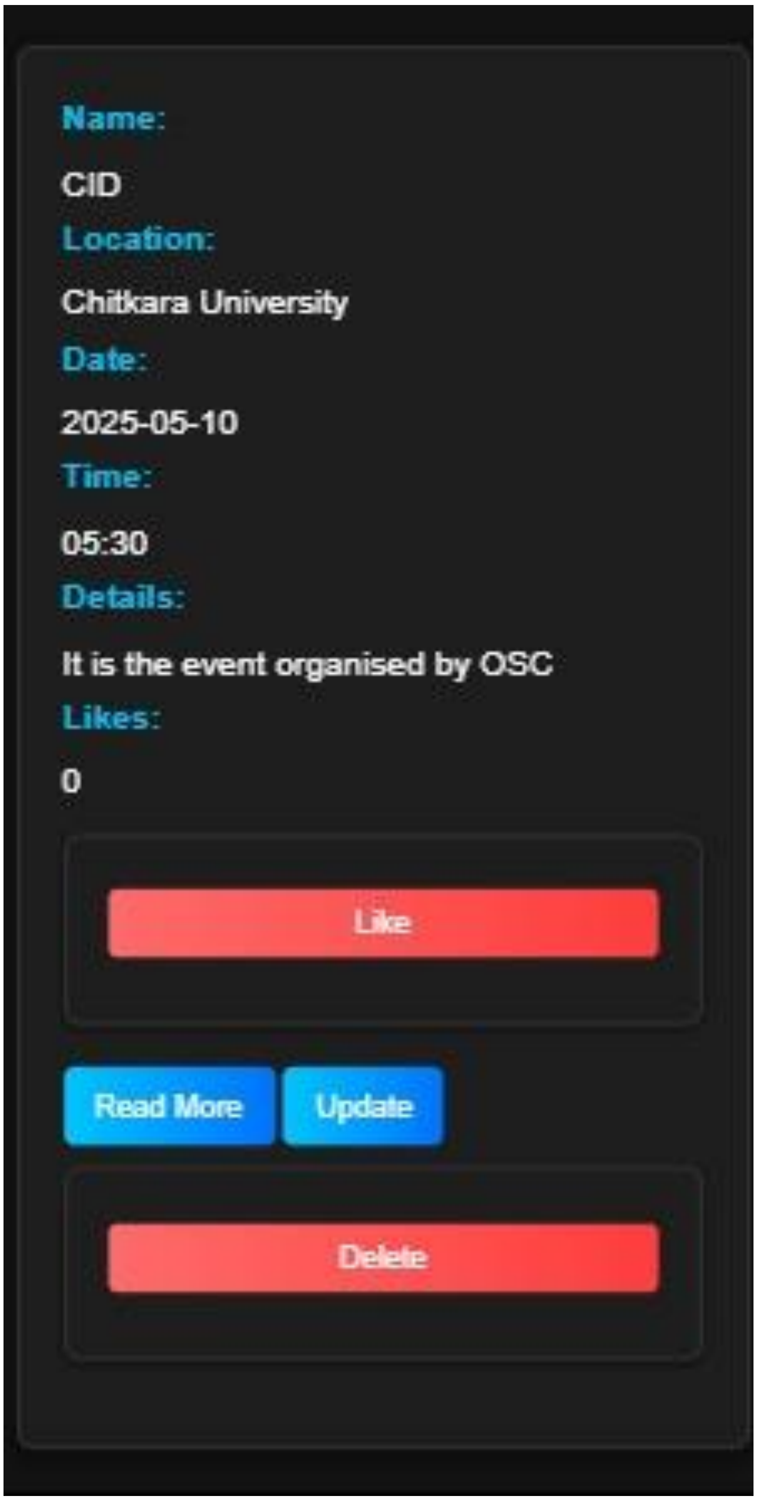
All Events

Name:
hello
Location:
chitkara
Date:
2025-05-22
Time:
22:56
Details:
event
Likes:
1

Name:
this is alpha
Location:
kuch toh hai
Date:
2025-06-10
Time:
10:10
Details:
kuchto hi
Likes:
0

Name:
CID
Location:
Chitkara University
Date:
2025-05-10
Time:
05:30
Details:
It is the event organised by Coding Ninga
Likes:
0

Figure 1.6


Name:

CID

Location:

Chitkara University

Date:

2025-05-10

Time:

05:30

Details:

It is the event organised by OSC

Likes:

0

Figure 1.7

5. References

- Node.js Documentation
- Express.js Guide
- React.js Documentation
- MongoDB Docs
- Handlebars.js
- Bootstrap Framework
- Font Awesome