PLAN EVENTS - EVENT MANAGEMENT SYSTEM

A PERSONAL PROJECT REPORT

Developed and Presented by

R. LAZARUS ROLANDO

Project Description

This project report titled “Event Management System” represents an independent software development initiative undertaken by R. Lazarus Rolando.
The system is designed to simplify and automate the planning, organization, and management of events using modern web technologies.

Developer Information

Name: R. Lazarus Rolando

Project Title: Event Management System

Date: October, 2025

CERTIFICATE

This is to certify that the project work entitled **“Event Management System”** is an original project personally developed and completed by **R. Lazarus Rolando**.

The project demonstrates independent research, design, and implementation in the field of web-based event planning and management systems.

This certificate acknowledges the completion of the project as part of the developer’s individual initiative to enhance technical proficiency and practical experience in full-stack application development.

Name & Signature of the Developer

R Lazarus Rolando

Project Completion Date

October 2025

DECLARATION

I, R. Lazarus Rolando, hereby declare that the project report entitled “Event Management System” has been personally developed and completed by me.

This project is the result of my own work, research, and implementation, carried out as part of my personal initiative to enhance my technical knowledge and software development skills.

I further declare that this project is original and has not been submitted, in part or full, for any academic qualification or professional certification elsewhere.

Date	Signature of Candidate
25/10/25	[illegible]

ACKNOWLEDGEMENT

First and foremost, I thank **God Almighty** for His abundant blessings, guidance, and strength throughout the development of this project.

I am deeply grateful to my **family** for their constant encouragement, valuable suggestions, and unwavering support, which motivated me to complete this project successfully.

I would also like to express my heartfelt appreciation to all those who have directly or indirectly contributed to the successful completion of my project titled **“Event Management System.”**

This project has been a great learning experience, and I sincerely acknowledge the support, feedback, and inspiration I received from various individuals and online resources that helped me refine my technical and problem-solving skills.

Finally, I take this opportunity to thank everyone who believed in my vision and encouraged me to pursue and complete this project with dedication and passion.

INDEX

ABSTRACT

PLAN EVENTS - EVENT MANAGEMENT SYSTEM

In today's fast-paced digital era, organizing and managing events efficiently has become a major challenge for individuals and organizations alike. The **Event Management System (EMS)** is a comprehensive web-based platform designed to simplify and automate the entire process of event planning, coordination, and execution.

The system enables users to **create, manage, and monitor events** such as conferences, seminars, workshops, weddings, and parties — all from a single, user-friendly interface. Through EMS, users can register events, schedule dates, allocate venues, manage participants, and track updates in real time.

A key feature of EMS is its **intuitive event registration and booking module**, allowing users to easily book events and receive instant confirmation through the system. Additionally, the platform provides **real-time dashboards** for administrators to manage event details, track attendee counts, and handle logistics efficiently.

The system also integrates **secure login authentication**, **automated notifications**, and **database-driven management** to ensure reliability and accuracy in every stage of event handling. By leveraging modern web technologies, EMS reduces manual effort, enhances collaboration, and ensures smooth coordination between organizers and participants.

Overall, the **Event Management System** offers a practical, efficient, and scalable solution for event organization — transforming traditional, time-consuming processes into a **streamlined, technology-driven experience** for users and administrators alike.

CHAPTER - 1

INTRODUCTION

The **Event Management System (EMS)** is an innovative web-based platform developed to simplify the complex process of planning, organizing, and executing events. In today's fast-moving world, event organizers face challenges in managing registrations, coordinating schedules, and maintaining clear communication with participants. EMS addresses these challenges by providing an integrated solution that combines automation, accessibility, and real-time management.

Through the system, users can easily **create events, register participants, assign venues, and monitor progress** — all within a unified digital environment. The platform features a **dynamic dashboard** that allows administrators to view event details, track attendance, and manage tasks efficiently. Participants can register online, receive instant confirmations, and use **QR code-based digital passes** for seamless entry at the venue.

Security and reliability are central to the system's design. EMS implements **secure authentication**, **database encryption**, and **audit tracking** to ensure data integrity and prevent unauthorized access. The system's **responsive interface** allows users to manage events from desktops, tablets, or mobile devices with equal ease.

By combining simplicity with advanced functionality, the **Event Management System** enhances the overall event experience for both organizers and attendees. It minimizes manual effort, improves coordination, and brings structure and transparency to event operations — making event management smarter, faster, and more efficient in the digital age.

- **Comprehensive Event Coordination:**
EMS simplifies the entire event process by integrating event creation, scheduling, participant management, and communication within a single, unified platform.

- **Smart Booking and Notifications:**
The system allows users to register for events online and receive instant confirmations, reminders, and updates through automated notifications, ensuring smooth coordination.

- **Enhanced Security and Data Protection:**
EMS ensures secure user access with login authentication and encrypted data storage, protecting sensitive event and participant information.

- **Efficient Administration and Reporting:**
Event organizers can monitor attendance, manage resources, and generate analytical reports in real time, enabling better decision-making and improved event outcomes.

1.1 OBJECTIVES

1. **Facilitating Inclusive Participation**
Ensure that all users, including individuals with disabilities, can seamlessly access and participate in events. Features such as multi-language support, screen reader compatibility, and voice-assisted navigation make the system inclusive and easy to use for everyone.

2. **Ensuring Security and Data Integrity**
Protect sensitive data and user information through role-based authentication, QR code–based entry verification, and encrypted storage. These measures prevent unauthorized access and provide a safe environment for both organizers and participants.

3. **Enhancing Accessibility and Convenience**
Provide a smooth and user-friendly experience with real-time event updates, automated notifications, and online registration. The intuitive interface allows participants and organizers to efficiently manage events from any device.

4. **Fostering Transparency and Accountability**
Maintain detailed audit trails and activity logs to track event registrations, payments, and attendance. This ensures clear and accountable operations for both administrators and users.

5. **Upholding Legal and Ethical Standards**
Comply with data protection regulations, digital privacy laws, and secure transaction protocols to create a trusted and legally compliant platform for event management.

6. **Promoting Continuous Improvement and Innovation**
Integrate modern technologies such as automated scheduling, AI-driven analytics, and cloud-based data management to enhance system efficiency, scalability, and overall user satisfaction.

Important Features of the Event Management System (EMS)

1. **Event Planning and Management**
Create, schedule, and organize events efficiently with all essential details.

2. **Online Registration and Secure Access**
Participants can register online and use QR codes for seamless, secure entry.

3. **Real-Time Updates and Reporting**
Receive automated notifications, track attendance, and generate reports for smooth event management.

CHAPTER - 2

SYSTEM SPECIFICATIONS

A **Software Requirements Specification (SRS)** is a formal document that defines the requirements of a software system to be developed. For the **Event Management System (EMS)**, the SRS describes both **functional and non-functional requirements** and outlines the interactions between users (organizers and participants) and the system.

The SRS establishes a **common understanding** between stakeholders—including developers, project managers, and end-users—on what the EMS is expected to do and what it is not intended to do. It provides a structured approach to **analyze requirements before the design phase**, reducing errors, minimizing costly redesigns, and ensuring the system meets its objectives.

The **Software Requirements Specification (SRS)** serves as a **communication bridge** between stakeholders—including event organizers, participants, and developers—and software designers. The specific goals of the EMS SRS are:

1. **Facilitating Reviews:**
   Enable stakeholders to review and validate system requirements at various stages of development.

2. **Defining Scope of Work:**
   Clearly outline the functionality and boundaries of the EMS, specifying what the system will and will not do.

3. **Providing a Reference for Designers:**
   Serve as a guide for developers and designers with structured documentation and navigation aids.

4. **Framework for Testing Use Cases:**
   Offer a basis for testing primary and secondary scenarios, ensuring that all user interactions are accounted for.

5. **Capturing Customer Requirements:**
   Include all essential features requested by stakeholders, ensuring the system meets end-user needs.

6. **Supporting Ongoing Refinement:** Allow for iterative updates, clarifications, and additions during the software development life cycle.

2.1 SOFTWARE REQUIREMENTS

Software requirements define the functional and non-functional characteristics that a software system must exhibit to achieve its intended objectives and satisfy stakeholder expectations. **Software requirements theory** encompasses the principles, methodologies, and practices used to systematically elicit, analyze, specify, validate, and manage these requirements throughout the entire software development lifecycle.

The **primary goal of software requirements theory** is to ensure that software systems are designed and developed to meet the needs, expectations, and objectives of users, stakeholders, and business entities. The theory encompasses a range of techniques and methodologies for gathering, analyzing, and managing requirements, including **stakeholder analysis**, **use case modeling**, **elicitation of functional and non-functional requirements**, **requirements validation**, and **requirements management** throughout the development lifecycle.

Operating System	Windows 10
Language	HTML, CSS, NodeJS, ReactJS, Python and SQLite3
Tool	Visual Studio Code

2.2 HARDWARE REQUIREMENTS

Hardware requirements specify the technical specifications that a computer or electronic device must satisfy to execute a particular task or run specific software efficiently. These requirements are influenced by several factors, including the **complexity of the software**, the **volume of data processed**, and the **desired performance level** of the system.

Generally, hardware requirements are categorized into **minimum requirements**—the essential specifications needed for the system to operate—and **recommended requirements**, which ensure optimal performance and user experience.

CPU Type	Intel i5 7th gen
RAM Size	8GB
Storage Size	500GB

CHAPTER - 3

SURVEY OF TECHNOLOGIES

3.1 FEATURES OF FRONTEND

The **frontend** of the *Event Management System* provides an interactive and user-friendly interface that allows users to browse, register for, and manage events seamlessly. It is developed using **HTML**, **CSS**, and **React.js**, ensuring a modern, responsive, and efficient user experience.

Frontend link: http://localhost:3000/ (Web App)

Key Features:

1. **User-Friendly Interface**

- Designed with **HTML** and **CSS** to provide a clean, intuitive layout that makes navigation easy for users.

- Allows users to view upcoming events, event details, and registration options in a well-structured format.

2. **Responsive Design**

- Ensures compatibility across devices (desktops, tablets, and smartphones) through **responsive web design** principles.

- Utilizes **CSS flexbox**, **grid systems**, and **media queries** to adapt to different screen sizes smoothly.

3. **Dynamic and Interactive Components**

- Built using **React.js**, enabling real-time updates without page reloads.

- Features include live event updates, participant registration, and interactive forms for feedback or queries.

4. **Real-Time Data Integration**

- The frontend communicates with the **Node.js** and **Python** backend through **RESTful APIs** for live event data, registrations, and user account management.

- Data is fetched and displayed dynamically from the **SQLite3** database to ensure accuracy and timeliness.

5. **Performance Optimization**

- Utilizes **React's virtual DOM**, **lazy loading**, and **code optimization** techniques to enhance speed and responsiveness.

- Reduces load times and improves overall user experience even during peak event registrations.

6. **Security and Validation**

- Implements **client-side form validation** to prevent invalid input and reduce server load.

- Ensures secure communication between frontend and backend components.

7. **Modern Design and User Experience (UX)**

- Follows modern UI/UX design standards, ensuring simplicity, consistency, and accessibility.

- Incorporates animations and transitions for smooth interactions.

3.2 FEATURES OF BACKEND

In software development, **backend** refers to the server-side logic of an application—the part that handles data processing, database management, and communication between users and servers. Python is one of the most popular languages for backend development due to its readability, versatility, and vast ecosystem of frameworks and libraries.

Python's backend capabilities allow developers to build scalable, maintainable, and efficient web applications, APIs, and microservices. Some of the most popular Python frameworks for backend development include **Flask**, **Django**, and **FastAPI**. Among these, **Flask** is widely known for its simplicity and flexibility.

The backend is responsible for managing data, processing requests, and providing responses tousers. It's also responsible for communicating with the frontend, which is the user-facing part of the application. In general, the backend can be divided into three main components:

Server: This is the hardware or software that provides the resources needed to run an application, such as CPU, RAM, and storage. The server is responsible for processing requests and returning responses.

Application: This is the software that runs on the server and is responsible for processing requests
and returning responses. It may use programming languages such as Java, Python, Ruby, or Node.js.

Backend Link: http://localhost:5000 (Python)

Backend Port: 3001 (Node.js)

Key Features:

1. **Simple and Readable**
- Python's clean and intuitive syntax makes it easy to read, write, and maintain, even for beginners.

2. **Interpreted and Dynamically Typed**
- Python executes code line-by-line without compilation, and variable types are determined automatically at runtime.

3. **Object-Oriented and Functional**
- Supports both object-oriented (classes, inheritance) and functional programming (lambda, map, filter) paradigms.

4. **Extensive Standard Library**
- Comes with built-in modules for file handling, networking, regular expressions, databases, and more.

5. **Cross-Platform Compatibility**
- Runs seamlessly on Windows, macOS, and Linux without changing the source code.

6. **Large Ecosystem and Community Support**
- Offers thousands of third-party libraries (Flask, Django, NumPy, etc.) and a strong developer community.

7. **Automatic Memory Management**
- Includes a built-in garbage collector that efficiently handles memory allocation and deallocation.

3.3 FEATURES OF DATABASE

A **database** is an organized collection of structured information, or data, that is stored and managed electronically. It serves as the **core component of backend systems**, allowing applications to store, retrieve, and manipulate data efficiently. Databases are designed to handle large volumes of information while maintaining accuracy, consistency, and accessibility. They are managed by specialized software called a **Database Management System (DBMS)**, which provides tools for data storage, querying, and security. Common examples include **MySQL**, **PostgreSQL**, **SQLite**, and **MongoDB**.

In a web application, the **database** stores essential data such as user details, product information, and transaction records. When a user interacts with the frontend—like submitting a form or logging in—the backend connects to the database to insert, update, or retrieve data, then returns the result. Databases can be **relational (SQL)**, storing data in tables with defined relationships, or **non-relational (NoSQL)**, managing unstructured data in collections or documents.

Key Features:

1. **Data Storage and Organization**
   Stores data in a structured and organized way for easy access and management. It ensures that information is systematically arranged for quick retrieval and updates.

2. **Data Integrity and Consistency**
   Ensures accuracy, reliability, and correctness of data across all transactions. This prevents duplication and maintains the validity of information.

3. **Data Security**
   Protects data through authentication, authorization, and encryption mechanisms. Only authorized users can access or modify sensitive data.

4. **Concurrent Access**
   Allows multiple users to access and modify data simultaneously without conflicts. This ensures smooth operation even in multi-user environments.

5. **Backup and Recovery**
   Provides mechanisms to restore data in case of failures or data loss. Regular backups protect against accidental deletions or hardware issues.

6. **Efficient Query Processing**
   Supports powerful query languages (like SQL) to retrieve and manipulate data quickly. This helps optimize system performance and response time.

7. **Scalability and Performance**
   Handles increasing amounts of data and user requests efficiently. Databases are designed to scale horizontally or vertically as system demands grow.

CHAPTER - 4

DEVELOPMENT PROCESS

4.1 REQUIREMENTS ANALYSIS

Requirement analysis is a crucial step in the software development process, where the software requirements are collected, documented, and analyzed to ensure that the final product meets the stakeholders' expectations. The goal of requirement analysis is to identify and document the functional and non-functional requirements of the software system to be developed.

