

# **EVENTIFY**

**A PROJECT REPORT  
for  
Mini Project (KCA353)  
Session (2024-25)**

**Submitted by**

**Anchal Tyagi  
(2300290140023 )**

**Anuj Singh  
(2300290140029 )**

**Aakrti Sharma  
(2300290140001 )**

**Akshat Gautam  
(2300290140017 )**

**Submitted in partial fulfilment of the  
Requirements for the Degree of**

## **MASTER OF COMPUTER APPLICATION**

**Under the Supervision of  
Dr. Amit Kumar Gupta  
Professor**



**Submitted to**

**DEPARTMENT OF COMPUTER APPLICATIONS  
KIET Group of Institutions, Ghaziabad  
Uttar Pradesh-201206  
(DECEMBER 2024)**

## **CERTIFICATE**

Certified that **Anchal Tyagi (2300290140023)**, **Aakrti Sharma (2300290140001)**,  
**Anuj Singh (2300290140029)**, **Akshat Gautam (2300290140017)** have carried out  
the project work having “**Eventify**” (**Mini-Project-KCA353**) for **Master of  
Computer Application** from Dr. A.P.J. Abdul Kalam Technical University(AKTU)  
(formerly UPTU),Lucknow under my supervision. The project report embodies  
original work, and studies are carried out by the student himself/herself and the  
contents of the project report do not form the basis for the award of any other degree  
to the candidate or to anybody else from this or any other University/Institution.

**Dr. Amit Kumar Gupta**  
**Professor**  
**Department of Computer Applications**  
**KIET Group of Institutions, Ghaziabad**

**Dr. Arun Kumar Tripathi**  
**Head**  
**Department of Computer Applications**  
**KIET Group of Institutions,**  
**Ghaziabad**

## **EVENTIFY**

**Anchal Tyagi  
Aakrti Sharma  
Anuj Singh  
Akshat Gautam**

## **ABSTRACT**

Eventify is a comprehensive and user-friendly web application designed to transform how college events are organized, managed, and experienced. This innovative platform serves as a centralized hub for all campus activities, enabling students, faculty, and event coordinators to plan, manage, and participate in events effortlessly. By consolidating event information into a single accessible interface, Eventify simplifies event discovery and registration, enhancing overall engagement and participation within the campus community.

The platform offers a range of powerful features, including real-time collaboration tools for seamless communication among organizers, customizable event templates for efficient planning, and feedback mechanisms to improve future events. Additionally, Eventify provides automated scheduling to reduce manual errors, attendance tracking to monitor participation, and communication channels to keep users informed.

With its intuitive design and robust functionality, Eventify aims to address common challenges in event management, fostering a more inclusive and organized event culture on campus. By making event-related information readily available and improving coordination, Eventify enhances the event experience for all stakeholders, bringing innovation and efficiency to campus life.

## ACKNOWLEDGEMENTS

Success in life is never attained single-handedly. My deepest gratitude goes to my project supervisor, **Dr. Amit Kumar Gupta** for his/ her guidance, help, and encouragement throughout my project work. Their enlightening ideas, comments, and suggestions.

Words are not enough to express my gratitude to Dr. Arun Kumar Tripathi, Professor and Head, Department of Computer Applications, for his insightful comments and administrative help on various occasions.

Fortunately, I have many understanding friends, who have helped me a lot on many critical conditions.

Finally, my sincere thanks go to my family members and all those who have directly and indirectly provided me with moral support and other kind of help. Without their support, completion of this work would not have been possible in time. They keep my life filled with enjoyment and happiness.

**Anchal Tyagi**

**Akshat Gautam**

**Aakrti Sharma**

**Anuj Singh**

# TABLE OF CONTENTS

Certificate	ii
Abstract	iii
Acknowledgements	iv
Table of Contents	v-vii
List of Figures	viii
1 Introduction	1-2
1.1 Overview	1
1.2 Basic communicational model	2
1.3 Data communication	2
2 Literature Review	3-4
2.1 Introduction	3
2.2 Existing event management practices	3
2.2.1 Traditional methods	3
2.2.2 Existing digital tools	4
2.3 Challenges in current practices	4
3 Methodology	5-7
3.1 Overview	5
3.2 Tools and technologies used	5-6
3.3 Development process	6
3.3.1 Planning	6
3.3.2 Designing	6
3.3.3 Implementation	6

3.3.4	Testing	7
3.3.5	Deployment	7
4	System Design	8-11
4.1	Overview	8
4.2	System Architecture	8
4.3	Key modules and components	8
4.3.1	Event modules	8
4.3.2	Registration module	8
4.3.3	User management module	9
4.4	Workflow of eventify	9
4.4.1	Event creation	9
4.4.2	User registration	9
4.5	Diagrams	10-11
5	Implementation	12-15
5.1	Overview	12
5.2	Development requirement	12-13
5.3	Deployment hosting services	14
5.3.1	Frontend	14
5.3.2	Backend	14
5.3.3	Database	14
5.3.4	Integration	14
5.3.5	Steps in deployment	14
5.4	Implementation challenges	14
5.4.1	Ensuring scalability	14
5.4.2	Security measures	14
5.4.3	Cross platform compatibility	14
5.4.4	Real-time features	14
5.5	Future plans	15

6	Testing	16-18
6.1	Testing methodology	16
6.1.1	Unit testing	16
6.1.2	Integration testing	16
6.1.3	User acceptance testing(UAT)	16
6.2	Test cases	17
6.3	Results	17
6.4	Conclusion	18
7	Conclusion and future scope	19-21
7.1	Conclusion	19
7.2	Achievements	19
7.3	Challenges faced	20
7.4	Future scope	20
7.5	Lessons learned	21
7.6	Final thoughts	21
8	Appendix	22-36
8.1	DFD	22
8.2	Admin frontend code	23-25
8.3	User frontend code	26-28
8.4	Backend code	29-32
8.5	Snapshots	33-36
9	References	37-38

## LIST OF FIGURES

<b>Name of Figure</b>	<b>Page No.</b>
Basic Communication model	2
Agile Model	5
ER Diagram Of user side	9
ER Diagram of admin side	10
Use case diagram	11
Test cases	17
DFD	22



# CHAPTER 1

## INTRODUCTION

### 1.1 OVERVIEW

In today's fast-paced environment, organizing and managing events has become increasingly complex. Traditional methods often involve cumbersome paperwork, manual registrations, and limited communication channels. Eventify is a modern, web-based event management platform designed to simplify the process of planning and executing events in a college setting.

Eventify serves as a comprehensive solution, allowing event organizers, participants, and attendees to interact seamlessly within a unified system. The platform integrates features such as event creation, registration, scheduling, notifications, and ticketing, all within an intuitive and user-friendly interface.

With Eventify, users can easily create, promote, and manage events, while ensuring a smooth and efficient experience for all parties involved. The system leverages advanced technologies like MongoDB for data management and React for responsive, dynamic user interfaces.

The core objectives of Eventify are:

- **Ease of Use:** Offering a simple and engaging user experience for both organizers and attendees.
- **Real-Time Communication:** Facilitating live updates, notifications, and interactive features.
- **Automation:** Streamlining event registration, scheduling, and management to save time and reduce errors.
- **Scalability:** Ensuring that the platform can handle events of various sizes, from small gatherings to large-scale conferences.

## 1.2 BASIC COMMUNICATION MODEL

In the context of Eventify, the communication model refers to the data exchange process between different components of the system to facilitate seamless interaction among event organizers, participants, and attendees. **Figure 1.1** depicts this flow, from event discovery to registration, event updates, and feedback.

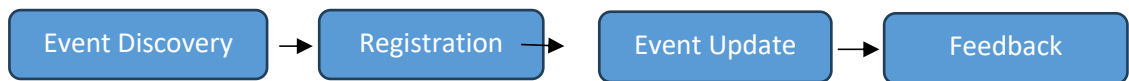


Fig. 1.1 Basic Communication Model

## 1.3 DATA COMMUNICATION

Data communication is a fundamental aspect of Eventify, ensuring that all users whether event organizers, participants, or attendees—can interact seamlessly with the platform. In the case of Eventify, both local and remote communication systems are utilized. Eventify leverages data communication in the following ways:

### 1. Local Data Communication

Local communication takes place when the communicating devices are in the same geographical area, same building, or face-to-face etc.

### 2. Remote Data Communication

Remote communication takes place over a distance i.e. the devices are farther. The effectiveness of a data communication can be measured through the following features:

**Delivery:** Delivery should be done to the correct destination.

**Timeliness:** Delivery should be on time.

**Accuracy:** Data delivered should be accurate.

## CHAPTER 2

### LITERATURE REVIEW

#### 2.1 INTRODUCTION

The organization and management of events in educational institutions play a pivotal role in fostering student engagement, professional development, and campus vibrancy. Despite their importance, managing events efficiently remains a challenge due to fragmented communication channels and lack of centralized systems. This chapter provides an in-depth review of existing event management practices and platforms, identifies their limitations, and highlights the necessity for a unified system like Eventify.

#### 2.2 EXISTING EVENT MANAGEMENT PRACTICES

##### 2.2.1 Traditional Methods

Historically, colleges have relied on manual methods to manage events. Common approaches include.

**Posters and Notice Boards:** These serve as primary tools for event promotion but lack real-time updates and have limited reach.

**Word-of-Mouth Communication:** This informal method often leads to incomplete or inaccurate dissemination of information.

**Emails and Messaging Apps:** While widely used, these tools are not designed.

##### 2.2.2 Existing Digital Tools

With advancements in technology, many colleges have adopted digital tools such as:

**Google Forms and Sheets:** These are used for registration and tracking but lack advanced features for real-time updates and integration.

**Eventbrite and Meetup:** These platforms provide functionalities like event promotion and ticketing but are not tailored for educational institutions, resulting in limited usability.

## 2.3 CHALLENGES IN CURRENT PRACTICES

A critical analysis of existing practices reveals the following challenges:

- **Lack of Integration:** Event data is scattered across multiple platforms, making it difficult to track and manage.
- **Communication Gaps:** Event updates often fail to reach all stakeholders in a timely manner.
- **Inefficient Scheduling:** Conflicts between overlapping events lead to reduced participation and engagement.
- **Manual Processes:** Reliance on manual methods increases the chances of errors and delays in event management.
- **Limited Accessibility:** Lack of centralized access makes it harder for users to stay informed about events.
- **Inconsistent Data:** Discrepancies in event information across platforms create confusion for users.
- **Time-Consuming Coordination:** Collaboration among event organizers is often unstructured and time-intensive.
- **Limited User Engagement:** Current systems do not adequately incentivize or track user participation.
- **Inadequate Feedback Mechanisms:** There is no systematic approach to collecting and utilizing user feedback for event improvements.
- **Resource Management Challenges:** Inefficient allocation and tracking of event-related resources, such as venues and equipment.

## CHAPTER 3

### METHODOLOGY

#### 3.1 OVERVIEW

The development of Eventify followed the Agile methodology, ensuring iterative improvements and constant feedback.



**Fig. Agile Model**

Fig. 3.1 Agile Model

#### 3.2 TOOLS AND TECHNOLOGIES USED

##### Frontend:

- **React.js:** A popular JavaScript library used to build dynamic and interactive user interfaces. React.js enables developers to create reusable components,

**ensuring** consistency and speeding up development. Its virtual DOM improves performance, providing a smooth and responsive user experience.

- **Tailwind CSS:** A utility-first CSS framework that allows developers to design directly in the markup. Tailwind CSS helps in creating a clean, professional, and consistent look for the platform, with minimal custom CSS required.

#### **Backend:**

- **Node.js and Express.js:** Node.js is a JavaScript runtime that enables server-side programming. Paired with Express.js, a web application framework, it facilitates efficient handling of server-side processes, API development, and seamless data flow between the frontend and backend.

#### **Database:**

- **MongoDB:** A NoSQL database used for storing and retrieving data. MongoDB is well-suited for handling diverse data types and supports high scalability, ensuring secure storage of event details, user information, and feedback.

#### **Other Tools:**

- **Visual Studio Code:** A widely used code editor that provides powerful features like syntax highlighting, debugging tools, and extensions, enhancing the productivity of developers.
- **Git:** A version control system that tracks code changes and facilitates collaboration among team members, ensuring smooth coordination and conflict resolution during development.
- **Postman:** An API testing tool used to validate and debug endpoints during development. Postman simplifies the process of ensuring that the backend APIs perform as expected.

### **3.3 DEVELOPMENT PROCESS**

#### **3.3.1 Planning**

Identified the key problems with current event management systems.

Collected requirements by talking to students, faculty, and event coordinators.

#### **3.3.2 Designing**

Created wireframes and mockups to visualize the platform's layout.

Designed workflows for event creation, registration, and real-time updates.

### **3.3.3 Implementation**

Developed features one by one:

**Event Creation:** A form to create and edit event details.

**Registration System:** Allows users to register for events.

Tested each feature to make sure it worked smoothly.

### **3.3.4 Testing**

Conducted unit testing for individual features.

Performed integration testing to check if all features worked well together.

### **3.3.5 Deployment**

Deployed the platform on a cloud server to make it accessible to everyone in the college.

## CHAPTER 4

### SYSTEM DESIGN

#### 4.1 OVERVIEW

System design is the blueprint for how a project functions and interacts with users. This chapter explains the architecture, key components, and workflows of Eventify. It outlines how the platform was structured to meet the goal of centralizing college event management.

#### 4.2 SYSTEM ARCHITECTURE

The platform follows a **three-tier architecture**:

##### **Presentation Layer**

The frontend, developed using **React.js**, provides an interactive interface for users like students, faculty, and event coordinators.

##### **Business Logic Layer**

The backend, built with **Node.js**, handles event data, user interactions, and application logic.

##### **Data Layer**

**MongoDB** is used for storing event details, user profiles, registrations, and feedback.

This layered approach ensures that the system is scalable, maintainable, and efficient.

#### 4.3 KEY MODULES AND COMPONENTS

##### **4.3.1 Event Module**

Allows event creation, editing, and deletion.



Displays all events in a centralized calendar view.

#### **4.3.2 Registration Module**

Users can register for events and view their registered events in their dashboard.

Coordinators can track registrations in real time.

#### **4.3.3 User Management Module**

Supports role-based access:

**Students:** View and register for events.

**Faculty/Organizers:** Create and manage events.

**Admins:** Oversee the entire platform.

### **4.4 WORKFLOW OF EVENTIFY**

#### **4.4.1 Event Creation**

Organizers log in and access the event creation form.

They fill out details like event name, date, time, and category.

The system checks for scheduling conflicts and saves the event in the database.

#### **4.4.2 User Registration**

Users browse events and click "Register."

Their dashboard is updated to display all registered events.