The general theory behind requirement analysis involves the following steps:

1. **Gathering requirements:**
   The first step in requirement analysis is to gather all the requirements from various stakeholders. The stakeholders may include the customers, end- users, domain experts, business analysts, and other members of the development team. The requirements can be gathered through various techniques, such as interviews, surveys, workshops, and observation.

2. **Analyzing requirements:**
   After gathering the requirements, the next step is to analyze them.The analysis involves examining the requirements to ensure that they are complete, consistent, and feasible. The requirements are also analyzed to determine the impact they may have on the software system's architecture, design, and development.

3. **Prioritizing requirements:**
   Once the requirements are analyzed, the next step is to prioritize them. Prioritization involves ranking the requirements based on their importance and urgency. The prioritization process helps in identifying the critical requirements that need to be addressed first.

4. **Documenting requirements:**
   The requirements are then documented in a clear and concise manner. The documentation should include all the relevant information about the requirements, such as their purpose, scope, and acceptance criteria. The documentation serves as a reference for the development team throughout the software development process.

5. **Validating requirements:**
   The final step in requirement analysis is to validate the requirements. Validation involves reviewing the requirements with the stakeholders to ensure that they accurately reflect their needs and expectations. The validation process helps in identifying any discrepancies or gaps in the requirements and making the necessary adjustments.

4.2 PROGRAMMING LANGUAGE

HTML

HTML (Hypertext Markup Language) is a markup language used to create web pages and other online content. It is a standardized system of codes used to create documents that can be viewed on the internet in a web browser. HTML is based on the concept of tags, which are used to define the structure and content of a document. Tags are surrounded by angle brackets, and usually come in pairs: an opening tag and a closing tag. The content of a document is placed between the opening and closing tags.

1. **Definition:**
   "Hypertext" refers to text that contains links to other text (the foundation of the web), and "Markup" refers to the tags used to structure the text.

2. **Primary Use:**
   HTML provides the skeleton or structure for all websites, defining where the headings, paragraphs, images, and links are placed. It is the fundamental building block that other technologies, like CSS (style) and JavaScript (interactivity), rely on.

3. **Standard:**
   The standard is maintained by the World Wide Web Consortium (W3C) to ensure that all web browsers and devices interpret and display web pages consistently. The current major version is HTML5, which introduced new features like <video>, <audio>, and semantic elements.

4. **Viewing:**
   It creates documents that can be viewed in a web browser.

5. **Core Concept:**
   HTML is based on the concept of **tags**.

6. **Tag Structure:**
   Tags are enclosed in angle brackets (< >), usually come in pairs (opening and closing), and content is placed between them.

CSS

CSS (Cascading Style Sheets) is a language used for describing the presentation of web pages. It is used to style and layout HTML and XML documents, as well as other markup languages.

1. **Selector:**
   A selector is used to target HTML elements on a web page. It can be an element name, class, ID, attribute, or a combination of these.

2. **Cascading:**
CSS uses cascading to determine which styles should be applied to an element. This means that styles can be inherited from parent elements, or overridden by more specific styles.

3. **Specificity:**
Specificity is a measure of how specific a selector is, and determines which styles take precedence when multiple styles are applied to an element. Generally, the more specific the selector, the higher the precedence.

4. **Box model:**
The box model is a concept used in CSS to describe how elements are laid out on a web page. Each element is treated as a rectangular box with content, padding, borders, and margins.

5. **Layout:**
CSS can be used to control the layout of elements on a web page. This includes controlling the position, size, and spacing of elements.

6. **Responsive design:**
CSS can be used to create responsive designs that adjust to different screen sizes and devices. This includes using media queries to target specific screen sizes, and using flexible units like percentages and viewport units.

7. Animations and transitions:
CSS can be used to create animations and transitions on a web page. This includes using key frames to define animations, and using transition properties to create smooth transitions between states.

NODEJS

Node.js is an open-source, back-end JavaScript runtime environment. It allows developers to run JavaScript code on the server, outside of a web browser. It is built on Chrome's V8 JavaScript engine and is designed for building scalable, fast network applications. It is particularly well-suited for applications that are data-intensive and require real-time communication.

1. **Event-Driven, Non-Blocking I/O:**
This is the core model of Node.js. Instead of waiting for a task (like reading a file or a database query) to finish, Node.js registers a callback and moves on to the next task. When the original task completes, the callback is executed. This allows it to handle thousands of simultaneous connections efficiently with a single thread.

2. **NPM (Node Package Manager):**
NPM is the default package manager for Node.js and the world's largest ecosystem of open-source libraries. It allows developers to easily install, share, and manage reusable code packages (or "modules") for their projects.

3. **V8 Engine:**
   Node.js runs on the same V8 JavaScript engine that powers Google Chrome. V8 compiles JavaScript directly into native machine code, which makes Node.js applications very fast and performance.

4. **Modules:**
   Node.js uses a module system (both CommonJS `require` and ES Modules `import`) that allows you to organize your code into separate, reusable files. This helps in keeping large applications structured and maintainable.

5. **Single-Threaded Event Loop:**
   While Node.js itself runs in a single thread, its event loop efficiently manages asynchronous operations. This model simplifies programming by avoiding the complexities of multi-threading, while still achieving high concurrency.

6. **Full-Stack JavaScript:**
   Node.js enables developers to use JavaScript for both the front-end (in the browser) and the back-end (on the server), creating a unified development experience and allowing code to be shared between client and server.

REACTJS

React.js (or simply React) is a free and open-source front-end JavaScript **library** for building user interfaces (UIs) based on components. It is maintained by Meta (formerly Facebook) and a community of developers. React is often used to build "Single Page Applications" (SPAs) and is known for its high performance and declarative approach to UI design.

1. **Component-Based Architecture:**
   React builds UIs from small, isolated, and reusable pieces of code called **components**. A complex UI can be broken down into many simple components (like a button, a search bar, or a user profile), which are then composed together to build the entire application.

2. **JSX (JavaScript XML):**
   JSX is a syntax extension for JavaScript that allows developers to write HTML-like code directly within their JavaScript files. This makes it easier to visualize and create the UI structure within the component logic.

3. **Virtual DOM (Document Object Model):**
   React uses a "virtual" copy of the browser's DOM. When a component's data changes, React first updates this lightweight virtual DOM. It then compares the virtual DOM with the real DOM and calculates the most efficient way to apply only the necessary changes, minimizing performance-heavy direct manipulations of the browser DOM.

4. **State and Props:**
   Components manage their own data using **state**. When the `state` of a component changes, React re-renders the component. Data can also be passed from a

parent component to a child component via **props** (properties), which are read-only.

5. **One-Way Data Flow:**
   In React, data flows in a single direction (down the component tree, from parent to child via props). This makes the application's logic more predictable, easier to understand, and less prone to bugs.

6. **Hooks:**
   Hooks (like `useState`, `useEffect`) are functions introduced in modern React that let you "hook into" React state and lifecycle features from function components. This allows for wr   iting powerful components without needing to use the more complex "class" syntax.

PYTHON

Python is a **high-level, interpreted, general-purpose programming language** known for its **simplicity, readability, and flexibility**. Created by **Guido van Rossum** and first released in 1991, Python emphasizes code clarity using significant indentation instead of braces, making it easy to learn and maintain.

Python supports multiple programming paradigms — **procedural, object-oriented, and functional programming** — and has an extensive standard library, which allows developers to perform complex tasks with minimal code. It is widely used in **web development, data science, artificial intelligence, automation, scientific computing, and cybersecurity**.

Its cross-platform compatibility and vast ecosystem of third-party libraries (such as NumPy, Pandas, Django, and TensorFlow) make Python one of the most versatile and powerful programming languages today.

1. **Syntax and Readability:**
   Python uses indentation to define code blocks, emphasizing readability and reducing syntactic clutter.

2. **Data Types and Structures:**
   Supports fundamental types like integers, floats, strings, and complex collections such as lists, tuples, sets, and dictionaries.

3. **Control Structures:**
   Provides robust constructs for decision-making and looping (`if`, `for`, `while`, etc.) and error handling (`try-except` blocks).

4. **Functions and Modules:**
   Encourages modular programming with functions and modules, promoting code reuse and organization.
   Example: `import math` or `from module import function`.

5. **Object-Oriented Programming (OOP):**
Allows defining classes and objects with inheritance, encapsulation, and polymorphism for building scalable systems.

6. **Libraries and Frameworks:**

- **Web Development:** Django, Flask, FastAPI

- **Data Science & AI:** NumPy, Pandas, Matplotlib, Scikit-learn, TensorFlow

- **Automation:** Selenium, PyAutoGUI

7. **Virtual Environments:**
Tools like `venv` and `pipenv` isolate dependencies for different projects to prevent package conflicts.

8. **Cross-Platform and Interpreted:**
Python code runs without compilation and works seamlessly on Windows, macOS, and Linux.

9. **Use Cases:**

- Web and backend development

- Data analysis and machine learning

- Scripting and automation

- Cybersecurity and system administration

- Game and GUI development

CHAPTER - 5

SYSTEM ANALYSIS

System Analysis is the process of **studying and understanding the existing system**, identifying its **problems, requirements, and objectives**, and determining how a new or improved system can meet those needs effectively. It involves examining the flow of data, the processes involved, the interactions between components, and the performance of the system to design a better solution.

In the context of the **Event Management System**, system analysis focuses on identifying user needs (such as event registration, management, and scheduling), understanding existing manual processes, and defining the functionalities that the new system should provide to improve efficiency, accuracy, and user experience.

System analysis also plays a vital role in bridging the gap between the **problem domain** and the **solution domain**. It helps analysts translate real-world business processes into logical system requirements by identifying the inputs, outputs, and transformations required to achieve organizational goals. Through systematic investigation and documentation, system analysis ensures that every functional aspect—such as data handling, user access, and security—is clearly understood before design begins. This minimizes misunderstandings between stakeholders and developers, leading to a more accurate and efficient system design.

Moreover, an effective system analysis provides a clear vision for future scalability and integration. By carefully examining workflows and dependencies, analysts can anticipate potential growth in data volume, user load, or feature expansion. For the Event Management System, this means ensuring that the platform can handle multiple concurrent events, large user bases, and advanced functionalities like payment integration or automated notifications. A thorough system analysis not only defines what the system should achieve today but also prepares it to adapt to future technological and operational needs.

IMPORTANCE OF SYSTEM ANALYSIS

1. **Foundation for System Development:**
   System analysis forms the basis for designing and developing a new system. It helps in understanding what the system must accomplish, ensuring that the final product meets user and business requirements effectively.

2. **Understanding the Existing System:**
   It provides a deep understanding of the current manual or existing system, identifying its strengths, weaknesses, and limitations. This helps in determining what needs to be improved or replaced in the proposed system.

3. **Identification of User Requirements:**
   System analysis ensures that user needs and expectations are clearly identified and documented. This step minimizes the risk of developing a system that fails to

satisfy end-user requirements.

4. **Improved Communication:**
   It acts as a bridge between users, stakeholders, and developers by converting business requirements into technical specifications. This ensures that all parties share a common understanding of the system objectives.

5. **Detection of Problems and Inefficiencies:**
   By analyzing data flow, processes, and workflows, system analysis identifies bottlenecks, redundancies, and inefficiencies that hinder productivity in the current system.

6. **Ensures System Quality and Reliability:**
   Proper analysis defines clear functional and non-functional requirements, resulting in a robust, reliable, and efficient system design that performs well under different conditions.

7. **Supports Feasibility Evaluation:**
   It helps assess the technical, operational, and economic feasibility of the proposed system, ensuring that the project is practical, achievable, and cost-effective.

8. **Enhances Decision-Making:**
   System analysis provides valuable insights and documentation that support informed decisions about design choices, resource allocation, and implementation strategies.

5.1 EXISTING SYSTEM

The existing system for managing events is largely **manual and paper-based**, involving several time-consuming and error-prone processes. Event organizers typically handle registrations, participant details, scheduling, and record-keeping through handwritten forms or basic spreadsheets. This approach lacks automation and centralization, making it difficult to access, update, or share information efficiently among users or departments.

In the current setup, participants are often required to **register manually** by filling out physical forms or sending details through emails, which are then recorded and managed by event coordinators. Confirmation of registration, event updates, and communication are carried out via phone calls or emails, leading to unnecessary delays and confusion. Furthermore, the absence of an integrated database makes it difficult to track participant attendance, event schedules, and resource allocation.

The manual system also faces challenges in **data accuracy and consistency**. Since most information is entered and maintained by hand, there is a high chance of errors such as duplicate entries, misplaced data, or incomplete records. Retrieving specific information, such as participant lists or event history, can be tedious and time-consuming. This lack of automation affects the overall efficiency of event management activities.

Manual Process:
The current event management process is performed manually using paper forms, emails, and spreadsheets, which makes it time-consuming and difficult to manage effectively.

Inefficient Data Handling:
Event details, participant information, and schedules are recorded manually, leading to issues such as data duplication, missing entries, and difficulty in updating records.

Lack of Centralized Storage:
There is no centralized database to store or retrieve information efficiently. As a result, data is scattered across multiple files or systems, making coordination difficult.

Poor Communication:
Communication between organizers and participants is handled through phone calls or emails, often causing delays, confusion, and lack of real-time updates.

Limited Reporting Capabilities:
Generating reports such as participant lists, event summaries, or attendance sheets requires manual compilation, which is both slow and prone to human error.

Low Data Security:
Since data is stored in physical files or unsecured spreadsheets, it is vulnerable to unauthorized access, loss, or damage, and lacks proper backup mechanisms.

Reduced Efficiency and User Satisfaction:
The manual system leads to unnecessary workload for event organizers, delayed responses, and poor user experience, emphasizing the need for an automated solution.

5.2 PROPOSED SYSTEM

The proposed Event Management System automates event operations through a secure, centralized, and user-friendly platform that ensures efficiency, accuracy, and real-time communication.

Automation of Processes:
The proposed Event Management System automates all major functions, including event creation, registration, scheduling, and report generation, reducing manual workload and human error.

Centralized Database Management:
All event-related data such as participant details, event schedules, and resources are stored in a secure, centralized database, allowing quick access, updates, and retrieval of information.

User-Friendly Interface:
The system provides an intuitive and interactive interface developed using **HTML, CSS, and React.js**, enabling both organizers and participants to navigate and perform tasks easily.

Efficient Communication:
Notifications, event updates, and confirmations are automatically sent via email or displayed on the user dashboard, ensuring timely communication and coordination between organizers and users.

Real-Time Data and Reporting:
The system generates accurate, real-time reports on registration status, attendance, and event performance, helping administrators make informed decisions quickly.

Enhanced Security and Backup:
User authentication and role-based access control ensure data confidentiality, while regular backups prevent data loss and provide reliable recovery options.

Scalability and Flexibility:
The system is designed to handle multiple events and a large number of users simultaneously. It can be easily modified or expanded to include new features as requirements grow.

5.2.1 ADVANTAGES:

1. **Improved Efficiency:**
Automation of event processes reduces manual work and saves time for both organizers and participants.

2. **High Accuracy:**
The system minimizes human errors in data entry, registration, and reporting through automated validation and database management.

3. **Centralized Data Management:**
All event-related information is stored in a single, secure database, allowing easy access, updates, and tracking.

4. **Enhanced Communication:**
Real-time notifications and updates ensure smooth coordination between event organizers and participants.

5. **Data Security and Backup:**
The use of authentication and backup mechanisms ensures data confidentiality, integrity, and protection against loss.

CHAPTER - 6

SYSTEM STUDY

6.1 FEASIBILITY STUDY

A feasibility study is conducted to determine whether the proposed system is practical, achievable, and beneficial for the organization. It evaluates different aspects of the system to ensure that it can be successfully developed, implemented, and maintained with available resources. The main goal of this study is to assess the viability of the proposed Event Management System before full-scale development begins.

The feasibility study helps identify potential challenges, estimate the cost and time required, and ensure that the proposed solution effectively meets user needs. It also provides a foundation for decision-making by comparing the expected benefits with the resources and risks involved.

6.1.1 TECHNICAL FEASIBILITY:

Technical Feasibility refers to the assessment of the hardware and software requirements needed to develop and implement the proposed system. It determines whether the current technology, infrastructure, and available resources are sufficient to support the successful development and operation of the system.

The proposed **Event Management System** is technically feasible because it utilizes widely available and reliable technologies such as **HTML, CSS, React.js, Node.js, Python, and SQLite3**. These technologies are open-source, cost-effective, and compatible with most operating systems, making the system easy to develop, deploy, and maintain.

The system requires only basic hardware configurations such as a standard computer with internet access, moderate processing power, and sufficient storage capacity. Since the application runs on a web-based platform, it does not demand high-end servers or complex networking infrastructure, reducing technical barriers to implementation.

Overall, the proposed system is **technically sound and feasible**. The technologies selected are modern, stable, and well-documented, ensuring easy development, future enhancements, and long-term system sustainability.

6.1.2. ECONOMIC FEASIBILITY:

Economic Feasibility is the process of determining whether the proposed system is financially viable and cost-effective. It involves analyzing the costs associated with the development, implementation, and maintenance of the system, and comparing them with the expected benefits and savings it will provide in the long run.

The **Event Management System** is economically feasible because it significantly reduces the expenses associated with manual processes such as paperwork, printing, and physical record-keeping. By automating event registration, scheduling, and communication, the system saves time and resources for both organizers and participants. The reduction in manual labor and administrative work leads to overall cost savings for the organization.

In the long term, the **benefits far outweigh the initial investment**, as the system enhances operational performance, reduces human errors, and provides long-term financial savings. Therefore, the Event Management System is considered **highly economical and financially viable** for implementation.

6.1.3 SOCIAL FEASIBILITY:

Social Feasibility examines how well the proposed system will be accepted and supported by the people who will use it, including administrators, organizers, participants, and other stakeholders. It focuses on user attitudes, adaptability, and the overall impact of the system on the working environment and society.

The proposed **Event Management System** is socially feasible because it simplifies the event management process, reduces manual workload, and enhances user satisfaction. Users can easily register, view event details, and receive updates through a user-friendly and interactive interface. This improves communication and engagement between event organizers and participants, fostering a more collaborative and efficient environment.

Additionally, the system is designed with user convenience in mind. Its simple layout and responsive design ensure that users with minimal technical knowledge can operate it without difficulty. The system’s ability to save time, minimize errors, and streamline communication has a positive social impact, as it promotes cooperation, efficiency, and satisfaction among all stakeholders.

Therefore, the proposed system is **socially acceptable and beneficial**, as it enhances productivity, supports sustainable practices, and is well-aligned with modern organizational and societal needs.

CHAPTER - 7

SYSTEM DESIGN

System Design is the process of defining the architecture, components, modules, interfaces, and data for a system to satisfy specified requirements. It serves as a blueprint for the development phase, translating the logical requirements identified during system analysis into a physical system that can be implemented efficiently.

The **Event Management System** is designed to automate the process of event registration, scheduling, and management. It ensures smooth communication between users and administrators, minimizes manual work, and enhances the overall user experience through a structured and interactive design.

The **frontend** is developed using **HTML, CSS, and React.js**, providing a responsive and user-friendly interface for interaction. The **backend** uses **Node.js and Python** to handle server-side logic, process requests, and communicate with the database. The **SQLite3** database stores and manages all event-related data, ensuring data integrity, accuracy, and easy retrieval.

The system architecture is based on the **client-server model**, where the client (user interface) interacts with the server through secure APIs to perform operations like login, event registration, and viewing schedules. The server processes the requests, interacts with the database, and sends responses back to the client.

- To create a structured framework that clearly defines system modules and their interactions.
- To ensure efficient data flow between the user interface, backend logic, and database.
- To provide a secure, scalable, and maintainable architecture for the Event Management System.
- To enhance performance, minimize redundancy, and ensure data consistency.
- To deliver a user-friendly interface that meets the needs of both administrators and participants.

7.1 DATABASE DESIGN

Database Design is a critical phase of system development that focuses on the logical and physical structure of the database. It defines how data will be stored, organized, and accessed efficiently. A well-designed database ensures data integrity, consistency, and accuracy while supporting the functional requirements of the system.

In the **Event Management System**, the database is designed using **SQLite3**, a lightweight and efficient relational database management system (RDBMS). It stores and manages all information related to users, events, registrations, schedules, and reports. The database design follows the principles of normalization to eliminate redundancy and ensure data accuracy.

The database consists of multiple interrelated tables, each representing a key entity of the system. Relationships between these tables are established using primary and foreign keys, allowing efficient data retrieval and maintaining referential integrity.

- To organize data systematically for efficient access and management.
- To eliminate data redundancy and maintain data integrity.
- To ensure secure storage and controlled access to sensitive information.
- To provide flexibility for easy updates, queries, and reporting.
- To support the scalability and performance needs of the system.

There are several fundamental principles and theories that guide the process of effective database design. These principles ensure that the database is efficient, reliable, and capable of meeting the system's functional requirements.

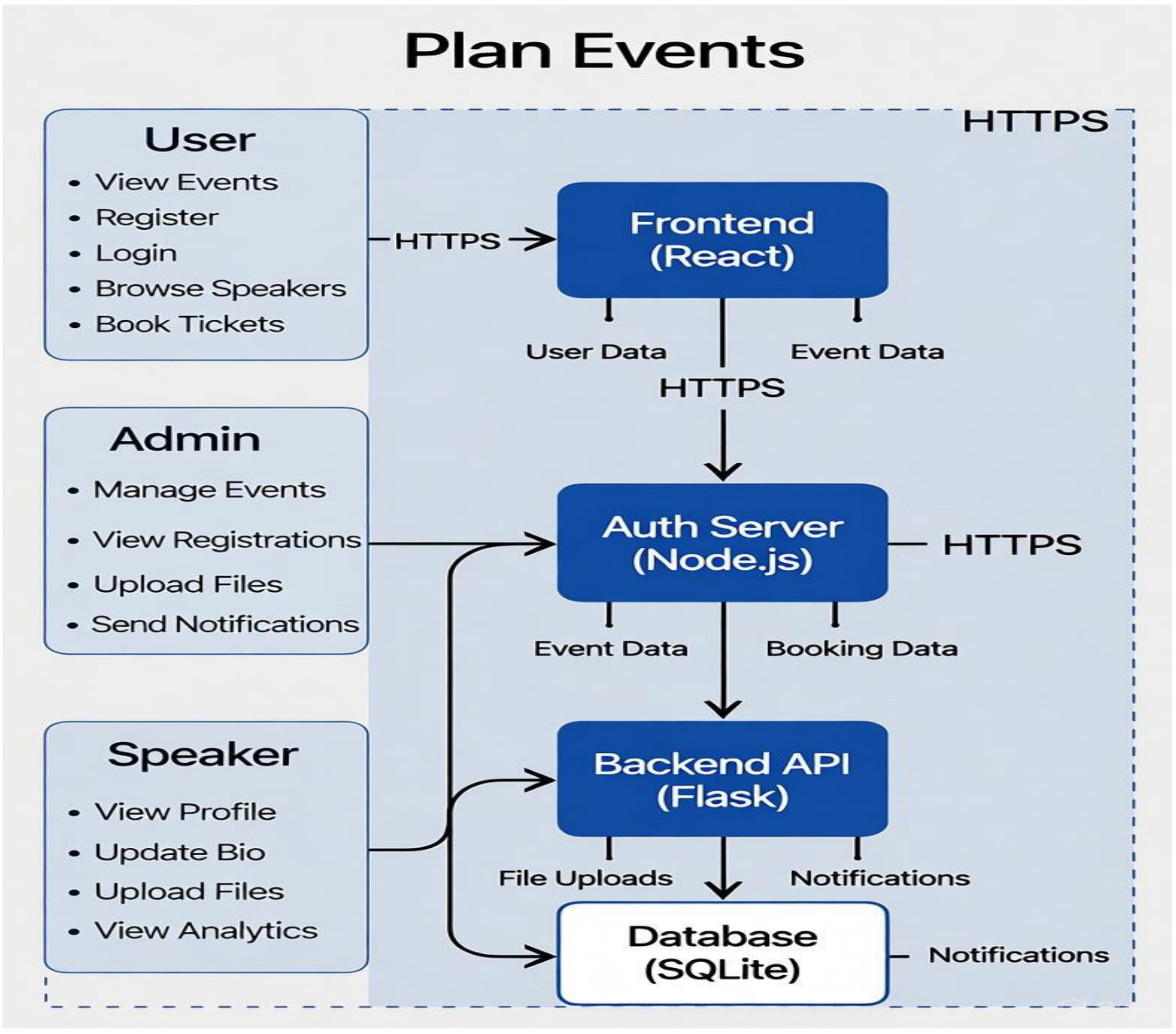
1. **Entity–Relationship (ER) Modeling:**
   ER modeling is one of the most widely used techniques in database design. It involves identifying the key entities (such as users, events, and registrations) and defining the relationships between them. ER diagrams are used to visually represent these entities, their attributes, and how they interact, providing a clear structure for the database.
2. **Normalization:**
   Normalization is the process of organizing data in a database to minimize redundancy and ensure data integrity. It involves dividing large tables into smaller, related tables and defining relationships between them. This helps prevent data duplication and ensures that each piece of data is stored only once, improving consistency and efficiency.
3. **Data Integrity:**
   Data integrity ensures that the information stored in the database is accurate, complete, and consistent. It is maintained through the use of constraints such as primary keys, foreign keys, and validation rules. High data integrity reduces errors and enhances the reliability of the entire system.
4. **Data Security:**
   Data security refers to protecting data from unauthorized access, modification, or destruction. In the Event Management System, security measures such as user authentication, access control, and database encryption are implemented to safeguard sensitive information like user details and event records.

5. **Performance:**
Database performance focuses on the speed and efficiency with which data is accessed, processed, and retrieved. A well-designed database ensures fast query execution and optimal storage usage. Indexing, query optimization, and proper schema design are crucial factors that improve performance and scalability.

7.2 DATAFLOW DIAGRAM

A **Data Flow Diagram (DFD)** is a graphical representation that illustrates how data moves through a system, showing the flow of information between processes, data stores, and external entities. It helps in understanding how the system functions and how various components interact with each other. DFDs are useful for analyzing the logical flow of data and identifying inputs, processes, and outputs.

The **Event Management System** uses DFDs to represent how data flows between users, the system processes, and the database. The diagrams help to visualize the system's operations, ensuring that all processes are well-structured and that data handling is efficient and accurate.



1. Context Level DFD (Level 0)

The **Context Level DFD** provides an overview of the entire system as a single process, showing its interaction with external entities such as users and administrators.

1.1 External Entities:

Administrator: Manages events, users, and reports.

User/Participant: Registers for events, views schedules, and provides feedback.

1.2 Processes:

- **Event Management System:** The central process that handles all user and admin requests.

1.3 Data Stores:

- **Database:** Stores user details, event information, registration records, and feedback.

1.4 Data Flow:

- Users send registration and login requests → The system verifies and processes the data → The system retrieves or stores data in the database → Responses (such as confirmations or event details) are sent back to the users.

○

2. Level 1 DFD

The **Level 1 DFD** expands the main system process into several sub-processes that show detailed data movement.

2.1 Main Processes:

User Registration and Login:
Users enter details which are validated and stored in the user database. Upon successful login, they can access event functionalities.

Event Creation and Management:
Admins add, update, or delete event details. The information is stored in the event table of the database.

Event Registration:
Users select events and register. Their registration details are verified and stored in the registration database.

Schedule Management:
The system maintains the event schedule, including dates, venues, and timings,

accessible to both users and admins.

Feedback Management:
After events, users can provide feedback, which is stored and later reviewed by administrators.

2.2 Data Stores:

User Data Store: Contains user profiles and login credentials.

Event Data Store: Stores event information and descriptions.

Registration Data Store: Maintains participant registration details.

Feedback Data Store: Stores user feedback and ratings.

2.3 Data Flow Summary:

- Data flows between users and the system through forms and interfaces.

- The system processes inputs, performs validation, and stores or retrieves data as needed.

- Processed outputs are then displayed or sent back to users or administrators.

3. Importance of DFD in the System

- Helps in understanding the logical structure and data movement of the system.

- Identifies key inputs, processes, and outputs clearly.