## 4.5 DIAGRAMS

### 4.5.1 ER Diagram

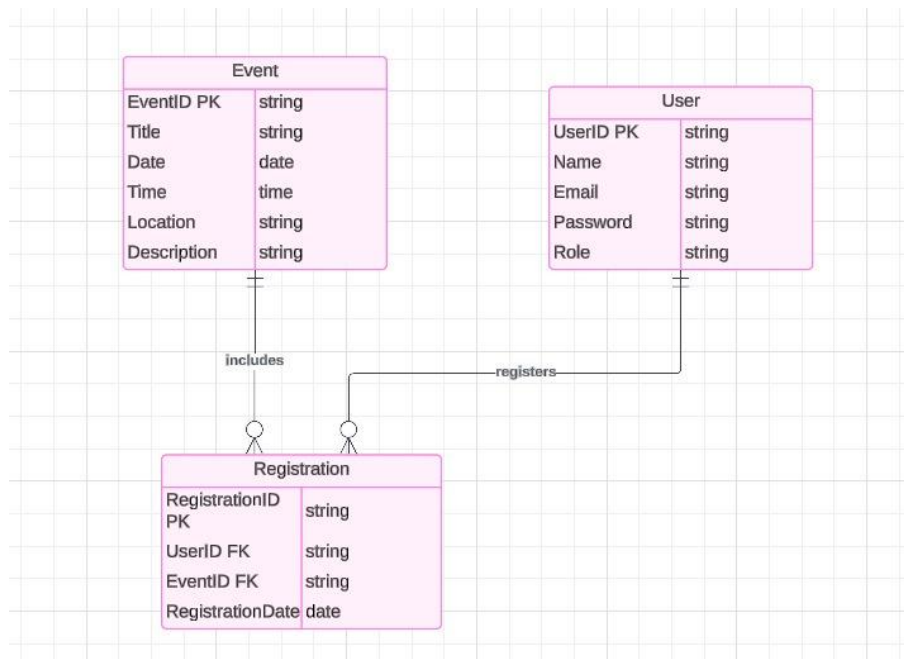


Fig. 5.1 ER Diagram of User Side

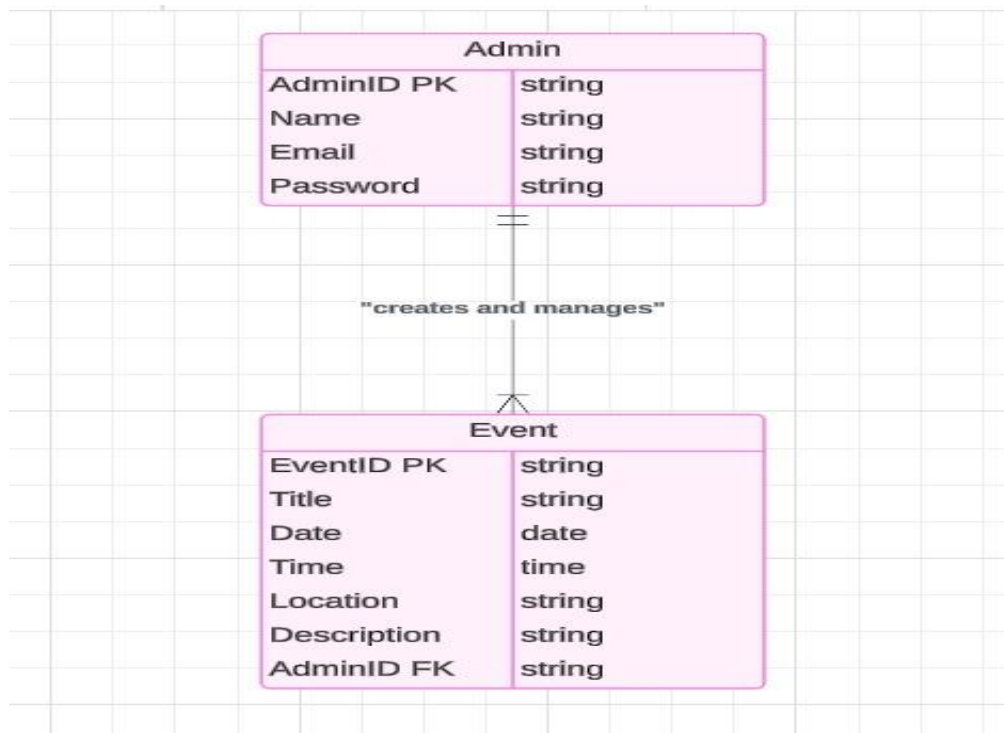


Fig. 5.2 ER Diagram of Admin Side

### 4.5.2 Use Case Diagram

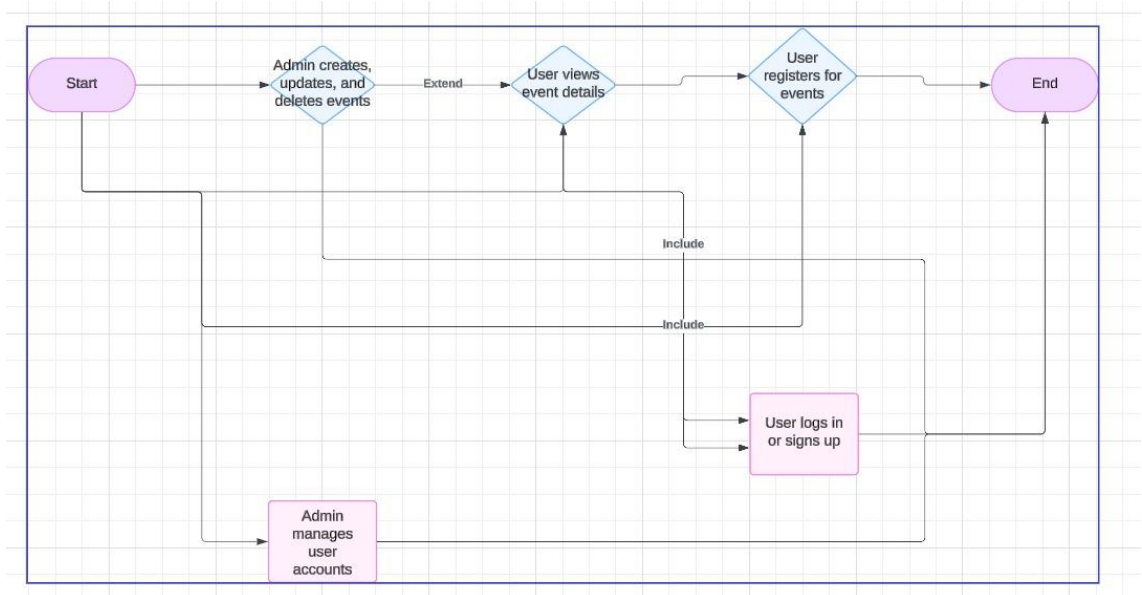


Fig. 5.3 Use case diagram

## CHAPTER 5

### IMPLEMENTATION

#### 5.1 OVERVIEW

The implementation phase for Eventify involved deploying the application in a structured manner, ensuring functionality, scalability, and usability. The process consisted of configuring the development environment, integrating the components, and deploying the platform for end-users.

#### 5.2 DEVELOPMENT REQUIREMENT

##### Frontend Development

##### Technologies Used:

**HTML, CSS, JavaScript, React:** These are core technologies for building web applications.

**HTML** structures the content of web pages.

**CSS** styles the user interface.

**JavaScript** adds interactivity.

**React** is a library used to create dynamic, reusable components.

**Tailwind CSS:** A utility-first CSS framework used for styling. It enables rapid development of responsive and modern designs through pre-defined classes.

##### Implementation:

**Developed user interfaces for login, signup, event listing:** This involves creating visually appealing and functional screens where:

Users can log in or register (authentication forms).

Events are listed dynamically, fetched from the backend.

**Ensured responsive design to support various devices:** Tailwind CSS and React ensure that the application adapts to different screen sizes (mobile, tablet, desktop).

## **Backend Development**

### **Technologies Used:**

**Node.js:** A runtime environment for building server-side applications. It processes backend logic, handles API requests, and manages databases.

### **Implementation:**

**Handled user authentication:** Created secure login/signup systems with encrypted password storage (e.g., bcrypt) and session/token-based management (e.g., JWT).

**Event creation and registration processes:** Enabled admins or users to create events and allowed participants to register. Backend APIs validate and store this data in the database.

**Integrated role-based access control for Admin and User functionalities:**  
Established permissions:

**Admin:** Can manage events, view all registrations, and access detailed reports.

**User:** Can register for events, provide feedback, and view their registered events.

## **Database Management**

### **Database Used:**

### **MySQL or MongoDB:**

**MySQL:** A relational database that uses structured tables with rows and columns.

**MongoDB:** A NoSQL database that stores data in JSON-like documents, offering flexibility in schema design.

### **Implementation:**

**Created tables for Users, Events, Registrations, and Feedback:**

**Users Table:** Stores user data such as name, email, password, and role (Admin/User).

**Events Table:** Stores details like event title, description, date, and organizer.

**Registrations Table:** Tracks which user registered for which event.

**Feedback Table:** Captures user feedback for specific events.

**Used indexing for faster query execution:**

Indexing improves database performance by creating an efficient lookup structure for frequently queried fields (e.g., email in the Users table or event\_id in Registrations).

For example, indexing the event\_id column allows faster retrieval of all registrations for a specific event.

## **5.3 DEPLOYMENT HOSTING SERVICES**

### **5.3.1 Frontend :**

Deployed on Vercel for a fast and reliable user interface.

### **5.3.2 Backend:**

Hosted on Vercel to manage APIs and server-side logic.

### **5.3.3 Database:**

Hosted on a cloud-based service like MongoDB Atlas.

### **5.3.4 Integration:**

Configured the backend to interact seamlessly with the database and frontend. Established secure connections using HTTPS and encryption techniques

### **5.3.5 Steps in Deployment:**

Set up a repository for version control using Git and GitHub.

Configured the server environment, ensuring dependencies were installed.

Deployed backend APIs and connected the frontend with API endpoints.

Linked the database and ensured it was ready to handle production data..

## **5.4 IMPLEMENTATION CHALLENGES**

### **5.4.1 Ensuring scalability:**

Optimized the database schema to handle large volumes of data efficiently.

Configured load balancing for seamless performance during high traffic

#### **5.4.2 Security Measures:**

Implemented encryption for sensitive data like passwords using bcrypt.

Ensured secure authentication tokens using JWT (JSON Web Tokens).

#### **5.4.3 Cross Platform Compatibility:**

Tested the application on various browsers and devices to ensure consistent Performance.

#### **5.4.4 Real-time Features:**

Prepared the system for future upgrades to include notifications and real-time

### **5.5 FUTURE PLANS :**

The future plans for the platform include expanding its capabilities to support personalized event recommendations based on user preferences by integrating AI driven algorithms that analyze user behavior and interests. Additionally, an enhanced admin dashboard with real-time analytics, predictive tools, and customizable reporting features will be implemented to give admins deeper insights into event performance and user engagement. Advanced features such as live streaming of events, real-time participant tracking, and virtual networking rooms will be introduced to enrich user interaction. The platform will also focus on improving user engagement through community-building features, gamification, and more robust feedback systems. Enhancements in event discovery, including dynamic calendars and mobile app optimization, along with multi-language support and monetization options like sponsorships and premium features, will make the platform more accessible, engaging, and profitable. These updates will ensure the platform evolves into a comprehensive tool for managing and participating in events.

## **CHAPTER 6**

### **TESTING**

#### **6.1 TESTING METHODOLOGY**

The Eventify platform underwent rigorous testing to ensure that it met all functional, performance, and usability requirements. The testing methodology involved the following approaches:

##### **6.1.1 Unit Testing :**

Verified individual components of the system, such as the event creation form, registration module, and user authentication.

Ensured that each function performed as expected in isolation.

##### **6.1.2 Integration Testing :**

Tested the interaction between different modules, such as how the User Registration module integrates with the Events module.

Ensured that data flow between components was seamless and error-free.

##### **6.1.3 User Acceptance Testing (UAT):**

Conducted testing sessions with college stakeholders, including students, faculty, and administrators.

Gathered feedback on the platform's usability, responsiveness, and

Overall effectiveness.



## 6.2 TEST CASES

The table below summarizes the key test cases performed during the testing phase:

Test ID	Test Case Description	Expected Outcome	Actual Outcome	Status
TC-01	Event creation with valid data.	Event saved successfully and listed.	As expected.	Pass
TC-02	Event creation without mandatory fields.	Error message displayed: "All fields are required."	As expected.	Pass
TC-03	User registration with valid data.	User account created successfully.	As expected.	Pass
TC-04	User registration with invalid email.	Error message displayed: "Invalid email address."	As expected.	Pass
TC-05	Registration for an event by a user.	Registration confirmation displayed.	As expected.	Pass

## 6.3 RESULTS

The testing phase yielded the following results:

### Test Case Results :

All test cases passed successfully without any critical defects.

Minor UI inconsistencies were identified and resolved promptly.

### System Performance :

The platform met all performance benchmarks, handling concurrent user without delays.

Average response time remained well within acceptable limits.

### User Feedback:

Positive feedback was received regarding the platform's ease of use and reliability.

Suggestions for real-time notifications and enhanced analytics were noted for future improvements.

## **6.4 CONCLUSION**

The comprehensive testing process ensured that Eventify is a reliable and efficient platform for managing college events. By addressing identified issues during testing, the platform is now ready for deployment, offering a seamless experience to users and administrators alike. Future updates will incorporate additional features based on user feedback to further enhance the system's capabilities.

## CHAPTER 7

### CONCLUSION AND FUTURE SCOPE

#### 7.1 CONCLUSION

Eventify was developed as a centralized platform to simplify and enhance the process of managing and participating in college events. It successfully bridges the gap between event organizers, administrators, and participants by offering a unified solution for event creation, management, and registration. The project achieved its primary objectives of:

- a. Providing a user-friendly interface for students and faculty to explore events.
- b. Enabling administrators to efficiently manage and update event details.
- c. Ensuring seamless user interaction with features like login, registration, and event tracking.
- d. The platform fosters greater engagement, transparency, and collaboration within the college community, making event management more efficient and accessible.

#### 7.2 ACHIEVEMENTS

The table below summarizes the key test cases performed during the testing phase: Key accomplishments of the Eventify project include:

1. **Centralized Event Management:** Consolidated all college events into a single platform for better accessibility
2. **Role-Based Functionality:** Implemented distinct features for users and administrators.
3. **Scalable and Secure Architecture:** Designed the platform to handle multiple users concurrently while maintaining data security.

4. **Enhanced User Experience:** Ensured a responsive and intuitive user interface.
5. **Scalable and Secure Architecture:** Designed the platform to handle multiple users concurrently while maintaining data security.

### 7.3 CHALLENGES FACED

During the development and implementation phases, the following challenges were encountered:

- 1) **Database Optimization:** Ensuring quick data retrieval for a large number of users and events.
- 2) **Security Concerns:** Implementing robust mechanisms to protect sensitive data like login credentials.
- 3) **User Feedback Integration:** Balancing immediate requirements with planned future enhancements.
- 4) **Scalability:** Preparing the platform to handle increased usage as more colleges or departments adopt the system.

These challenges were addressed through iterative development, thorough testing, and continuous improvement.

### 7.4 FUTURE SCOPE

Eventify has significant potential for future development and growth. Some proposed enhancements include:

1. **Real-Time Notifications:** Integration of notifications for event updates, reminders, and announcements
2. **Mobile Application:** Developing a mobile app to provide users with greater convenience and accessibility.
3. **Advanced Analytics for Admins:** Adding reporting tools to give insights into event performance and user engagement metrics.
4. **Social Media Integration:** Allowing users to share events on social platforms to increase participation and visibility.
5. **Gamification:** Introducing rewards, leaderboards, and badges to incentivize participation and interaction.

6. **Multi-Language Support:** Expanding the platform's accessibility by supporting multiple languages.
7. **Event Feedback Mechanism:** Adding a feature for collecting post-event feedback to improve future events.

## 7.5 LESSONS LEARNED

- The importance of user-centric design in enhancing platform usability.
- The need for robust security practices in web application development.
- The value of continuous testing to ensure system reliability.
- Effective teamwork and communication are critical to the success of any project.
- Iterative development approaches enable better adaptability to changing requirements and feedback.

The value of continuous testing to ensure system reliability.

## 7.6 FINAL THOUGHTS

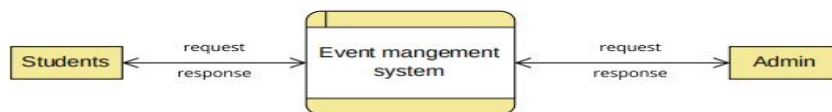
Eventify has proven to be a robust and efficient solution for managing college events, benefiting both organizers and participants. With planned enhancements and regular updates, the platform has the potential to become an indispensable tool for colleges, fostering a more engaging and connected campus environment. Eventify represents a step forward in digitizing and streamlining event management, aligning with the growing need for technology-driven solutions in education.

Moreover, the lessons learned from this project provide a strong foundation for tackling future challenges and innovating further in the domain of educational technology. By integrating emerging technologies and maintaining a focus on user satisfaction, Eventify can continue to evolve and make a meaningful impact in the academic community.