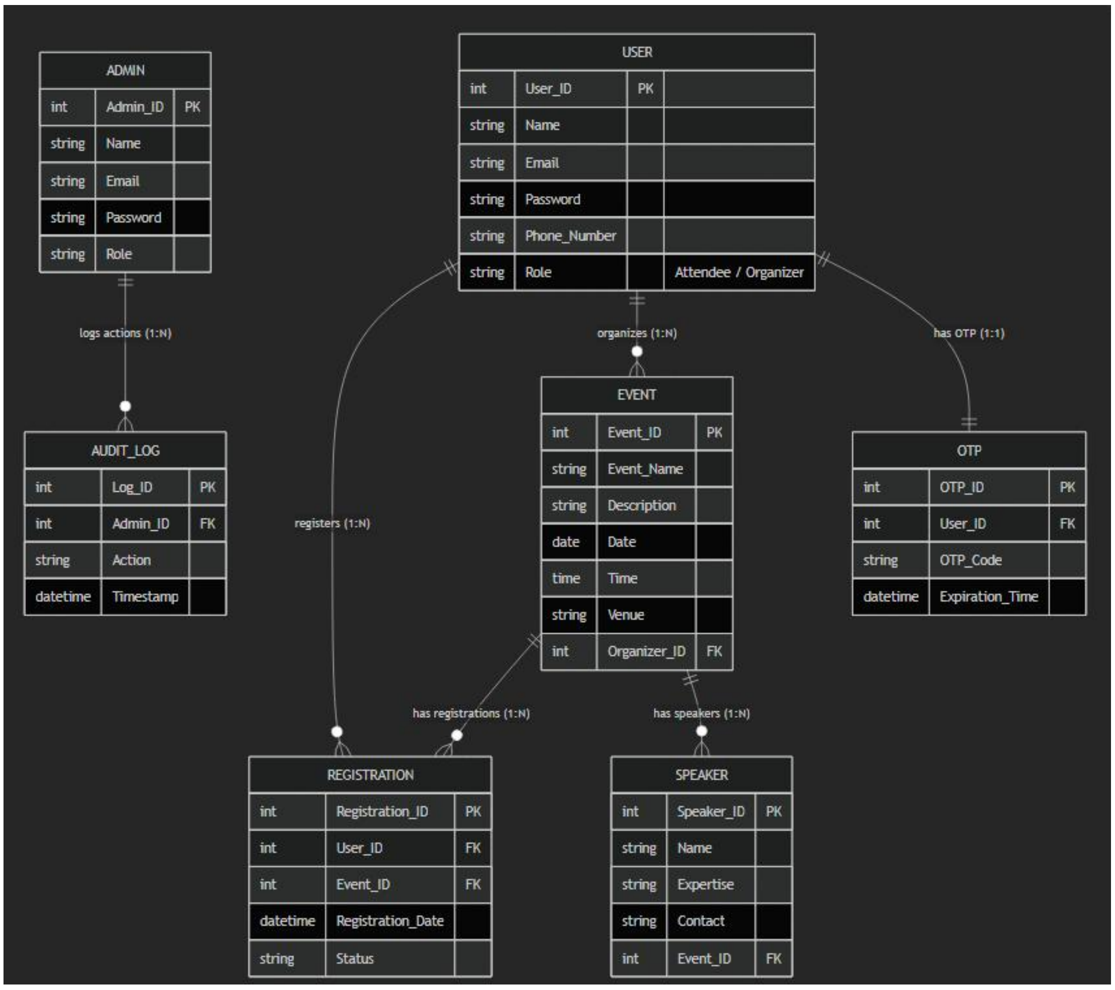
- Aids developers in designing efficient database structures and interactions.

- Ensures transparency in system operations and communication flow.

7.3 ER Diagram

An **Entity–Relationship (ER) Diagram** is a visual representation of the data model that illustrates how entities (tables) are related within a system. It defines the logical structure of the database and helps in identifying the relationships between different data components.

For the **Event Management System**, the ER diagram provides a detailed view of entities such as users, events, registrations, and feedback, along with their attributes and relationships.



ADMIN		
int	Admin_ID	PK
string	Name	
string	Email	
string	Password	
string	Role	

USER			
int	User_ID	PK	
string	Name		
string	Email		
string	Password		
string	Phone_Number		
string	Role		Attendee / Organizer

AUDIT_LOG		
int	Log_ID	PK
int	Admin_ID	FK
string	Action	
datetime	Timestamp	

EVENT		
int	Event_ID	PK
string	Event_Name	
string	Description	
date	Date	
time	Time	
string	Venue	
int	Organizer_ID	FK

OTP		
int	OTP_ID	PK
int	User_ID	FK
string	OTP_Code	
datetime	Expiration_Time	

REGISTRATION		
int	Registration_ID	PK
int	User_ID	FK
int	Event_ID	FK
datetime	Registration_Date	
string	Status	

SPEAKER		
int	Speaker_ID	PK
string	Name	
string	Expertise	
string	Contact	
int	Event_ID	FK

1. Entities and Attributes

a) User

User_ID (PK): Unique identification number for each user.

Name: Full name of the user.

Email: Email address used for login and communication.

Password: Encrypted user password.

Role: Defines the role of the user (Admin / Participant).

Phone: Contact number of the user.

b) Event

Event_ID (PK): Unique identification number for each event.

Event_Name: Title of the event.

Description: Brief details about the event.

Date: Scheduled date of the event.

Venue: Location of the event.

Organizer_ID (FK): References the user organizing the event (User_ID).

c) Registration

Registration_ID (PK): Unique identification number for each registration.

User_ID (FK): References the participant registering for the event.

Event_ID (FK): References the event being registered for.

Registration_Date: Date when the user registered.

Status: Indicates confirmation or cancellation status.

d) Feedback

Feedback_ID (PK): Unique identification number for each feedback entry.

User_ID (FK): References the user giving the feedback.

Event_ID (FK): References the event for which feedback is given.

Rating: Rating provided by the user.

Comments: Textual feedback or suggestions from the user.

2. Relationships Between Entities

Relationship	Description
User → Event	One user (admin) can organize many events. (1:M)
User → Registration	One user can register for many events. (1:M)
Event → Registration	One event can have many registered users. (1:M)
User → Feedback	One user can give feedback for multiple events. (1:M)
Event → Feedback	One event can have feedback from many users. (1:M)
User/Event → Payment	A user can make multiple payments for various events. (1:M)

CHAPTER - 8

SYSTEM MODULE

A system module is a distinct component of a software system that performs a specific function.

The **Plan Events Application** is divided into several interrelated modules, each responsible for a specific task. These modules work together to provide an efficient, user-friendly, and automated platform for managing events, registrations, and participants.

Each module within the system is designed to operate independently yet remain fully integrated with other modules to ensure seamless communication and data consistency. This modular structure allows for easy maintenance, scalability, and flexibility—enabling developers to add or modify features without affecting the entire system.

The modular design also enhances system performance by dividing complex operations into smaller, manageable components such as **User Management**, **Event Management**, **Registration Handling**, **Notification and Communication**, and **Reporting & Analytics**. These modules collaborate to deliver a streamlined experience for administrators, event organizers, and users alike.

By employing a modular architecture, the Plan Events Application ensures that the system remains robust, adaptable to future upgrades, and capable of supporting diverse event types and user requirements. This approach ultimately leads to improved reliability, efficiency, and user satisfaction across all levels of system operation.

Stakeholders:

Users (Attendees): Use the platform to browse, register for, and manage event participation.

Event Organizers / Admins: Manage events, speakers, users, and system configurations.

Speakers: Manage their profiles, view assigned events, and share session details.

System Administrators: Maintain backend systems, ensure data security, and manage platform integrity.

Economic Feasibility:

- Low development and maintenance costs using **open-source technologies**.

- Reduces manual workload by automating event registration and management.

- Generates potential revenue from event hosting fees or premium listing features.

- Increases efficiency and reduces human error through digital record management.

Technical Feasibility:

- Developed using **React.js (Frontend)**, **Flask + Node.js (Backend)**, and **SQLite3 (Database)**.

- Supports RESTful APIs for seamless communication between frontend and backend.

- Integrates **OTP-based authentication** for secure login and password recovery.

- Enables **JSON-based data handling** for flexible event and speaker management.

- Deployed in a **modular, scalable architecture** that supports future expansion (e.g., online payments, analytics).

Operational Feasibility:

- Provides a **user-friendly interface** that allows easy navigation and interaction.

- Simplifies **event scheduling and participant management** through automation.

- Supports role-based access for Admins, Users, and Speakers for secure operations.

- Offers real-time registration and event status tracking to improve management efficiency.

System Workflow – Step-by-Step Process

User Registration & Login:
Users sign up, verify via OTP, and log in to access event features.

Event Creation (Admin):
Admins or organizers create and publish event details including date, time, venue, and category.

Event Browsing:
Users view and filter events by category, location, or type.

Registration:
Users register for events by providing details such as name, email, and ticket type.

Confirmation & Notification:
The system confirms the registration and sends an email notification.

Speaker Assignment:
Admins assign speakers to events, managed through JSON-based relationships.

Audit Logging:
All administrative actions are recorded in the **Audit Log** module for transparency.

Feedback & Reporting:
Admins can monitor registrations, generate summaries, and view analytics.

System Architecture – Architectural Model

The **Plan Events Application** follows a **3-Tier Architecture**:

Presentation Layer:
Developed using **React.js**, it provides the user interface for browsing, registration, and admin management.

Application Layer:
Implemented using **Flask** and **Node.js**, this layer handles business logic such as event scheduling, authentication, and speaker management.

Database Layer:
Managed using **SQLite3**, it stores all essential data, including users, events, registrations, and audit logs.

Functional & Non-Functional Requirements

Functional Requirements:

- Users must be able to **register, log in, and reset passwords** using OTP verification.

- Admins can **create, update, and delete events**.

- Users can **view and register for events** online.

- Speakers can **update their profiles** and **link to events**.

- The system must **record all admin activities** in an Audit Log.

Non-Functional Requirements:

Performance:
System should handle multiple concurrent users efficiently.

Security:
OTP verification and password hashing ensure secure authentication.

Scalability:
Designed for easy integration with future payment or analytics modules.

Usability:
Clean UI and intuitive navigation ensure easy adoption by all users.

Reliability:
Data consistency ensured across all modules through centralized database.

Security Considerations

Authentication & Authorization:

- Email-based registration and **OTP verification** for password resets.
- Role-based access (Admin, User, Speaker) ensures proper authorization levels.

Data Protection:

- Passwords are stored in **hashed form** for security.
- Event and registration data are stored in a **secure SQLite database**.
- Communication between frontend and backend uses secure HTTP protocols.

System Backup & Recovery:

- Regular **database backups** ensure no data loss.
- Local and cloud storage support ensures **data reliability and recovery**.

CHAPTER - 9

SAMPLE SOURCE CODE

9.1 HTML CODE

public/index.html

```html
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <link rel="icon" href="%PUBLIC_URL%/plan_events.png" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <meta name="theme-color" content="#000000" />
    <meta
      name="description"
      content="Web site created using create-react-app"
    />
    <link rel="apple-touch-icon" href="%PUBLIC_URL%/logo192.png" />
    <link rel="manifest" href="%PUBLIC_URL%/manifest.json" />
    <link
href="https://fonts.googleapis.com/css2?family=DM+Sans:ital,opsz,wght@0,9..40,100..1000;1,9..4
0,100..1000&display=swap" rel="stylesheet">
    <link
href="https://fonts.googleapis.com/css2?family=DM+Mono:ital,wght@0,300..500;1,300..500&display
=swap" rel="stylesheet">
    <title>Plan Events</title>
    <meta name="description" content="Plan and organize your events seamlessly with our app."
/>
  </head>
  <body>
    <noscript>You need to enable JavaScript to run this app.</noscript>
    <div id="root"></div>
  </body>
</html>
```

9.2 CSS CODE

src/app.css

```css
* {
  margin: 0;
  padding: 0;
  box-sizing: border-box;
}

body {
  font-family: 'DM Sans', sans-serif;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
  line-height: 1.6;
  background-color: #ffffff;
  color: #333;
}

.App {
  min-height: 100vh;
  display: flex;
  flex-direction: column;
}

a {
  color: inherit;
  text-decoration: none;
}

button {
  font-family: inherit;
}
```