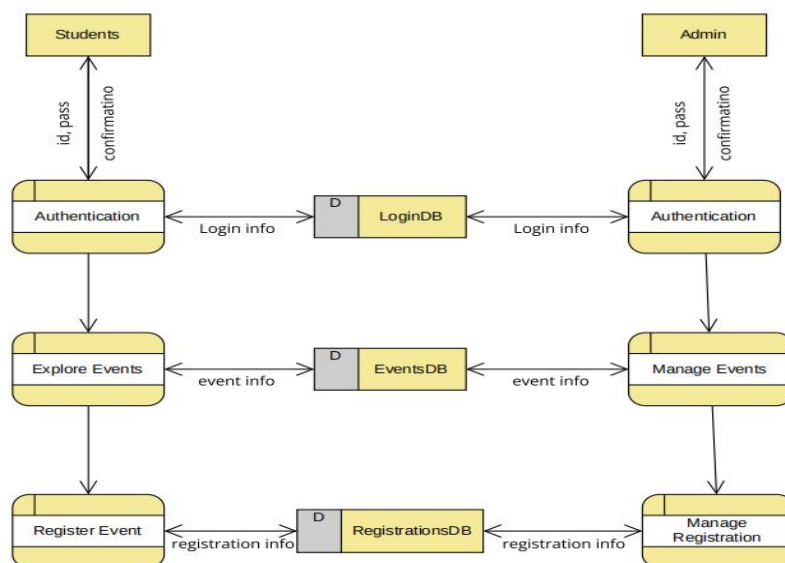
## CHAPTER 9

## APPENDIX

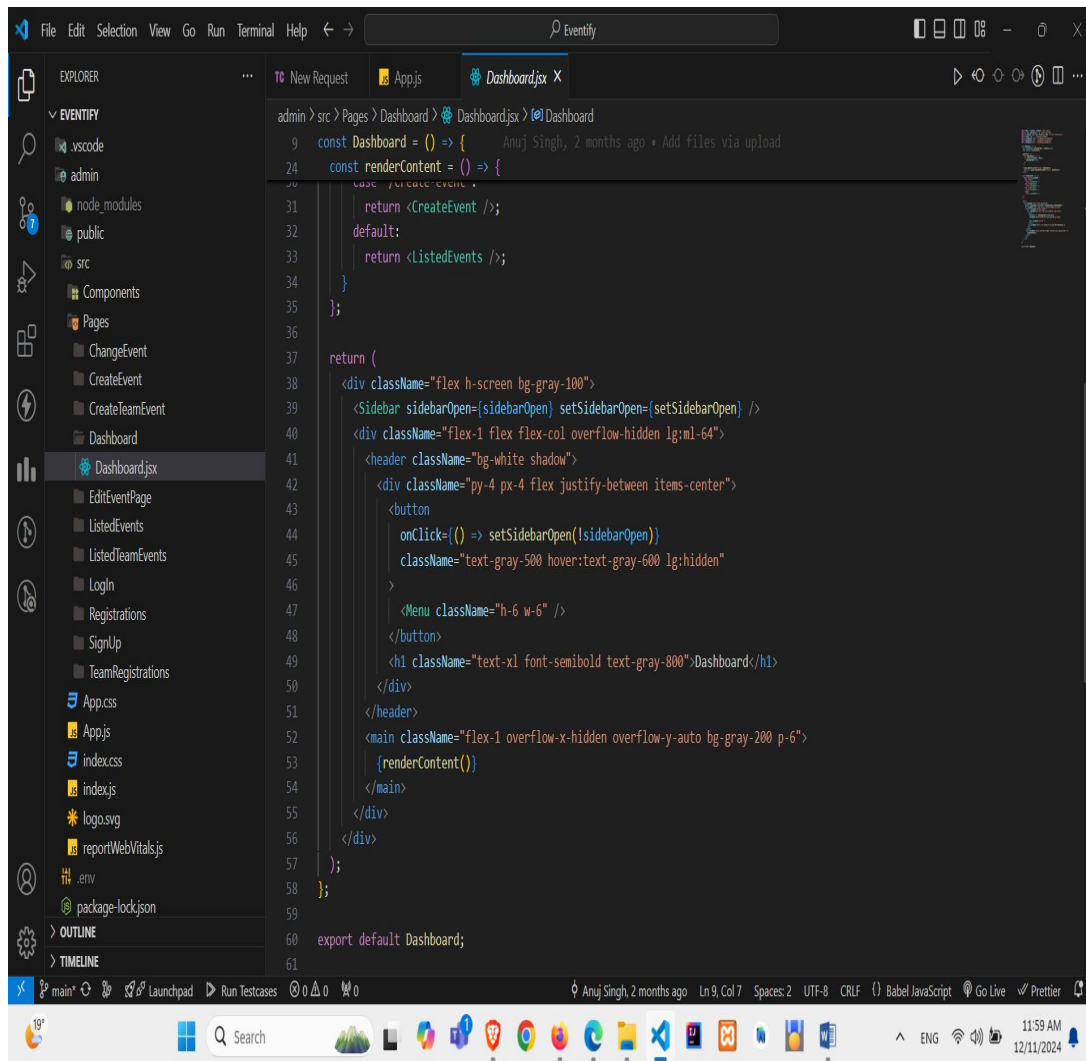
### 8.1 0 level DFD



### 8.2 1 level DFD



### 8.3 ADMIN FRONTEND CODE



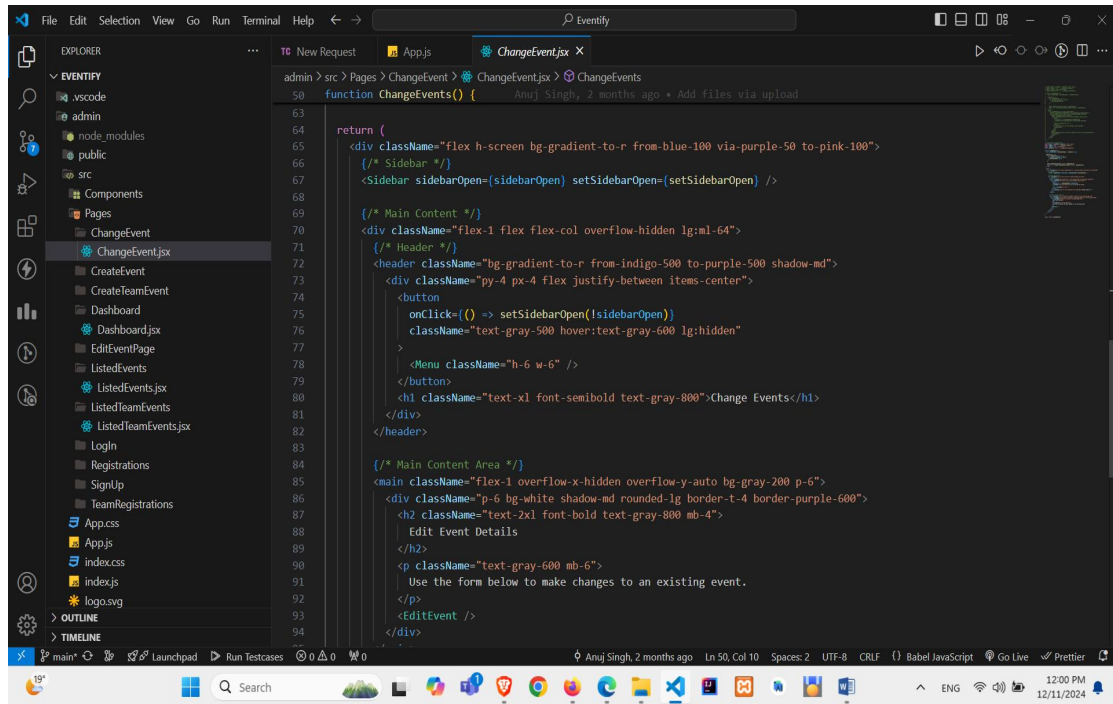
The screenshot shows the VS Code editor with the `Dashboard.jsx` file open. The Explorer sidebar on the left shows the project structure, with `Dashboard` selected under `Pages`. The main editor area displays the following code:

```
admin > src > Pages > Dashboard > Dashboard.jsx > [9] Dashboard
9  const Dashboard = () => {
24  const renderContent = () => {
31      return <CreateEvent />;
32      default:
33      return <listedEvents />;
34  }
35  };
36
37  return (
38    <div className="flex h-screen bg-gray-100">
39      <Sidebar sidebarOpen={sidebarOpen} setSidebarOpen={setSidebarOpen} />
40      <div className="flex-1 flex flex-col overflow-hidden lg:ml-64">
41        <header className="bg-white shadow">
42          <div className="py-4 px-4 flex justify-between items-center">
43            <button
44              onClick={() => setSidebarOpen(!sidebarOpen)}
45              className="text-gray-500 hover:text-gray-600 lg:hidden">
46              <Menu className="h-6 w-6" />
47            </button>
48            <h1 className="text-xl font-semibold text-gray-800">Dashboard</h1>
49          </div>
50        </header>
51        <main className="flex-1 overflow-x-hidden overflow-y-auto bg-gray-200 p-6">
52          {renderContent()}
53        </main>
54      </div>
55    </div>
56  );
57  };
58  };
59
60  export default Dashboard;
61
```

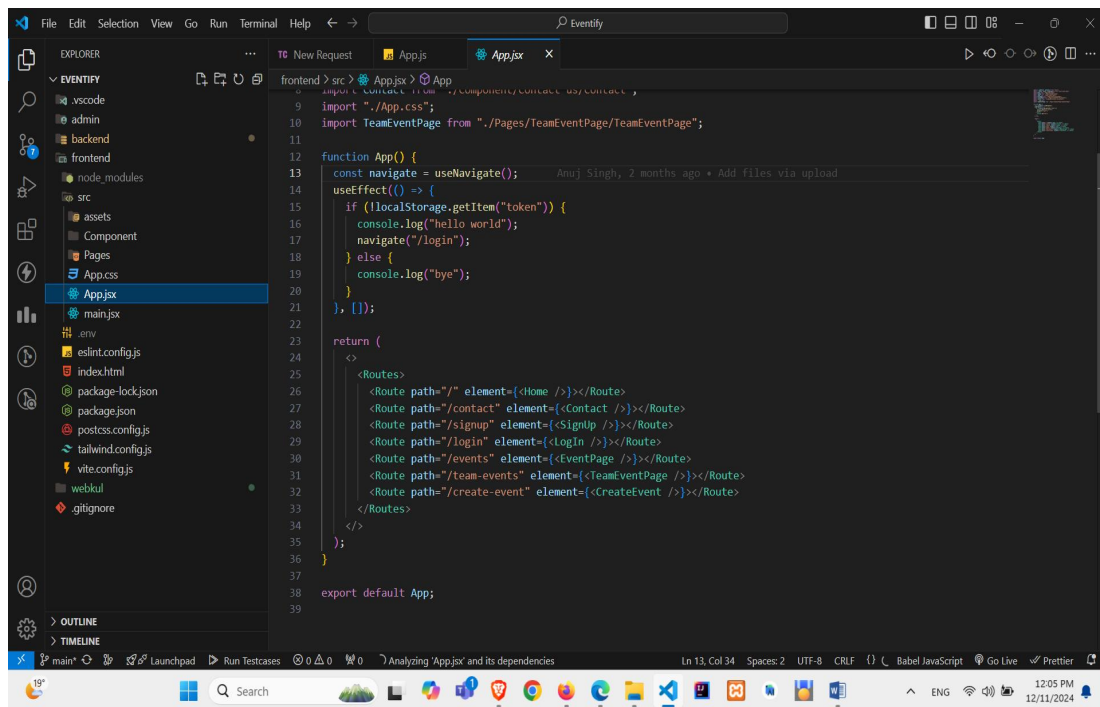
The status bar at the bottom indicates the file is at line 9, column 7, using UTF-8 encoding and CRLF line endings. It also shows that Babel JavaScript, Go Live, and Prettier are active.







## 8.4 USER FRONTEND CODE



The screenshot shows a Visual Studio Code editor window with the 'Eventify' project open. The Explorer sidebar on the left shows the file structure, with 'App.jsx' selected under the 'frontend' directory. The main editor area displays the code for 'App.jsx'. The code imports 'Contact' and 'TeamEventPage', defines a 'useEffect' hook to check for a 'token' in local storage, and sets up a routing system with several routes including Home, Contact, Signup, Login, EventPage, TeamEventPage, and CreateEvent. The status bar at the bottom indicates the file is at line 13, column 34, and is using UTF-8 encoding with CRLF line endings.

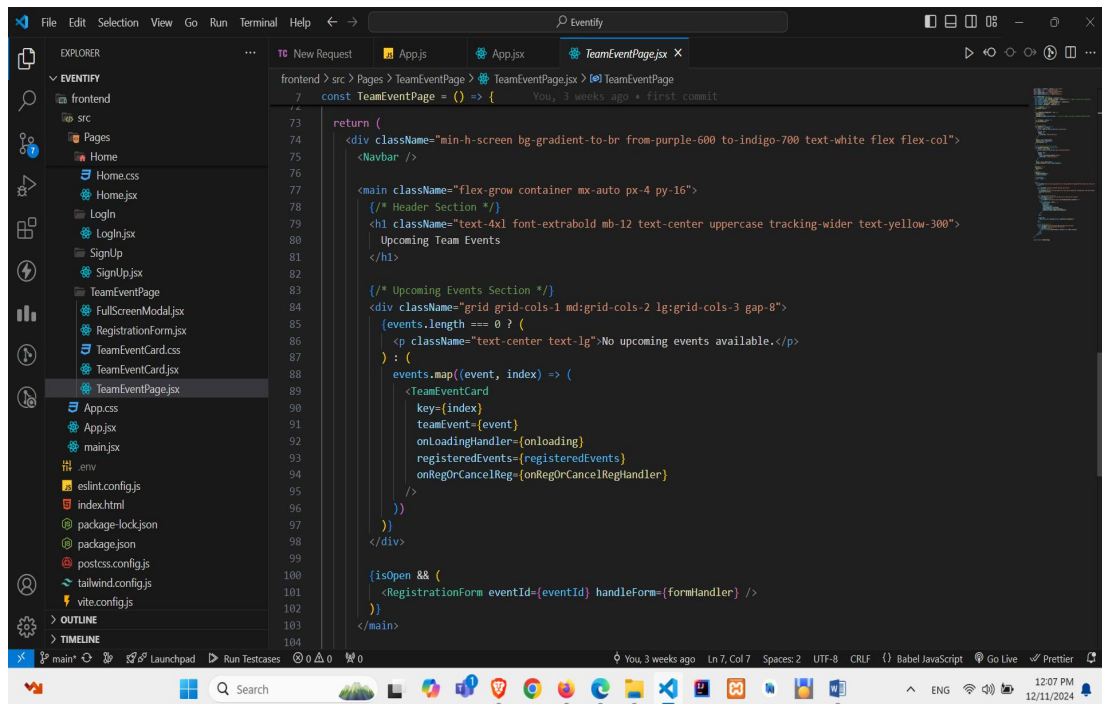
```
1 import Contact from "../Component/Contact/Contact";
2
3 import TeamEventPage from "../Pages/TeamEventPage/TeamEventPage";
4
5 function App() {
6   const navigate = useNavigate();
7
8   useEffect(() => {
9     if (!localStorage.getItem("token")) {
10       console.log("hello world");
11       navigate("/login");
12     } else {
13       console.log("bye");
14     }
15   }, []);
16
17   return (
18     <>
19     <Routes>
20       <Route path="/" element={<Home />}></Route>
21       <Route path="/contact" element={<Contact />}></Route>
22       <Route path="/signup" element={<Signup />}></Route>
23       <Route path="/login" element={<Login />}></Route>
24       <Route path="/events" element={<EventPage />}></Route>
25       <Route path="/team-events" element={<TeamEventPage />}></Route>
26       <Route path="/create-event" element={<CreateEvent />}></Route>
27     </Routes>
28   );
29 }
30
31 export default App;
```

```
15 const Home = () => {
  return (
    <div className="min-h-screen bg-gray-50">
      <Navbar />
      <div className="relative w-full max-w-10xl mx-auto">
        <div className="relative h-96 overflow-hidden rounded-xl">
          {slides.map((slide, index) => (
            <div
              key={index}
              className={`absolute w-full h-full transition-opacity duration-500 ${
                index === currentSlide ? "opacity-100" : "opacity-0"
              }`}
            >
              <img
                src={slide}
                alt={`Slide ${index + 1}`}
                className="w-full h-full object-cover"
              />
            </div>
          ))}
          <button
            onClick={prevSlide}
            className="absolute left-4 top-1/2 -translate-y-1/2 bg-black/50 p-2 rounded-full text-white hover:bg-black"
          >
            <ChevronLeft className="w-6 h-6" />
          </button>
          <button
            onClick={nextSlide}
            className="absolute right-4 top-1/2 -translate-y-1/2 bg-black/50 p-2 rounded-full text-white hover:bg-black"
          >
            <ChevronRight className="w-6 h-6" />
          </button>
        </div>
      </div>
    </div>
  )
}
```

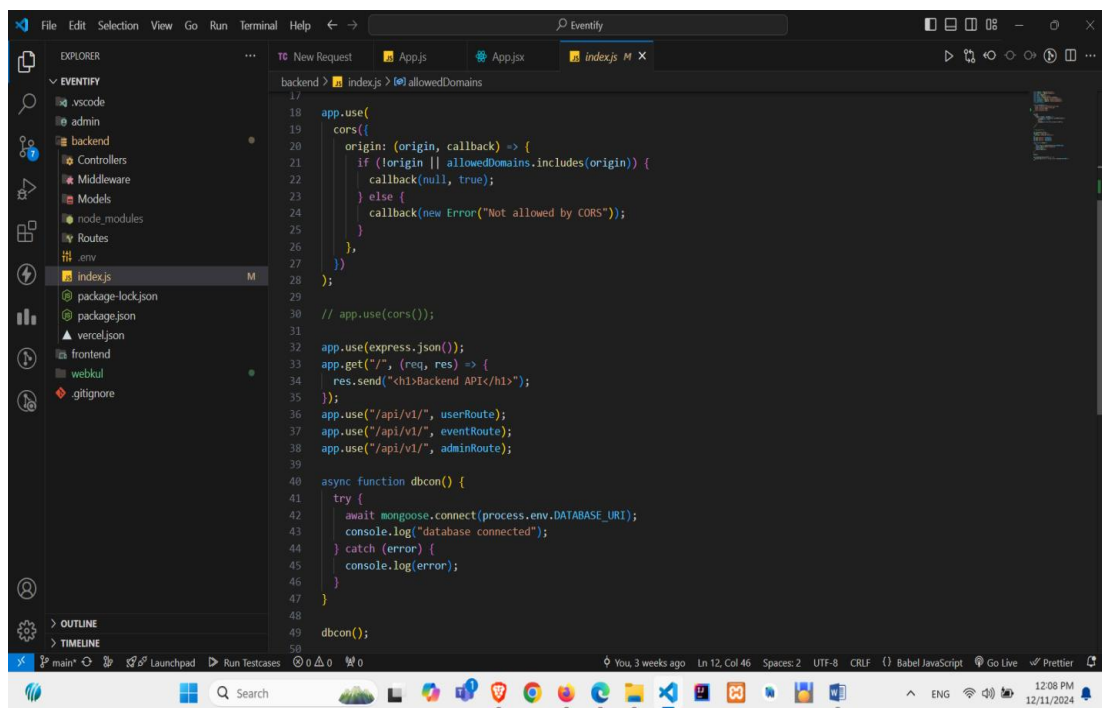
```
6 const SignUp = () => {
  return (
    <div className="flex justify-center items-center min-h-screen bg-gray-100">
      <div className="w-full max-w-md p-8 space-y-6 bg-white shadow-2xl rounded-xl">
        <h2 className="text-3xl font-bold text-center text-[#9333ea]">
          Create an Account
        </h2>
        <form onSubmit={handleSubmit} className="space-y-5">
          <TextField
            id="outlined-required"
            label="First Name"
            placeholder="Enter the first name"
            className="w-full"
            onChange={(e) => setFirstName(e.target.value)}
          />
          <TextField
            id="outlined-required"
            label="Last Name"
            placeholder="Enter the last name"
            className="w-full"
            onChange={(e) => setLastName(e.target.value)}
          />
        </form>
      </div>
    </div>
  )
}
```

```
frontent > src > Pages > Login > Login.jsx > Login > *8.MuiOutlinedInput-root*  
5 const Login = () => {  
35  
36   return (  
37     <div className="flex justify-center items-center min-h-screen bg-gray-100">  
38       <div className="w-full max-w-md p-8 space-y-6 bg-white shadow-2xl rounded-xl">  
39         <h2 className="text-3xl font-bold text-center text-[#9333ea]">  
40           Welcome Back!  
41         </h2>  
42         <form onSubmit={loginHandler} className="space-y-5">  
43           <TextField  
44             id="outlined-required"  
45             label="Email"  
46             placeholder="Enter E-mail"  
47             className="w-full"  
48             sx={{  
49               "&.MuiOutlinedInput-root": {  
50                 "&.Mui-focused fieldset": {  
51                   borderColor: "#9333ea",  
52                 },  
53               },  
54               "&.MuiInputLabel-root.Mui-focused": {  
55                 color: "#9333ea",  
56               },  
57             }}  
58           </TextField>  
59           <TextField  
60             id="outlined-password-input"  
61             label="Password"  
62             type="password"  
63             autoComplete="current-password"  
64             placeholder="Enter Password"  
65             className="w-full"  
66             sx={{
```

```
frontent > src > Pages > Events > EventPage.jsx > EventPage  
7 const EventPage = () => {  
73   return (  
74     <div className="min-h-screen bg-gradient-to-br from-purple-600 to-indigo-700 text-white">  
75       <Navbar />  
76  
77       <main className="container mx-auto px-4 py-16">  
78         <div>  
79           <h1 className="text-4xl font-extrabold mb-12 text-center uppercase tracking-wider text-yellow-300">  
80             Upcoming Events  
81           </h1>  
82  
83           <div>  
84             <div className="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-3 gap-8">  
85               {upcomingEvents.length === 0 ? (  
86                 <p className="text-center text-lg">No upcoming events available.</p>  
87               ) : (  
88                 upcomingEvents.map((event, index) => (  
89                   <EventCard  
90                     key={index}  
91                     event={event}  
92                     onLoadingHandler={onLoading}  
93                     registeredEvents={registeredEvents}  
94                     onRegOrCancelReg={onRegOrCancelRegHandler}  
95                     isPast={false}  
96                   </EventCard>  
97                 )  
98               )  
99             </div>  
100           </div>  
101  
102           <div>  
103             <h1 className="text-4xl font-extrabold mt-16 mb-12 text-center uppercase tracking-wider text-yellow-300">  
104               Past Events  
105             </h1>  
106           </div>  
107         </div>  
108       </div>  
109     </div>  
110   )  
111 }
```