9.3 REACT. JS

src/app.js

```javascript
import React, { useEffect, useState } from 'react';
import { Routes, Route, useLocation, Navigate } from 'react-router-dom';
import { ToastContainer } from 'react-toastify';
import 'react-toastify/dist/ReactToastify.css';
import './App.css';
import Header from './components/Header';
import AuthProvider, { useAuth } from './contexts/AuthContext';
import Hero from './components/Hero';
import Features from './components/Features';
import EventPreview from './components/EventPreview';
import InTheSpotlight from './components/InTheSpotlight';
import Footer from './components/Footer';
import EventList from './components/EventList';
import EventDetails from './components/EventDetails';
import EventForm from './components/EventForm';
import RegistrationForm from './components/RegistrationForm';
import About from './components/About';
import Contact from './components/Contact';
import Products from './components/Products';
import PrivacyPolicy from './components/PrivacyPolicy';
import TermsOfService from './components/TermsOfService';
import PlanningSoftware from './components/PlanningSoftware';
import SchedulingTools from './components/SchedulingTools';
import BudgetManagement from './components/BudgetManagement';
import TeamCollaboration from './components/TeamCollaboration';
import Newsletter from './components/Newsletter';
import Careers from './components/Careers';
import CareersOverview from './components/CareersOverview';
import CareersOpenPositions from './components/CareersOpenPositions';
import CareersCulture from './components/CareersCulture';
import CareersBenefits from './components/CareersBenefits';
import CareersInclusion from './components/CareersInclusion';
import CareersEventPlanning from './components/CareersEventPlanning';
import CareersMarketing from './components/CareersMarketing';
import CareersEngineering from './components/CareersEngineering';
import CareersAdministration from './components/CareersAdministration';
import CareersInterviewing from './components/CareersInterviewing';
import CareersInternships from './components/CareersInternships';
import JobApplication from './components/JobApplication';
import OurTeams from './components/OurTeams';
import AwardsRecognition from './components/AwardsRecognition';
import SecurityTrust from './components/SecurityTrust';
import VenueLogistics from './components/VenueLogistics';
import VenueBooking from './components/VenueBooking';
import LogisticsManagement from './components/LogisticsManagement';
import VendorCoordination from './components/VendorCoordination';
import OnlineTicketing from './components/OnlineTicketing';
import RegistrationForms from './components/RegistrationForms';
import PaymentProcess from './components/PaymentProcess';
import AttendanceTracking from './components/AttendanceTracking';
import TicketingOverview from './components/TicketingOverview';
import FeedbackSurvey from './components/FeedbackSurvey';
import Feedback from './components/Feedback';
//user
import SignUp from './components/SignUp';
import Login from './components/Login';
//admin
import AdminLogin from './components/AdminLogin';
import AdminSignUp from './components/AdminSignUp';
import AdminDashboard from './components/AdminDashboard';
import ProtectedAdminRoute from './components/ProtectedAdminRoute';
//speaker
import SpeakerLogin from './components/SpeakerLogin';
import SpeakerSignUp from './components/SpeakerSignUp';
import SpeakerDashboard from './components/SpeakerDashboard';
import SpeakerProfile from './components/SpeakerProfile';
import SpeakerUpcomingTalks from './components/SpeakerUpcomingTalks';
import SpeakerPastEvents from './components/SpeakerPastEvents';
import SpeakerSettings from './components/SpeakerSettings';
import ProtectedSpeakerRoute from './components/ProtectedSpeakerRoute';

import ForgotPassword from './components/ForgotPassword';
import Dashboard from './components/Dashboard';
import Profile from './components/Profile';
import Tickets from './components/Tickets';
import Recents from './components/Recents';
import Settings from './components/Settings';
import AdminUserEdit from './components/AdminUserEdit';
import AdminUserManagement from './components/AdminUserManagement';
import AdminEvents from './components/AdminEvents';
import AdminEventEdit from './components/AdminEventEdit';
import AdminTickets from './components/AdminTickets';
import AdminAnalytics from './components/AdminAnalytics';
import AdminSettings from './components/AdminSettings';

// Protected Route Component
const ProtectedRoute = ({ children }) => {
  const { user, loading } = useAuth();
  if (loading) {
    return <div>Loading...</div>;
  }
  return user ? children : <Navigate to="/login" />;
};

function App() {
  const location = useLocation();
  const [showFeedback, setShowFeedback] = useState(false);

  useEffect(() => {
    const observer = new IntersectionObserver((entries) => {
      entries.forEach(entry => {
        if (entry.isIntersecting) {
          entry.target.classList.add('animate');
        }
      });
    }, { threshold: 0.1 });

    const elements = document.querySelectorAll('.animate-on-scroll');
    elements.forEach(el => observer.observe(el));

    return () => observer.disconnect();
  }, [location]);

  useEffect(() => {
    const toggleFeedbackHandler = (e) => {
      setShowFeedback(e.detail);
    };
    window.addEventListener('toggleFeedback', toggleFeedbackHandler);
    return () => {
      window.removeEventListener('toggleFeedback', toggleFeedbackHandler);
    };
  }, []);

  const handleCloseFeedback = () => {
    setShowFeedback(false);
  };

  return (
    <AuthProvider>
      <div className="App">
        <Header />
        <Routes>
          <Route path="/" element={
            <>
              <Hero />
              <Features />
              <EventPreview />
              <InTheSpotlight />
            </>
          } />
          <Route path="/events" element={<EventList />} />
          <Route path="/events/new" element={<EventForm />} />
          <Route path="/events/:id" element={<EventDetails />} />
          <Route path="/events/:id/register" element={<RegistrationForm />} />
          <Route path="/products" element={<Products />} />
          <Route path="/products/planning-software" element={<PlanningSoftware />} />
          <Route path="/products/scheduling-tools" element={<SchedulingTools />} />
          <Route path="/products/budget-management" element={<BudgetManagement />} />
          <Route path="/products/team-collaboration" element={<TeamCollaboration />} />
          <Route path="/products/venue-logistics" element={<VenueLogistics />} />
          <Route path="/products/venue-booking" element={<VenueBooking />} />
          <Route path="/products/logistics-management" element={<LogisticsManagement />} />
          <Route path="/products/vendor-coordination" element={<VendorCoordination />} />
          <Route path="/ticketing">
            <Route index element={<TicketingOverview />} />
            <Route path="online" element={<OnlineTicketing />} />
            <Route path="registration" element={<RegistrationForms />} />
            <Route path="payment" element={<PaymentProcess />} />
            <Route path="attendance" element={<AttendanceTracking />} />
          </Route>
          <Route path="/about" element={<About />} />
          <Route path="/about/who-we-are" element={<About />} />
          <Route path="/about/our-teams" element={<OurTeams />} />
          <Route path="/about/awardsrecognition" element={<AwardsRecognition />} />
          <Route path="/about/security-trust" element={<SecurityTrust />} />
          <Route path="/contact" element={<Contact />} />
          <Route path="/privacy" element={<PrivacyPolicy />} />
          <Route path="/terms" element={<TermsOfService />} />
          <Route path="/newsletter" element={<Newsletter />} />
          <Route path="/careers" element={<Careers />}>
            <Route index path="overview" element={<CareersOverview />} />
            <Route path="open-positions" element={<CareersOpenPositions />} />
            <Route path="culture" element={<CareersCulture />} />
            <Route path="benefits" element={<CareersBenefits />} />
            <Route path="inclusion" element={<CareersInclusion />} />
            <Route path="event-planning" element={<CareersEventPlanning />} />
            <Route path="marketing" element={<CareersMarketing />} />
            <Route path="engineering" element={<CareersEngineering />} />
            <Route path="administration" element={<CareersAdministration />} />
            <Route path="interviewing" element={<CareersInterviewing />} />
            <Route path="internships" element={<CareersInternships />} />
          </Route>
          <Route path="/careers/apply/:id" element={<JobApplication />} />
          <Route path="/signup" element={<SignUp />} />
          <Route path="/login" element={<Login />} />
          <Route path="/admin/login" element={<AdminLogin />} />
          <Route path="/admin/signup" element={<AdminSignUp />} />
          <Route path="/admin/dashboard" element={<ProtectedAdminRoute><AdminDashboard
/></ProtectedAdminRoute>} />
          <Route path="/admin/users" element={<ProtectedAdminRoute><AdminUserManagement
/></ProtectedAdminRoute>} />
          <Route path="/admin/users/:id/edit" element={<ProtectedAdminRoute><AdminUserEdit
/></ProtectedAdminRoute>} />
          <Route path="/admin/events" element={<ProtectedAdminRoute><AdminEvents
/></ProtectedAdminRoute>} />
          <Route path="/admin/events/:id/edit" element={<ProtectedAdminRoute><AdminEventEdit
/></ProtectedAdminRoute>} />
          <Route path="/admin/tickets" element={<ProtectedAdminRoute><AdminTickets
/></ProtectedAdminRoute>} />
          <Route path="/admin/analytics" element={<ProtectedAdminRoute><AdminAnalytics
/></ProtectedAdminRoute>} />
```

```
          <Route path="/admin/settings" element={<ProtectedAdminRoute><AdminSettings
/></ProtectedAdminRoute>} />
          <Route path="/speaker/login" element={<SpeakerLogin />} />
          <Route path="/speaker/signup" element={<SpeakerSignUp />} />
          <Route path="/speaker/dashboard" element={<ProtectedSpeakerRoute><SpeakerDashboard
/></ProtectedSpeakerRoute>} />
          <Route path="/speaker/profile" element={<ProtectedSpeakerRoute><SpeakerProfile
/></ProtectedSpeakerRoute>} />
          <Route path="/speaker/talks" element={<ProtectedSpeakerRoute><SpeakerUpcomingTalks
/></ProtectedSpeakerRoute>} />
          <Route path="/speaker/past-events"
element={<ProtectedSpeakerRoute><SpeakerPastEvents /></ProtectedSpeakerRoute>} />
          <Route path="/speaker/settings" element={<ProtectedSpeakerRoute><SpeakerSettings
/></ProtectedSpeakerRoute>} />
          <Route path="/forgot" element={<ForgotPassword />} />
          <Route path="/dashboard" element={<ProtectedRoute><Dashboard /></ProtectedRoute>} />
          <Route path="/profile" element={<ProtectedRoute><Profile /></ProtectedRoute>} />
          <Route path="/registrations" element={<ProtectedRoute><Tickets /></ProtectedRoute>}
/>
          <Route path="/recents" element={<ProtectedRoute><Recents /></ProtectedRoute>} />
          <Route path="/settings" element={<ProtectedRoute><Settings /></ProtectedRoute>} />
        </Routes>
        {location.pathname !== '/dashboard' && location.pathname !== '/recents' &&
location.pathname !== '/settings' && location.pathname !== '/registrations' &&
location.pathname !== '/ticketing/attendance' && location.pathname !== '/profile' &&
location.pathname !== '/admin/events' && location.pathname !== '/admin/users' &&
location.pathname !== '/admin/dashboard' && location.pathname !== '/admin/profile' &&
location.pathname !== '/admin/tickets' && location.pathname !== '/admin/analytics' &&
location.pathname !== '/admin/settings' && location.pathname !== '/speaker/dashboard' &&
location.pathname !== '/speaker/profile' && !location.pathname.startsWith('/admin/users/') &&
<Feedback />}
        {showFeedback && <FeedbackSurvey onClose={handleCloseFeedback} />}
        <Footer />
        <ToastContainer />
      </div>
    </AuthProvider>
  );
}

export default App;
```

/frontend/components/Hero.js

```
<section className="hero">
      <div className="hero-container">
        <div className="hero-content animate-on-scroll slide-up">
          <h1 className="hero-title">PLAN. MANAGE. <br /> SUCCEED.</h1>
          <p className="hero-subtitle">
            Organize and scale your events effortlessly with our comprehensive event
management platform. From planning to execution, we provide the tools to make every event
unforgettable.
          </p>
          <div className="hero-actions animate-on-scroll slide-up">
            <Link to="/events" className="btn-primary-large">Browse Events</Link>
            <Link to="/events/new" className="btn-secondary-large">Try it Free</Link>
          </div>
        </div>
        <div className="hero-image animate-on-scroll slide-up">
          <img src={img} alt="Hero" height={'450px'} className="hero-img" />
        </div>
      </div>
    </section>
```

/frontend/components/Hero.css

```
.hero {
  background: #f8f9fa;
  color: #333;
  padding: 4rem 2rem;
  position: relative;
  overflow: hidden;
}

.hero-container {
  max-width: 1200px;
  margin: 0 auto;
  display: flex;
  align-items: flex-start;
  justify-content: space-between;
  gap: 3rem;
}

.hero-content {
  flex: 1;
  margin: 30px 0 60px 0;
  max-width: 600px;
}

.hero-title {
  font-size: 3.5rem;
  font-weight: bold;
  margin-bottom: 1rem;
  color: #333;
  line-height: 1.2;
  text-align: left;
}

.hero-subtitle {
  font-size: 1.3rem;
  margin-bottom: 2rem;
  line-height: 1.6;
  color: #666;
  text-align: left;
}

.hero-actions {
  display: flex;
  gap: 1rem;
  flex-wrap: wrap;
  margin-bottom: 20px;
}

.btn-primary-large {
  background: #ff6b35;
  border: none;
  color: white;
  padding: 1rem 2rem;
  font-size: 1rem;
  border-radius: 4px;
```

```css
  cursor: pointer;
  transition: background 0.3s;
  font-weight: 500;
}

.btn-primary-large:hover {
  background: #e55a2b;
}

.btn-secondary-large {
  background: #fff;
  border: 1px solid #292828;
  color: #333;
  padding: 1rem 2rem;
  font-size: 1rem;
  border-radius: 4px;
  cursor: pointer;
  transition: background 0.3s;
  font-weight: 500;
}

.btn-secondary-large:hover {
  background: #333;
  color: white;
}

.hero-img {
  background: rgb(255, 107, 53);
}

.feedback-button {
  position: fixed;
  left: 0;
  top: 50%;
  transform: translateY(-50%);
  background: #ff6b35;
  color: white;
  padding: 1rem;
  border: none;
  border-radius: 0 8px 8px 0;
  cursor: pointer;
  writing-mode: vertical-rl;
  text-orientation: mixed;
  z-index: 1000;
  font-weight: bold;
  width: 50px;
  transition: width 0.3s ease;
}

.feedback-button:hover {
  width: 65px;
}

.modal-overlay {
  position: fixed;
  top: 0;
  left: 0;
  width: 100%;
  height: 100%;
  background: rgba(0, 0, 0, 0.5);
  backdrop-filter: blur(5px);
  display: flex;
  justify-content: center;
  align-items: center;
  z-index: 2000;
}

.modal-content {
  background: #fff;
  border-radius: 12px;
  padding: 2rem;
  box-shadow: 0 8px 32px rgba(0, 0, 0, 0.3);
  text-align: left;
  width: 400px;
  border: 2px solid #ff6b35;
}

.modal-content h3 {
  margin-bottom: 1rem;
  color: #333;
  font-size: 1.5rem;
}

.modal-content p {
  margin-bottom: 1rem;
  color: #666;
}

.modal-content textarea {
  width: 100%;
  height: 100px;
  padding: 0.5rem;
  border: 1px solid #ddd;
  border-radius: 4px;
  margin-bottom: 1rem;
  resize: vertical;
  font-family: inherit;
}

.btn-feedback {
  background: #ff6b35;
  border: none;
  color: white;
  padding: 0.5rem 1rem;
  border-radius: 4px;
  cursor: pointer;
  transition: background 0.3s;
  font-size: 1rem;
}

.btn-feedback:hover {
  background: #e55a2b;
}

.hero-tabs {
  display: flex;
  gap: 0.5rem;
  margin-bottom: 1rem;
  border-bottom: 1px solid #eee;
  padding-bottom: 0.5rem;
}
```

```css
.tab {
  background: none;
  border: none;
  padding: 0.5rem 1rem;
  color: #666;
  cursor: pointer;
  font-size: 1rem;
  position: relative;
  transition: color 0.3s;
}

.tab:hover {
  color: #ff6b35;
}
.tab.active {
  color: #ff6b35;
  font-weight: 600;
}

.tab.active::after {
  content: '';
  position: absolute;
  bottom: -1px;
  left: 0;
  width: 100%;
  height: 2px;
  background: #ff6b35;
}

.hero-features {
  list-style: none;
  padding: 0;
  margin: 1.5rem 0 2rem 0;
}

.hero-features li {
  display: flex;
  align-items: center;
  margin-bottom: 0.75rem;
  font-size: 1.1rem;
  color: #333;
  line-height: 1.4;
}

.feature-icon {
  margin-right: 0.75rem;
  font-size: 1.3rem;
}

.hero-visuals {
  display: flex;
  flex-direction: column;
  gap: 1.5rem;
  align-items: center;
  height: 400px;
  justify-content: space-around;
  width: 100%;
}

.visual-item {
  text-align: center;
  width: 100%;
  flex: 1;
}

.visual-label {
  margin-top: 0.5rem;
  font-weight: 500;
  color: #666;
  font-size: 0.9rem;
}

.chart-bar {
  display: flex;
  gap: 0.5rem;
  height: 80px;
  align-items: flex-end;
  justify-content: space-around;
  background: #f8f9fa;
  padding: 1rem;
  border-radius: 8px;
  border: 1px solid #e9ecef;
}

.bar {
  width: 25px;
  background: linear-gradient(to top, #ff6b35, #ff8c5a);
  border-radius: 4px 4px 0 0;
  transition: height 0.3s ease;
}

.chart-line {
  position: relative;
  height: 80px;
  background: #f8f9fa;
  border-radius: 8px;
  padding: 1rem;
  border: 1px solid #e9ecef;
  overflow: hidden;
}

.chart-line::before {
  content: '';
  position: absolute;
  bottom: 0;
  left: 10%;
  right: 10%;
  height: 2px;
  background: #ddd;
}

.line-point {
  position: absolute;
  width: 10px;
  height: 10px;
  background: #ff6b35;
  border-radius: 50%;
  border: 2px solid white;
  box-shadow: 0 2px 4px rgba(0,0,0,0.1);
}

.chart-metrics {
  height: 80px;
  display: flex;
  align-items: center;
  justify-content: center;
  background: #f8f9fa;
  border-radius: 8px;
  border: 1px solid #e9ecef;
}

.metric-circle {
  width: 60px;
  height: 60px;
  border-radius: 50%;
  background: conic-gradient(#ff6b35 0deg 252deg, #ddd 252deg 360deg);
  position: relative;
}

.metric-circle::after {
  content: '';
  position: absolute;
  top: 50%;
  left: 50%;
  transform: translate(-50%, -50%);
  width: 40px;
  height: 40px;
  background: white;
  border-radius: 50%;
}

.btn-cta {
  background: #ff6b35;
  color: white;
  border: none;
  padding: 0.75rem 1.5rem;
  border-radius: 4px;
  cursor: pointer;
  font-weight: 500;
  font-size: 1rem;
  transition: background 0.3s;
  margin-top: auto;
}

/* Mobile Responsiveness */
@media (max-width: 768px) {
  .hero {
    padding: 2rem 1rem;
  }

  .hero-container {
    flex-direction: column;
    gap: 2rem;
    text-align: left;
  }

  .hero-content {
    max-width: none;
  }

  .hero-title {
    font-size: 2.5rem;
  }
  .hero-subtitle {
    font-size: 1.1rem;
  }

  .hero-actions {
    justify-content: center;
  }

  .hero-actions button {
    width: 100%;
    max-width: 200px;
  }

  .hero-visuals {
    height: auto;
    flex-direction: column;
    gap: 1rem;
  }

  .visual-item {
    width: 100%;
  }

  .chart-bar,
  .chart-line,
  .chart-metrics {
    height: 60px;
  }

  .modal-content {
    width: 90%;
    max-width: 400px;
  }

  .feedback-button {
    display: none; /* Hide on mobile or adjust */
  }
}

@media (max-width: 480px) {
  .hero {
    padding: 1.5rem 1rem;
  }

  .hero-title {
    font-size: 2rem;
  }

  .hero-subtitle {
    font-size: 1rem;
  }

  .hero-features li {
    font-size: 1rem;
  }

  .feature-icon {
    font-size: 1.1rem;
  }
}
```