## 8.5 BACKEND CODE



```
1 const mongoose = require("mongoose");
2
3 const eventSchema = new mongoose.Schema({
4   eventName: {
5     type: String,
6     required: true,
7   },
8   description: {
9     type: String,
10    required: true,
11  },
12  eventType: {
13    type: String,
14  },
15  date: {
16    type: Date,
17  },
18  endDate: {
19    type: Date,
20  },
21  startTime: {
22    type: String,
23  },
24  endTime: {
25    type: String,
26  },
27  location: {
28    type: String,
29  },
30  capacity: {
31    type: Number,
32  },
33 });
```

```
1 const mongoose = require("mongoose");
2
3 const teamEventSchema = new mongoose.Schema({
4   eventName: {
5     type: String,
6     required: true,
7   },
8   eventType: {
9     type: String,
10    required: true,
11  },
12  minTeamSize: {
13    type: Number,
14    required: true,
15    min: 1,
16  },
17  maxTeamSize: {
18    type: Number,
19    required: true,
20    min: 1,
21  },
22  description: {
23    type: String,
24    required: true,
25  },
26  rules: {
27    type: String,
28    required: true,
29  },
30  startDate: {
31    type: Date,
32    required: true,
33  },
34 });
```



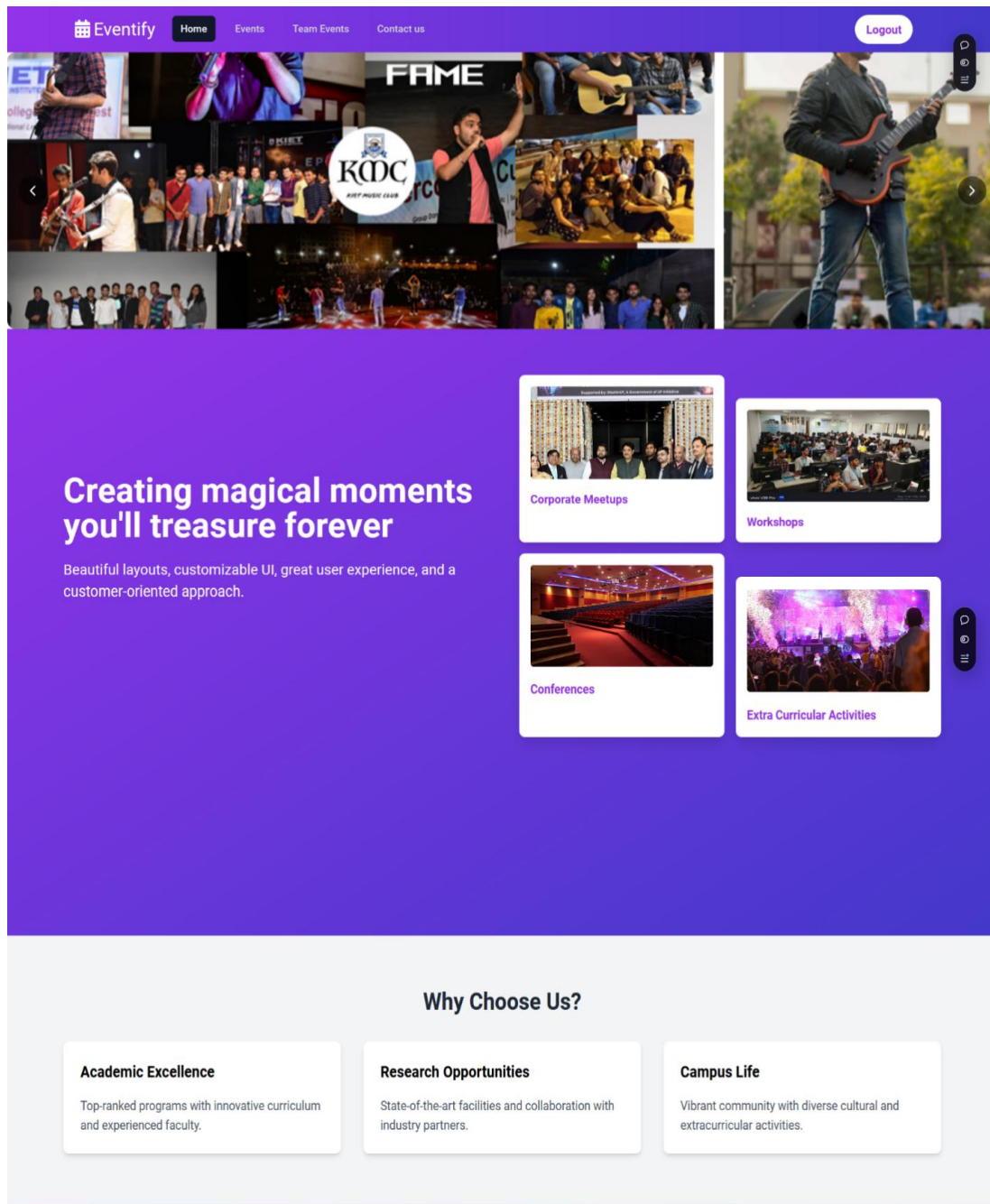
```
1 const express = require("express");
2
3 const {
4   signupController,
5   loginController,
6   getUserController,
7 } = require("../Controllers/userController");
8 const { isAuthenticated } = require("../Middleware/isAuthenticated");
9 const router = express.Router();
10
11 router.post("/admin-signup", signupController);
12 router.post("/admin-login", loginController);
13 router.get("/me", isAuthenticated, getUserController);
14
15 module.exports = router;
```

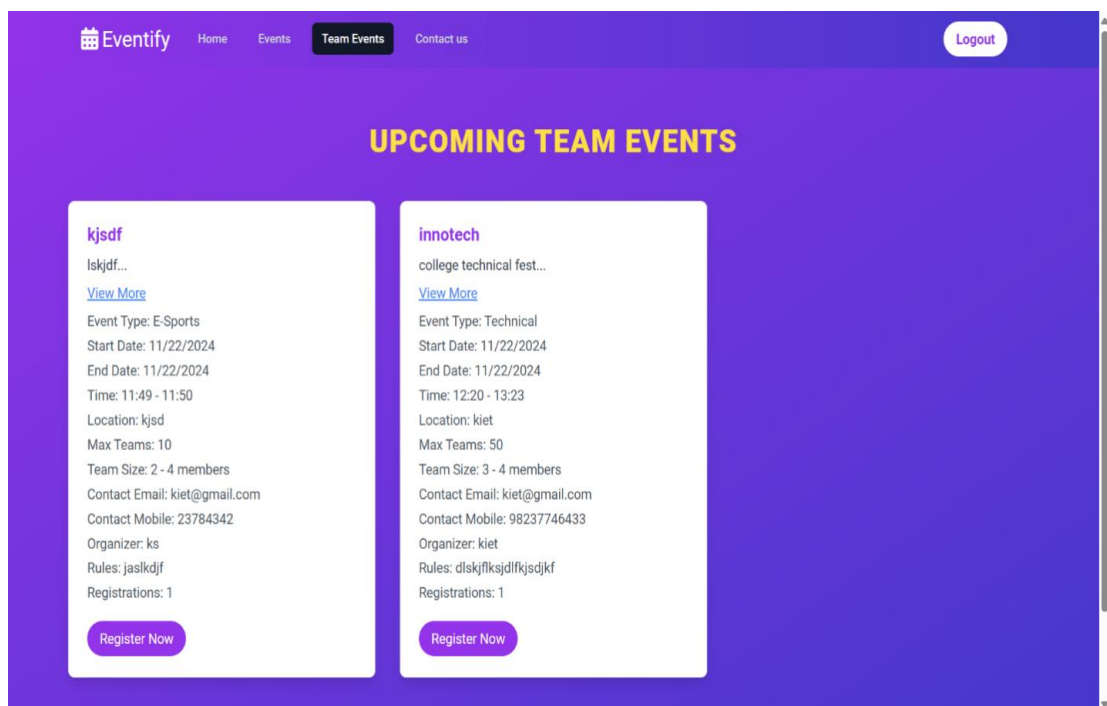
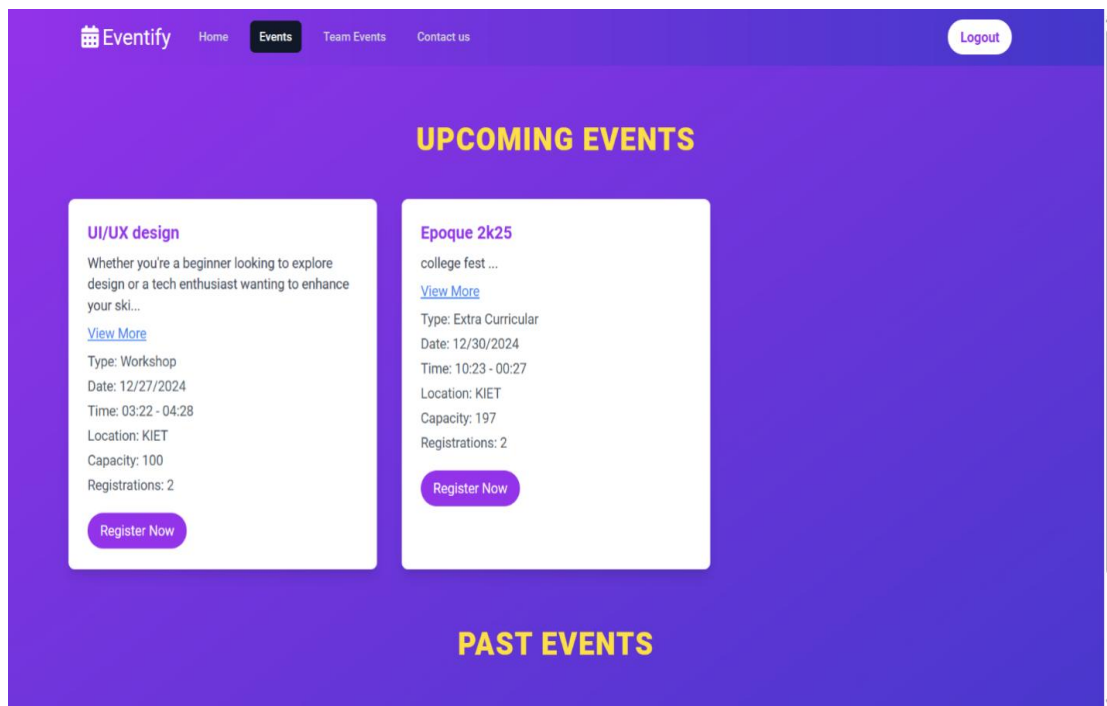
```
18   isAuthenticatedAdmin,
19 } = require("../Middleware/isAuthenticated");
20
21 const router = express.Router();
22
23 //user side event routes
24 router.get("/events", getAllEvents);
25 router.get("/team-events", getAllTeamEvents);
26 router.post("/register-event", isAuthenticated, registerEvent);
27 router.post("/register-team-event", isAuthenticated, registerTeamEvent);
28 router.get("/registered-events", isAuthenticated, alreadyRegisteredEvents);
29 router.post("/cancel-registration", isAuthenticated, cancelRegistration);
30
31 //admin side event routes
32 router.post("/create-event", isAuthenticatedAdmin, createEvent);
33 router.put("/events/:eventId", isAuthenticatedAdmin, updateEvent);
34 router.get("/events/:eventId", getEventById);
35 router.get("/registrations", isAuthenticatedAdmin, getUserRegistrations);
36 router.get("/team-registrations", isAuthenticatedAdmin, getTeamRegistrations);
37 router.post("/create-team-event", isAuthenticatedAdmin, createTeamEvent);
38
39 module.exports = router;
```

```
backend > Middleware > isAuthenticated.js > ...
Anuj Singh, 2 months ago | 1 author (Anuj Singh)
1 const jwt = require("jsonwebtoken");
2 const User = require("../Models/userModel");
3
4 const isAuthenticated = async (req, res, next) => {
5   try {
6     const token = req.headers["token"];
7     jwt.verify(token, process.env.MY_SECRET_KEY, async (err, user_id) => {
8       // console.log(user_id);
9       try {
10        const user = await User.findOne({ _id: user_id });
11        if (user) {
12          req.user = user;
13          next();
14        } else {
15          return res.status(400).json({
16            success: "false",
17            message: "Login first in order to access the resource",
18          });
19        }
20      } catch (error) {
21        return res.status(400).json({
22          success: "false",
23          message: "Login first in order to access the resource",
24        });
25        // console.log(error.message)
26      }
27    });
28  } catch (error) {
29    // console.log(error.message);
30    return res.status(400).json({
31      success: "false",
32      message: "Invalid User token not found or expired",
33    });
34  }
35}
```

```
backend > Controllers > eventController.js > ...
205 const getAllEvents = async (req, res) => {
206   try {
207     const page = parseInt(req.query.page) || 1;
208     const limit = parseInt(req.query.limit) || 9;
209     const skip = (page - 1) * limit;
210
211     const events = await Event.aggregate([
212       { $sort: { createdAt: -1 } }, // Sort by newest events
213       { $skip: skip },
214       { $limit: limit },
215       {
216         $lookup: {
217           from: "registrations", // Ensure correct collection name
218           localField: "_id", // Local field in Event
219           foreignField: "event_id", // Foreign field in Registration
220           as: "registrations", // Alias for joined data
221         },
222       },
223       {
224         $addFields: {
225           registrationsCount: { $size: "$registrations" }, // Count registrations
226         },
227       },
228     ]);
229
230     res.status(200).json({
231       success: true,
232       events,
233     });
234   } catch (error) {
235     console.error("Error in getAllEvents:", error.message);
236     res.status(500).json({
237       success: false,
238       message: "Failed to fetch events"
239     });
240   }
241 }
```