/frontend/components/Products.js

```
<section className="products">
      <div className="products-container">
        <div className="products-content animate-on-scroll slide-up">
          <h1 className="products-title">Our Products</h1>
          <p className="products-subtitle">
            Discover our suite of tools designed to streamline every aspect of event
management. From planning to execution, we have the solutions you need.
          </p>
        </div>
        <div className="products-grid animate-on-scroll slide-up">
          {products.map((product, index) => (
            <Link key={index} to={product.path} className="product-card">
              <h3>{product.icon} {product.name}</h3>
              <p>{product.description}</p>
            </Link>
          ))}
        </div>
      </div>
    </section>
```

/frontend/components/Products.css

```
.products {
  background: #f8f9fa;
  color: #333;
  padding: 4rem 2rem;
  position: relative;
  overflow: hidden;
}

.products-container {
  max-width: 1200px;
  margin: 0 auto;
  display: flex;
  flex-direction: column;
  align-items: center;
  gap: 3rem;
}

.hero-section {
  text-align: center;
  margin-bottom: 3rem;
}

.hero-section h1 {
  font-size: 2.5rem;
  margin-bottom: 1rem;
  color: #333;
}

.hero-section p {
  font-size: 1.2rem;
  color: #666;
  line-height: 1.6;
  max-width: 600px;
  margin: 0 auto 2rem;
}
```

```css
.features-section {
  text-align: center;
  margin-bottom: 3rem;
}

.features-section h2 {
  font-size: 2rem;
  margin-bottom: 2rem;
  color: #333;
}

.features-section ul {
  list-style: none;
  padding: 0;
  max-width: 600px;
  margin: 0 auto;
}

.features-section li {
  background: white;
  padding: 1rem;
  margin-bottom: 1rem;
  border-radius: 8px;
  box-shadow: 0 2px 4px rgba(0,0,0,0.1);
  border: 1px solid #e9ecef;
}

.pricing-section {
  text-align: center;
  margin-bottom: 3rem;
}

.pricing-section h2 {
  font-size: 2rem;
  margin-bottom: 2rem;
  color: #333;
}

.pricing-info {
  background: white;
  padding: 2rem;
  border-radius: 8px;
  box-shadow: 0 2px 4px rgba(0,0,0,0.1);
  border: 1px solid #e9ecef;
  max-width: 600px;
  margin: 0 auto;
  text-align: center;
}

.pricing-info p {
  font-size: 1.1rem;
  color: #666;
  margin-bottom: 1.5rem;
}

.timeline-section {
  text-align: center;
  margin-bottom: 3rem;
}
```

```css
.timeline-section h2 {
  font-size: 2rem;
  margin-bottom: 2rem;
  color: #333;
}

.timeline {
  max-width: 600px;
  margin: 0 auto;
  border-left: 3px solid #ff6b35;
  padding-left: 1rem;
}

.timeline-item {
  position: relative;
  margin-bottom: 2rem;
  padding-left: 1rem;
}

.timeline-item h3 {
  font-size: 1.3rem;
  margin-bottom: 0.3rem;
  color: #ff6b35;
}

.timeline-item p {
  font-size: 1rem;
  color: #666;
  margin: 0;
}

.timeline-item::before {
  content: '';
  position: absolute;
  left: -11px;
  top: 8px;
  width: 15px;
  height: 15px;
  background: #ff6b35;
  border-radius: 50%;
  border: 3px solid white;
  box-shadow: 0 0 0 2px #ff6b35;
}

.budget-breakdown-section {
  text-align: center;
  margin-bottom: 3rem;
}

.budget-breakdown-section h2 {
  font-size: 2rem;
  margin-bottom: 2rem;
  color: #333;
}

.budget-table {
  max-width: 800px;
  margin: 0 auto;
  background: white;
  border-radius: 8px;
  box-shadow: 0 2px 4px rgba(0,0,0,0.1);
  border: 1px solid #e9ecef;
  overflow: hidden;
}

.budget-table table {
  width: 100%;
  border-collapse: collapse;
}

.budget-table th,
.budget-table td {
  padding: 1rem;
  text-align: left;
  border-bottom: 1px solid #e9ecef;
}

.budget-table th {
  background: #f8f9fa;
  font-weight: 600;
  color: #333;
}

.budget-table td {
  color: #666;
}

.budget-table tr:last-child td {
  font-weight: 600;
  color: #ff6b35;
}

.products-content {
  text-align: center;
  max-width: 800px;
}

.products-title {
  font-size: 3.5rem;
  font-weight: bold;
  margin-bottom: 1rem;
  color: #333;
  line-height: 1.2;
}

.products-subtitle {
  font-size: 1.3rem;
  margin-bottom: 2rem;
  line-height: 1.6;
  color: #666;
}

.products-grid {
  display: grid;
  grid-template-columns: repeat(auto-fit, minmax(250px, 1fr));
  gap: 2rem;
  width: 100%;
}

.product-card {
  background: white;
  border: 1px solid #e9ecef;
  border-radius: 8px;
  padding: 2rem;
  text-align: center;
  text-decoration: none;
  color: #333;
  transition: transform 0.3s, box-shadow 0.3s;
  box-shadow: 0 2px 4px rgba(0,0,0,0.1);
}

.product-card:hover {
  transform: translateY(-5px);
  box-shadow: 0 4px 8px rgba(0,0,0,0.15);
}

.product-card h3 {
  font-size: 1.5rem;
  margin-bottom: 0.5rem;
  color: #ff6b35;
}

.product-card p {
  font-size: 1rem;
  color: #666;
  line-height: 1.5;
}

/* Mobile Responsiveness */
@media (max-width: 768px) {
  .products {
    padding: 2rem 1rem;
  }

  .products-title {
    font-size: 2.5rem;
  }

  .products-subtitle {
    font-size: 1.1rem;
  }

  .products-grid {
    grid-template-columns: 1fr;
    gap: 1.5rem;
  }

  .product-card {
    padding: 1.5rem;
  }
}

@media (max-width: 480px) {
  .products {
    padding: 1.5rem 1rem;
  }

  .products-title {
    font-size: 2rem;
  }

  .products-subtitle {
    font-size: 1rem;
  }
}
```

/frontend/components/EventList.js

```
<div className="event-list">
      <header className="page-header">
        <video playsInline autoPlay loop muted className="header-video">
          <source src={require('../videos/banner.mp4')} type="video/mp4" />
        </video>
        <div className="header-left">
          <h1>Plan Events - Event Management</h1>
          <p>Explore upcoming events and discover the latest in data</p>
          <button className="btn-view-calendar">View full calendar</button>
        </div>
      </header>
      <div className="tabs">
        <div className="tab-buttons">
          <button
            className={activeTab === 'featured' ? 'active' : ''}
            onClick={() => setActiveTab('featured')}
          >
            Featured
          </button>
          <button
            className={activeTab === 'upcoming' ? 'active' : ''}
            onClick={() => setActiveTab('upcoming')}
          >
            Upcoming
          </button>
          <button
            className={activeTab === 'demand' ? 'active' : ''}
            onClick={() => setActiveTab('demand')}
          >
            Demand
          </button>
        </div>
        <div className="header-right">
          {isAdmin && <Link to="/events/new" className="btn-primary">+ Add Event</Link>}
        </div>
      </div>
      <div className="event-list-container">
        <aside className="sidebars">
          <div className="search-box">
            <input
              type="text"
              placeholder="Search events..."
              value={searchTerm}
              onChange={(e) => setSearchTerm(e.target.value)}
            />
          </div>
          <div className="filter">
            <label>Event Type:</label>
            <select value={selectedCategory} onChange={(e) =>
setSelectedCategory(e.target.value)}>
              <option value="">All</option>
              {categories.map(cat => <option key={cat} value={cat}>{cat}</option>)}
            </select>
          </div>
          <div className="filter">
```

```
            <label>Region:</label>
            <select value={selectedRegion} onChange={(e) =>
setSelectedRegion(e.target.value)}>
              <option value="">All</option>
              {regions.map(reg => <option key={reg} value={reg}>{reg}</option>)}
            </select>
          </div>
        </aside>
        <main className="main-content">
          <div className="event-grid">
            {filteredEvents.map(event => (
              <div key={event.id} className="event-card">
                {event.banner && <img src={event.banner.startsWith('http') ? event.banner :
`http://localhost:5000/uploads/${event.banner}`} alt="Event Banner" className="event-banner"
/>}
                <h3>{event.title}</h3>
                <p>{event.description}</p>
                <p><strong>Date:</strong> {event.date}</p>
                <p><strong>Time:</strong> {event.time}</p>
                <p><strong>Location:</strong> {event.location}</p>
                <p><strong>Category:</strong> {event.category}</p>
                <p><strong>Organizer:</strong> {event.organizer}</p>
                <Link to={`/events/${event.id}`}>View Details</Link>
              </div>
            ))}
          </div>
        </main>
      </div>
    </div>
```

/frontend/components/EventLists.css

```
.event-list {
  width: 100%;
  box-sizing: border-box;
}

.page-header {
  position: relative;
  height: 400px;
  overflow: hidden;
  margin-bottom: 30px;
  max-width: 1500px;
  width: 100%;
}

.header-video {
  position: absolute;
  top: 0;
  left: 0;
  width: 100%;
  height: 100%;
  object-fit: cover;
  z-index: 0;
}

.header-left {
  position: absolute;
  top: 50%;
  left: 50px;
```

```css
  transform: translateY(-50%);
  z-index: 1;
  background: transparent;
  padding: 20px;
  border-radius: 8px;
  color: white;
  text-align: left;
}

.page-header h1 {
  font-size: 3.5rem;
  margin: 0 0 10px 0;
  font-weight: 700;
  text-shadow: 2px 2px 4px rgba(0,0,0,0.5);
}

.page-header p {
  font-size: 1.4rem;
  margin: 0 0 10px 0;
  text-shadow: 1px 1px 3px rgba(0,0,0,0.4);
}

.btn-view-calendar {
  background: rgba(255, 107, 53, 0.426);
  color: white;
  border: 1px solid white;
  padding: 12px 24px;
  font-size: 1rem;
  border-radius: 4px;
  cursor: pointer;
  transition: background-color 0.3s ease, border-color 0.3s ease;
  box-shadow: 0 4px 6px rgba(0,0,0,0.3);
}

.btn-view-calendar:hover {
  background: rgba(255, 255, 255, 0.3);
  border-color: rgba(255, 255, 255, 0.8);
}

@media (max-width: 768px) {
  .header-left {
    left: 20px;
    width: calc(100% - 40px);
    text-align: center;
  }

  .page-header h1 {
    font-size: 2.5rem;
  }

  .page-header p {
    font-size: 1.2rem;
  }
}

.event-list-container {
  display: flex;
  gap: 20px;
}

.sidebars {
  width: 250px;
  background-color: #f9f9f9;
  padding: 20px;
  border-radius: 8px;
  box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
  margin-top: 30px;
}

.search-box input {
  width: 100%;
  padding: 10px;
  border: 1px solid #ccc;
  border-radius: 4px;
  font-size: 1rem;
  margin-bottom: 20px;
}

.filter {
  margin-bottom: 15px;
}

.filter label {
  display: block;
  margin-bottom: 5px;
  font-weight: bold;
}

.filter select {
  width: 100%;
  padding: 8px;
  border: 1px solid #ccc;
  border-radius: 4px;
  font-size: 1rem;
}

.main-content {
  flex: 1;
  padding: 20px;
}

.tabs {
  display: flex;
  justify-content: space-between;
  margin-left: 80px;
  margin-right: 70px;
  margin-top: 20px;
  align-items: center;
}

.tab-buttons {
  display: flex;
}

.tabs button {
  padding: 10px 20px;
  border: none;
  background-color: #e0e0e0;
  cursor: pointer;
  font-size: 1rem;
  border-radius: 4px 4px 0 0;
  margin-right: 5px;
  transition: background-color 0.3s;
}

.tabs button.active {
  background-color: #ff6b35;
  color: white;
}

.tabs button:hover {
  background-color: #ccc;
}

.event-grid {
  display: grid;
  grid-template-columns: repeat(2, 1fr);
  grid-template-rows: repeat(2, 1fr);
  width: 100%;
  gap: 20px;
}

.event-card {
  background-color: white;
  border: 1px solid #ddd;
  border-radius: 8px;
  padding: 20px;
  box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
  transition: transform 0.3s, box-shadow 0.3s;
}

.event-card:hover {
  transform: translateY(-5px);
  box-shadow: 0 4px 8px rgba(0, 0, 0, 0.2);
}

.event-card img {
  width: 100%;
  height: 150px;
  object-fit: cover;
  border-radius: 4px;
  margin-bottom: 15px;
}

.event-card h3 {
  margin-top: 0;
  color: #333;
}

.event-card p {
  margin: 5px 0;
  color: #666;
}

.event-card a {
  display: inline-block;
  margin-top: 10px;
  color: #ff6b35;
  text-decoration: none;
}

.event-card a:hover {
  text-decoration: underline;
```

```
}

@media (max-width: 768px) {
  .event-list-container {
    flex-direction: column;
  }
  .sidebars {
    width: 100%;
  }
  .tabs {
    flex-wrap: wrap;
  }
  .tabs button {
    flex: 1;
    margin-right: 0;
    margin-bottom: 5px;
  }
}
```

/frontend/components/About.js

```
<section className="about-container">
      <div className="about-content">
        <h1>Plan Events is the premier event management platform</h1>
        <p>
          With Plan Events, you can effortlessly plan, manage, and scale your events. Our
platform provides a unified foundation for all your event needs, combining powerful tools with
an intuitive interface.
        </p>
        <p>
          Founded with a vision to simplify event management, Plan Events empowers organizers
to create unforgettable experiences with ease and efficiency.
        </p>
        <p>
          Established in 2025 by a person of seasoned event professionals and tech innovators,
Plan Events was born from the need for a more streamlined approach to handling everything from
small gatherings to large-scale conferences.
        </p>
        <div className="mission-section">
          <h2>Our Mission</h2>
          <p>
            At Plan Events, our mission is to democratize event planning by providing
accessible, powerful tools that enable anyone—from individuals to corporations—to host
successful events without the hassle.
          </p>
        </div>
        <div className="values-section">
          <h2>Our Core Values</h2>
          <ul>
            <li><strong>Innovation:</strong> Continuously evolving our platform with cutting-
edge features to stay ahead in the event industry.</li>
            <li><strong>Reliability:</strong> Ensuring seamless performance so you can focus
on what matters—creating memorable experiences.</li>
            <li><strong>User-Centric Design:</strong> Building intuitive interfaces that make
event management enjoyable and efficient for all users.</li>
          </ul>
        </div>
      </div>
      <div className="about-image">
        <img src={collageImage} alt="Plan Events collage" />
      </div>
    </section>
```