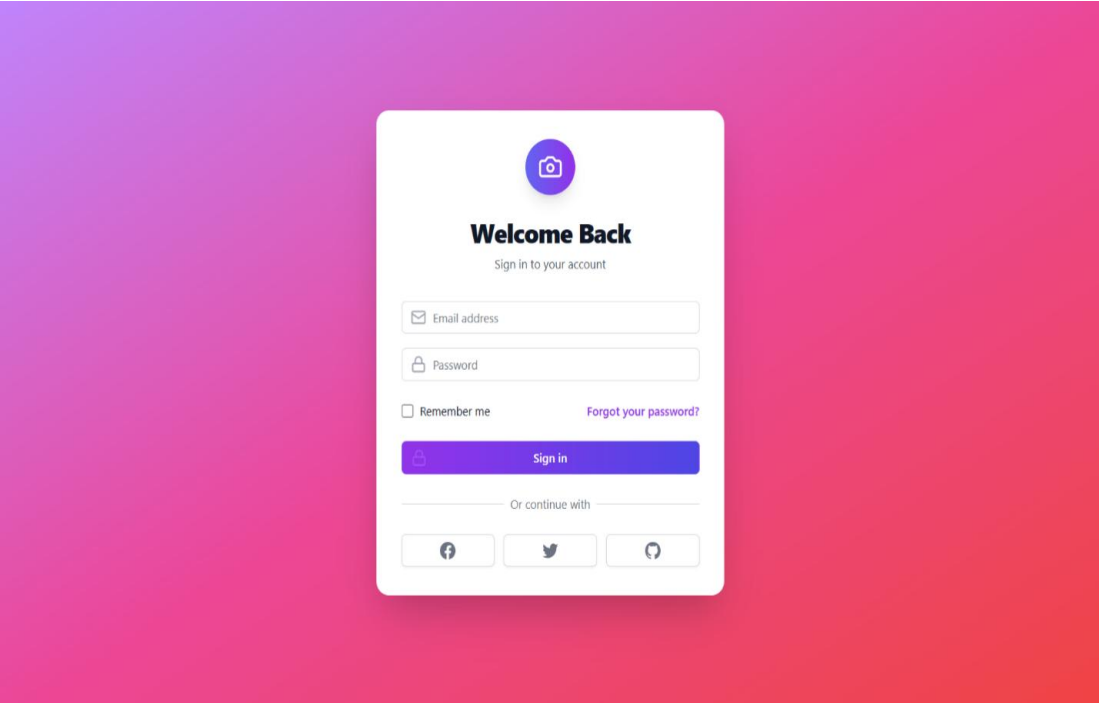
### Create an Account

Select Course

[Already have an account? Login](#)

### Welcome Back!

[Forgot Password?](#) [Don't have an account? Sign Up](#)



## CHAPTER 9

### REFERENCES

#### MYLOFT E-Resources

**1-HTML5, JavaScript, and jQuery 24-Hour Trainer-**

<https://ebooks.wileyindia.com/home/product-details/284541>

**2-HTML & CSS design and build websites-**

<https://ebooks.wileyindia.com/home/product-details/394331>

**3-Web technologies and application 2 Edition -**

<https://ebookstore.bspublications.net/reader/web-technologies-and-applications-1609052716?location=273>

**4-Web technologies , Black Book-**<https://ebooks.wileyindia.com/home/product-details/282807>

**5-HTML, CSS, and JavaScript All in One-**

[https://ebooks.elibrary.in.pearson.com/wr/viewer.html#book/86f76471-a963-4fc7-9aaf-d1bb18f699a4/title/pos\\_0](https://ebooks.elibrary.in.pearson.com/wr/viewer.html#book/86f76471-a963-4fc7-9aaf-d1bb18f699a4/title/pos_0)

**6-HTML 5 Black Book, Covers CSS 3, JavaScript, XML, XHTML, AJAX,**

**7-PHP and jQuery, 2ed-**<https://ebooks.wileyindia.com/home/product-details/283099>

**8-An Introduction to Database Systems-**

<https://ebooks.elibrary.in.pearson.com/wr/pdfviewer.html#book/4a67a382-df9b-4e9d-8ac6-cdf779137071>

**9-Concept of database management system-**

<https://ebooks.elibrary.in.pearson.com/wr/pdfviewer.html#book/a4b5c462-ca51-45db-ab12-e1203fea660d>

**10-Journal of Database system-**

[https://research.ebsco.com/c/fwkguc/search/advanced/publications/0L4?selectedDb=crh\\_jnh&db=crh](https://research.ebsco.com/c/fwkguc/search/advanced/publications/0L4?selectedDb=crh_jnh&db=crh)

**MongoDB Documentation:** <https://www.mongodb.com/docs/>

**Express.js Documentation:** <https://expressjs.com/>

**React Documentation:** <https://reactjs.org/docs/getting-started.html>

**Node.js Documentation:** <https://nodejs.org/en/docs/>

**Vercel Documentation:** <https://vercel.com/docs>

**Mongoose Documentation:** <https://mongoosejs.com/docs/>

**JWT Documentation:** <https://jwt.io/>

**MDN Web Docs:** <https://developer.mozilla.org/>

**W3Schools:** <https://www.w3schools.com/>

**GeeksforGeeksMERN Tutorials:** <https://www.geeksforgeeks.org/mern-stack>

**Lucidchart:** <https://www.lucidchart.com>

**Postman:** <https://www.postman.com>

**Vercel Documentation:** <https://vercel.com/docs>