/frontend/components/About.css

```
.about-container {
  display: flex;
  align-items: center;
  justify-content: space-between;
  max-width: 1100px;
  margin: 3rem auto;
  padding: 0 1rem;
  gap: 2rem;
}

.about-content {
  flex: 1;
  max-width: 600px;
}

.about-content h1 {
  font-size: 2.8rem;
  font-weight: 700;
  margin-bottom: 1.5rem;
  color: #222;
}

.about-content p {
  font-size: 1.2rem;
  line-height: 1.6;
  margin-bottom: 1.2rem;
  color: #444;
}

.btn-primary {
  background-color: #ff6b35;
  color: white;
  padding: 0.75rem 1.8rem;
  border: none;
  border-radius: 4px;
  font-weight: 600;
  cursor: pointer;
  transition: background-color 0.3s;
}

.btn-primary:hover {
  background-color: #e55a2b;
}

.about-image {
  flex: 1;
  display: flex;
  justify-content: center;
}

.about-image img {
  max-width: 100%;
  height: auto;
  border-radius: 12px;
}
```

/frontend/components/Dashboard.js

```
<div className="dashboard">
      <nav className="sidebar">
        <ul>
          <li><Link to="/dashboard"><FontAwesomeIcon icon={faHome}/> Home</Link></li>
          <li><Link to="/profile"><FontAwesomeIcon icon={faPerson}/>Profile</Link></li>
          <li><Link to="/events"><FontAwesomeIcon icon={faCalendar}/> Upcoming 
Events</Link></li>
          <li><Link to="/registrations"><FontAwesomeIcon icon={faTicket}/> Tickets</Link></li>
          <li><Link to="/ticketing/attendance"><FontAwesomeIcon icon={faClipboard} /> 
Attendance</Link></li>
          <li><Link to="/recents"><FontAwesomeIcon icon={faHistory} /> Recents</Link></li>
          <li><Link to="/settings"><FontAwesomeIcon icon={faGears} /> Settings</Link></li>
        </ul>
      </nav>

      {/* Main Content */}
      <div className="main-content">
        {/* Header */}
        <header className="headers">
          <div className="headers-left">
            <h1 style={{ textAlign: 'center' }}>Welcome to User Dashboard</h1>
          </div>
          <div className="headers-right">
            <input
              type="text"
              className="search-input"
              placeholder="Search, visualize, and more"
              value={searchTerm}
              onChange={(e) => setSearchTerm(e.target.value)}
            />
          </div>
        </header>
        <div className="available-events-card">
          <h3>Available Events</h3>
          <p>{availableEvents.length} events available to register for</p>
          <Link to="/events" className="view-available">View Available Events</Link>
        </div>

        <section className="charts-section">
          <div className="chart-container">
            <h3>Total Events</h3>
            <ResponsiveContainer width="100%" height={200}>
              <PieChart>
                <Pie
                  data={[
                    { name: 'Registered', value: registeredEvents.length },
                    { name: 'Available', value: availableEvents.length }
                  ]}
                  cx="50%"
                  cy="50%"
                  outerRadius={80}
                  fill="#8884d8"
                  dataKey="value"
                >
                  <Cell key="registered" fill="#007BFF" />
                  <Cell key="available" fill="#e9ecef" />
                </Pie>
                <text x="50%" y="50%" textAnchor="middle" dominantBaseline="middle" fill="#000"
fontSize="20" fontWeight="bold">
                  {totalPercentage}%
                </text>
              </PieChart>
            </ResponsiveContainer>
          </div>
          <div className="chart-container">
            <h3>Events Attended</h3>
            {registeredEvents.length > 0 ? (
              <ResponsiveContainer width="100%" height={200}>
              <PieChart>
                <Pie
                  data={[
                    { name: 'Attended', value: attendedCount },
                    { name: 'Not Attended', value: registeredEvents.length - attendedCount }
                  ]}
                  cx="50%"
                  cy="50%"
                  outerRadius={80}
                  fill="#8884d8"
                  dataKey="value"
                >
                  <Cell key="attended" fill="#28a745" />
                  <Cell key="not-attended" fill="#dc3545" />
                </Pie>
                <text x="50%" y="50%" textAnchor="middle" dominantBaseline="middle" fill="#000"
fontSize="20" fontWeight="bold">
                  {attendedPercentage}%
                </text>
              </PieChart>
              </ResponsiveContainer>
            ) : (
              <p>No events registered yet.</p>
            )}
          </div>
        </section>
      </div>
    </div>
```

/frontend/components/Dashboard.css

```
.dashboard {
  display: flex;
  min-height: 100vh;
  background-color: #ffffff;
  font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
}

.sidebar {
  position: relative;
  left: 0;
  top: 0;
  width: 200px;
  height: 100vh;
  background-color: #1C242A;
  padding: 20px 0;
  box-shadow: 2px 0 5px rgba(0, 0, 0, 0.5);
  z-index: 1000;
  list-style-type: none;
  padding: 0;
  margin: 0 auto;
}

.sidebar ul {
  list-style: none;
  padding: 0;
  margin: 0;
}

.sidebar li {
  margin: 15px 0;
  cursor: pointer;
  padding: 10px 15px;
  border-radius: 5px;
  color: white;
  transition: background-color 0.3s;
  text-align: left;
  align-items: center;
  justify-content: flex-start;
  display: flex;
  width: 100%;
  border-bottom: none;
  border-radius: 10px 10px !important;
}

.sidebar li i {
  margin-right: 10px;
  font-size: 16px;
}

.sidebar li:hover {
  background: linear-gradient(135deg, #333 0%, #555 100%);
  transition: all 0.3s ease;
}

.sidebar a {
  color: white;
  text-decoration: none;
  font-size: 16px;
  display: block;
}

.sidebar a:hover {
  color: #0066CC;
}

.sidebar button {
  background: none;
  border: none;
  color: white;
  padding: 0;
  width: auto;
  text-align: center;
  cursor: pointer;
  font-size: 16px;
}

.sidebar button:hover {
  color: #0066CC;
}
```

```css
.sidebar h2 {
  margin-bottom: 20px;
  text-align: center;
  color: white;
}

.main-content {
  flex: 1;
  display: flex;
  flex-direction: column;
  padding: 20px;
  margin: 0 auto;
}

.headers {
  margin-bottom: 20px;
}

.headers-left {
  align-items: center;
}

.headers-left h1 {
  margin: 0;
  font-size: 2rem;
  color: #333;
}

.headers-right {
  text-align: center;
}

.headers-right input {
  border-radius: 50px;
}

.search-input {
  width: 75%;
}

.searching-container {
  position: relative;
  background-color: #fff;
  border-radius: 5px;
  padding: 10px 10px 10px 10px;
  border: none;
}

.searching-container:focus-within {
  border: none;
  outline: none;
  box-shadow: none;
}

.searching-container input:focus {
  border: none;
  outline: none;
  box-shadow: none;
}
```

```css
.searchbar {
  background-color: transparent;
  color: #333;
  font-size: 1rem;
  outline: none;
  width: 100%;
}

.profile-icon {
  font-size: 2rem;
  cursor: pointer;
  color: white;
}

.filter-cards {
  display: flex;
  flex-direction: row;
  justify-content: center;
  align-items: center;
  gap: 15px;
  margin: 20px 0;
  flex-wrap: wrap;
}

.filter-card {
  display: flex;
  align-items: center;
  padding: 20px 25px;
  border-radius: 25px;
  color: white;
  text-decoration: none;
  font-weight: bold;
  transition: transform 0.2s ease;
  min-width: 150px;
  justify-content: center;
  box-shadow: 0 2px 10px rgba(0, 0, 0, 0.1);
  border: none;
  cursor: pointer;
}

.filter-card:hover {
  transform: scale(1.05);
}

.blue-card {
  background-color: #007BFF;
}

.purple-card {
  background-color: #6F42C1;
}

.teal-card {
  background-color: #20C997;
}

.icon {
  font-size: 1.8rem;
  margin-right: 8px;
}
```

```css
.label {
  font-size: 1rem;
}

.badge {
  background: rgba(255, 255, 255, 0.3);
  border-radius: 10px;
  padding: 2px 6px;
  font-size: 0.8rem;
  margin-left: 5px;
}

.recent-events {
  margin-top: 30px;
  background: #f8f9fa;
  padding: 20px;
  border-radius: 10px;
}

.recent-events h3 {
  color: #333;
  margin-bottom: 15px;
}

.recent-events ul {
  list-style: none;
  padding: 0;
  margin-bottom: 15px;
}

.recent-events li {
  padding: 8px;
  background: white;
  margin: 5px 0;
  border-radius: 5px;
  display: flex;
  justify-content: space-between;
  align-items: center;
}

.status {
  font-size: 0.8rem;
  padding: 2px 8px;
  border-radius: 12px;
}

.status.checked-in {
  background: #d4edda;
  color: #155724;
}

.status.pending {
  background: #fff3cd;
  color: #856404;
}

.view-all {
  background: #007BFF;
  color: white;
  padding: 10px 20px;
  border-radius: 5px;
  text-decoration: none;
  display: inline-block;
  margin-top: 10px;
}

.view-all:hover {
  background: #0056b3;
}

.available-events-card {
  background: #e9ecef;
  padding: 20px;
  border-radius: 10px;
  text-align: center;
  border: 1px solid #dee2e6;
  margin-top: 20px;
}

.available-events-card h3 {
  color: #333;
  margin-bottom: 10px;
}

.available-events-card p {
  color: #6c757d;
  margin-bottom: 15px;
  font-size: 1.1rem;
}

.view-available {
  background: #007BFF;
  color: white;
  padding: 10px 20px;
  border-radius: 5px;
  text-decoration: none;
  display: inline-block;
}

.view-available:hover {
  background: #0056b3;
}

.loading {
  text-align: center;
  font-size: 1.5rem;
  color: white;
}

/* Charts Section */
.charts-section {
  display: grid;
  grid-template-columns: 1fr 1fr;
  gap: 20px;
  margin: 20px 0;
}

.chart-container {
  background: #f8f9fa;
  padding: 20px;
  border-radius: 10px;
  text-align: center;
  border: 1px solid #dee2e6;
}

.chart-container h3 {
  color: #333;
  margin-bottom: 15px;
}

/* My Registrations Section */
.registrations-section {
  margin-top: 30px;
}

.registrations-section h3 {
  color: #333;
  margin-bottom: 20px;
  text-align: center;
}

.registrations-list {
  display: grid;
  grid-template-columns: repeat(auto-fill, minmax(300px, 1fr));
  gap: 20px;
}

.registration-item {
  background: #f8f9fa;
  padding: 20px;
  border-radius: 10px;
  border: 1px solid #dee2e6;
  display: flex;
  flex-direction: column;
  justify-content: space-between;
}

.event-info h4 {
  color: #333;
  margin-bottom: 10px;
}

.event-info p {
  margin: 5px 0;
  color: #6c757d;
}

.event-actions {
  display: flex;
  gap: 10px;
  margin-top: 15px;
}

.view-details-btn {
  background: #007BFF;
  color: white;
  padding: 8px 15px;
  border-radius: 5px;
  text-decoration: none;
  display: inline-block;
  text-align: center;
  flex: 1;
}
```

```css
.view-details-btn:hover {
  background: #0056b3;
}

.checkin-btn {
  background: #28a745;
  color: white;
  padding: 8px 15px;
  border: none;
  border-radius: 5px;
  cursor: pointer;
  flex: 1;
}

.checkin-btn:hover {
  background: #218838;
}

/* Responsive */
@media (max-width: 768px) {
  .sidebar {
    width: 100%;
    height: auto;
    position: relative;
  }

  .main-content {
    margin-left: 0;
  }

  .sidebar li {
    width: 80%;
  }

  .charts-section {
    grid-template-columns: 1fr;
  }

  .registrations-list {
    grid-template-columns: 1fr;
  }
}
```

/frontend/components/AdminDashboard.js

```jsx
<div className="admin-dashboard-new">
      <div className="admin-sidebar">
        <ul className="admin-sidebar__menu">
          <li className="admin-sidebar__group">
            <Link to="/admin/dashboard" className="admin-sidebar__item">
              <FontAwesomeIcon icon={faHome} className="admin-sidebar__icon" />
              <span className="admin-sidebar__text">Dashboard</span>
            </Link>
          </li>

          <li className="admin-sidebar__group">
            <Link to="/admin/users" className="admin-sidebar__item">
              <FontAwesomeIcon icon={faUser} className="admin-sidebar__icon" />
              <span className="admin-sidebar__text">Users</span>
            </Link>
          </li>

          <li className="admin-sidebar__group">
            <Link to="/admin/events" className="admin-sidebar__item">
              <FontAwesomeIcon icon={faCalendar} className="admin-sidebar__icon" />
              <span className="admin-sidebar__text">Events</span>
            </Link>
          </li>

          <li className="admin-sidebar__group">
            <Link to="/admin/tickets" className="admin-sidebar__item">
              <FontAwesomeIcon icon={faTicket} className="admin-sidebar__icon" />
              <span className="admin-sidebar__text">Tickets</span>
            </Link>
          </li>

          <li className="admin-sidebar__group">
            <Link to="/admin/analytics" className="admin-sidebar__item">
              <FontAwesomeIcon icon={faChartSimple} className="admin-sidebar__icon" />
              <span className="admin-sidebar__text">Analytics</span>
            </Link>
          </li>

          <li className="admin-sidebar__group">
            <Link to="/admin/settings" className="admin-sidebar__item">
              <FontAwesomeIcon icon={faCog} className="admin-sidebar__icon" />
              <span className="admin-sidebar__text">Settings</span>
            </Link>
          </li>
        </ul>
      </div>

      <div className="main-content">
        <header className="dashboard-header">
          <h1>Admin Dashboard Overview</h1>
          <input
            type="text"
            className="search-box"
            placeholder="Search your query"
            value={searchQuery}
            onChange={handleSearchChange}
          />
        </header>
        <div className="stats-section">
          <div className="stat-card">
            <h3>Total Events</h3>
            <p>{stats.totalEvents}</p>
          </div>
          <div className="stat-card">
            <h3>Total Users</h3>
            <p>{stats.totalUsers}</p>
          </div>
          <div className="stat-card">
            <h3>Upcoming Events</h3>
            <p>{stats.totalUpcomingEvents}</p>
          </div>
          <div className="stat-card">
            <h3>Total Tickets</h3>
            <p>{stats.totalTickets}</p>
          </div>
        </div>
        <div className="recent-activities">
          <h2>Recent Activities</h2>
          <table className="activities-table">
            <thead>
              <tr>
                <th>Action</th>
                <th>Details</th>
                <th>Timestamp</th>
              </tr>
            </thead>
            <tbody>
              {recentActivities.map(activity => (
                <tr key={activity.id}>
                  <td>{activity.action}</td>
                  <td>{activity.details}</td>
                  <td>{activity.timestamp}</td>
                </tr>
              ))}
            </tbody>
          </table>
        </div>
        <div className="audit-logs">
          <h2>Audit Logs</h2>
          <input
            type="text"
            className="search-box"
            placeholder="Audit Logs by"
            value={auditLogsSearchQuery}
            onChange={(e) => setAuditLogsSearchQuery(e.target.value)}
          />
          <table className="audit-table">
            <thead>
              <tr>
                <th>Action</th>
                <th>Details</th>
                <th>Admin User</th>
                <th>Timestamp</th>
              </tr>
            </thead>
            <tbody>
              {(() => {
                const filteredLogs = auditLogs.filter(log =>
                  log.action.toLowerCase().includes(auditLogsSearchQuery.toLowerCase()) ||
                  log.details.toLowerCase().includes(auditLogsSearchQuery.toLowerCase())
                );
                if (filteredLogs.length === 0) {
                  return (
                    <tr>
                      <td colSpan="4" style={{ textAlign: 'center', padding: '20px', color:
'#666' }}>
                        No audit logs available. Perform an admin action to generate logs.
                      </td>
                    </tr>
                  );
                }
                return filteredLogs.map(log => (
                  <tr key={log.id}>
                    <td>{log.action}</td>
                    <td>{log.details}</td>
                    <td>{log.admin_user}</td>
                    <td>{new Date(log.timestamp).toLocaleString()}</td>
                  </tr>
                ));
              })()}
            </tbody>
          </table>
        </div>
      </div>
    </div>
```

/frontend/components/SpeakerDashboard.js

```
  <div className="speaker-dashboard">
    <div className="speaker-sidebar">
      <ul className="speaker-sidebar__menu">
        <li className="speaker-sidebar__group">
          <Link to="/speaker/dashboard" className="speaker-sidebar__item active">
            <FontAwesomeIcon icon={faHome} className="speaker-sidebar__icon" />
            <span className="speaker-sidebar__text">Dashboard</span>
          </Link>
        </li>
        <li className="speaker-sidebar__group">
          <Link to="/speaker/profile" className="speaker-sidebar__item">
            <FontAwesomeIcon icon={faUser} className="speaker-sidebar__icon" />
            <span className="speaker-sidebar__text">Profile</span>
          </Link>
        </li>
        <li className="speaker-sidebar__group">
          <Link to="/speaker/talks" className="speaker-sidebar__item">
            <FontAwesomeIcon icon={faCalendar} className="speaker-sidebar__icon" />
            <span className="speaker-sidebar__text">Upcoming Talks</span>
          </Link>
        </li>
        <li className="speaker-sidebar__group">
          <Link to="/speaker/past-events" className="speaker-sidebar__item">
            <FontAwesomeIcon icon={faHistory} className="speaker-sidebar__icon" />
            <span className="speaker-sidebar__text">Past Events</span>
          </Link>
        </li>
        <li className="speaker-sidebar__group">
          <Link to="/speaker/settings" className="speaker-sidebar__item">
            <FontAwesomeIcon icon={faCog} className="speaker-sidebar__icon" />
            <span className="speaker-sidebar__text">Settings</span>
          </Link>
        </li>
      </ul>
    </div>

    {/* Main Content */}
    <div className="main-content">
      {/* Header */}
      <header className="headers">
        <div className="headers-left">
          <h1 style={{ textAlign: 'center' }}>Welcome to Speaker Dashboard</h1>
        </div>
        <div className="headers-right">
          <input
            type="text"
            className="search-input"
```

```
              placeholder="Search talks, events, and more"
              value={searchTerm}
              onChange={(e) => setSearchTerm(e.target.value)}
            />
          </div>
        </header>

        <div className="stats-section">
          <div className="stat-card">
            <h3>Total Talks</h3>
            <p>{totalEvents}</p>
          </div>
          <div className="stat-card">
            <h3>Upcoming Talks</h3>
            <p>{upcomingEvents.length}</p>
          </div>
          <div className="stat-card">
            <h3>Past Talks</h3>
            <p>{pastEvents.length}</p>
          </div>
        </div>

        <section className="charts-section">
          <div className="chart-container">
            <h3>Talks Overview</h3>
            <ResponsiveContainer width="100%" height={200}>
              <PieChart>
                <Pie
                  data={[
                    { name: 'Upcoming', value: upcomingEvents.length },
                    { name: 'Past', value: pastEvents.length }
                  ]}
                  cx="50%"
                  cy="50%"
                  outerRadius={80}
                  fill="#8884d8"
                  dataKey="value"
                >
                  <Cell key="upcoming" fill="#007BFF" />
                  <Cell key="past" fill="#e9ecef" />
                </Pie>
                <text x="50%" y="50%" textAnchor="middle" dominantBaseline="middle" fill="#000"
fontSize="20" fontWeight="bold">
                  {upcomingPercentage}%
                </text>
              </PieChart>
            </ResponsiveContainer>
          </div>
        </section>

        <div className="speaker-events">
          <h2>Your Talks</h2>
          <table className="events-table">
            <thead>
              <tr>
                <th>Event Title</th>
                <th>Date</th>
                <th>Status</th>
              </tr>
            </thead>
            <tbody>
```

```
              {speakerEvents.length > 0 ? (
                speakerEvents.map(event => (
                  <tr key={event.id}>
                    <td>{event.title}</td>
                    <td>{new Date(event.date).toLocaleDateString()}</td>
                    <td>{new Date(event.date) > new Date() ? 'Upcoming' : 'Past'}</td>
                  </tr>
                ))
              ) : (
                <tr>
                  <td colSpan="3" style={{ textAlign: 'center', padding: '20px', color:
'#666' }}>
                    No talks assigned yet.
                  </td>
                </tr>
              )}
            </tbody>
          </table>
        </div>
      </div>
    </div>
```

/frontend/components/SpeakerDashboard.js

```
.speaker-dashboard {
  display: flex;
  min-height: 100vh;
  background-color: #ffffff;
  font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
}

.speaker-sidebar {
  position: relative;
  left: 0;
  top: 0;
  width: 200px;
  height: 100vh;
  background-color: #1C242A;
  padding: 20px 0;
  box-shadow: 2px 0 5px rgba(0, 0, 0, 0.5);
  z-index: 1000;
  list-style-type: none;
  padding: 0;
  margin: 0 auto;
}

.speaker-sidebar ul {
  list-style: none;
  padding: 0;
  margin: 0;
}

.speaker-sidebar li {
  margin: 15px 0;
  cursor: pointer;
  padding: 10px 15px;
  border-radius: 5px;
  color: white;
  transition: background-color 0.3s;
```

```css
  text-align: left;
  align-items: center;
  justify-content: flex-start;
  display: flex;
  width: 100%;
  border-bottom: none;
  border-radius: 10px 10px !important;
}

.speaker-sidebar li i {
  margin-right: 10px;
  font-size: 16px;
}

.speaker-sidebar li:hover {
  background: linear-gradient(135deg, #333 0%, #555 100%);
  transition: all 0.3s ease;
}

.speaker-sidebar a {
  color: white;
  text-decoration: none;
  font-size: 16px;
  display: block;
}

.speaker-sidebar a:hover {
  color: #0066CC;
}

.speaker-sidebar button {
  background: none;
  border: none;
  color: white;
  padding: 0;
  width: auto;
  text-align: center;
  cursor: pointer;
  font-size: 16px;
}

.speaker-sidebar button:hover {
  color: #0066CC;
}

.speaker-sidebar h2 {
  margin-bottom: 20px;
  text-align: center;
  color: white;
}

.speaker-dashboard .main-content {
  flex: 1;
  display: flex;
  flex-direction: column;
  padding: 20px;
  margin: 0 auto;
}

.speaker-dashboard .headers {
  margin-bottom: 20px;
```

```
}

.speaker-dashboard .headers-left {
  align-items: center;
}

.speaker-dashboard .headers-left h1 {
  margin: 0;
  font-size: 2rem;
  color: #333;
}

.speaker-dashboard .headers-right {
  text-align: center;
}

.speaker-dashboard .headers-right input {
  border-radius: 50px;
}

.speaker-dashboard .search-input {
  width: 75%;
}

.speaker-dashboard .searching-container {
  position: relative;
  background-color: #fff;
  border-radius: 5px;
  padding: 10px 10px 10px 10px;
  border: none;
}

.speaker-dashboard .searching-container:focus-within {
  border: none;
  outline: none;
  box-shadow: none;
}

.speaker-dashboard .searching-container input:focus {
  border: none;
  outline: none;
  box-shadow: none;
}

.speaker-dashboard .searchbar {
  background-color: transparent;
  color: #333;
  font-size: 1rem;
  outline: none;
  width: 100%;
}

.speaker-dashboard .stats-section {
  display: flex;
  justify-content: space-around;
  margin: 20px 0;
  flex-wrap: wrap;
}

.speaker-dashboard .stat-card {
  background: #f8f9fa;
  padding: 20px;
  border-radius: 10px;
  text-align: center;
  border: 1px solid #dee2e6;
  min-width: 150px;
  margin: 10px;
}

.speaker-dashboard .stat-card h3 {
  color: #333;
  margin-bottom: 10px;
}

.speaker-dashboard .stat-card p {
  font-size: 2rem;
  color: #007BFF;
  font-weight: bold;
}

.speaker-dashboard .charts-section {
  display: grid;
  grid-template-columns: 1fr;
  gap: 20px;
  margin: 20px 0;
}

.speaker-dashboard .chart-container {
  background: #f8f9fa;
  padding: 20px;
  border-radius: 10px;
  text-align: center;
  border: 1px solid #dee2e6;
}

.speaker-dashboard .chart-container h3 {
  color: #333;
  margin-bottom: 15px;
}
.speaker-dashboard .speaker-events {
  margin-top: 30px;
  background: #f8f9fa;
  padding: 20px;
  border-radius: 10px;
}

.speaker-dashboard .speaker-events h2 {
  color: #333;
  margin-bottom: 15px;
}

.speaker-dashboard .events-table {
  width: 100%;
  border-collapse: collapse;
}

.speaker-dashboard .events-table th,
.speaker-dashboard .events-table td {
  padding: 10px;
  text-align: left;
  border-bottom: 1px solid #dee2e6;
}
```

```
.speaker-dashboard .events-table th {
  background: #e9ecef;
  color: #333;
}

.speaker-dashboard .loading {
  text-align: center;
  font-size: 1.5rem;
  color: #333;
}

/* Responsive */
@media (max-width: 768px) {
  .speaker-sidebar {
    width: 100%;
    height: auto;
    position: relative;
  }

  .speaker-dashboard .main-content {
    margin-left: 0;
  }

  .speaker-sidebar li {
    width: 80%;
  }

  .speaker-dashboard .charts-section {
    grid-template-columns: 1fr;
  }

  .speaker-dashboard .stats-section {
    flex-direction: column;
  }
}
```

/frontend/components/Login.js

```
<div className="login-page">
              <aside className="login-left">
                   <div className="login-left-inner">
                        <h1>Welcome back</h1>
                        <p className="lead">Sign in to continue managing your events</p>

                        <ul className="benefits">
                             <li><span className="check">✓</span>Quick access to your
dashboards</li>
                             <li><span className="check">✓</span>Manage attendees and
tickets</li>
                             <li><span className="check">✓</span>Secure, enterprise-ready
tools</li>
                        </ul>
                   </div>
              </aside>

              <main className="login-right">
                   <div className="login-card">
                        <div className="brand">Plan Events</div>
```

```
                    <h2>Log in</h2>
                    <p className="muted small">Sign in to your account</p>

                    <form onSubmit={handleSubmit}>
                        <label htmlFor="email" className="input-label">Email or 
Username</label>

                        <input
                            id="email"
                            className="email"
                            placeholder="Email or Username"
                            aria-label="Email or Username"
                            value={email}
                            onChange={(e) => setEmail(e.target.value)}
                        />
                        <label htmlFor="password" className="input-
label">Password</label>
                        <div className="password-wrap">
                            <input
                                id="password"
                                className={`password ${showPassword ? 'shown' : ''}`}
                                type={showPassword ? 'text' : 'password'}
                                placeholder="Password"
                                aria-label="Password"
                                value={password}
                                onChange={(e) => setPassword(e.target.value)}
                            />
                            <button type="button" className="show-toggle" 
onClick={toggleShow} aria-pressed={showPassword} aria-label={showPassword ? 'Hide password' : 
'Show password'}>{showPassword ? 'Hide' : 'Show'}</button>
                        </div>

                        <div className="actions">
                            <label className="remember"><input type="checkbox" /> 
Remember me</label>

                            <a className="forgot" href="/forgot">Forgot?</a>
                        </div>

                        <button className="btn primary full" type="submit" 
disabled={loading}>
                            {loading ? 'Logging in...' : 'Login'}
                        </button>
                    </form>

                    <div className="signup-link">New to Plan Events? <a 
href="/signup">Create an account</a></div>
                </div>
            </main>
        </div>
```

/frontend/src/Login.css

```
.login-page {
    display: flex;
    min-height: 88vh;
    font-family: Inter, system-ui, -apple-system, 'Segoe UI', Roboto, Arial
}

.login-left {
    flex: 1;
    padding: 64px 72px;
```

```css
     display: flex;
     text-align: center;
     align-items: center;
     background: radial-gradient(600px 300px at 25% 30%, rgba(255, 140, 60, 0.04), transparent
25%), linear-gradient(90deg, #2a2740 0%, #213843 100%);
     color: #f1eee9
}

.login-left-inner {
     text-align: left;
     width: 560px;
     padding: 36px 40px;
     border-radius: 8px;
     display: flex;
     flex-direction: column;
     align-items: left;
     justify-content: left;
     gap: 18px;
}

.login-left h1 {
     font-size: 36px;
     margin: 0 0 12px;
     color: #fff9f6
}

.login-left .lead {
     color: #e6dbd3;
     margin-bottom: 22px
}

.benefits {
     list-style: none;
     padding: 0;
     margin: 0;
     display: flex;
     flex-direction: column;
     gap: 12px;
     align-items: flex-start;
}

.benefits li {
     display: flex;
     align-items: center;
     justify-content: center;
     gap: 12px;
     margin: 0;
}

.benefits .check {
     display: inline-flex;
     align-items: center;
     justify-content: center;
     width: 28px;
     height: 28px;
     background: #ff8a34;
     color: white;
     border-radius: 6px;
     font-weight: 700
}
```

```
.login-right {
    width: 420px;
    display: flex;
    align-items: center;
    justify-content: center;
    background: linear-gradient(180deg, rgba(0, 0, 0, 0.45), rgba(0, 0, 0, 0.55));
    padding: 48px
}

.login-card {
    width: 100%;
    background: linear-gradient(180deg, #0b0b0b 0%, #081012 100%);
    border-radius: 12px;
    padding: 28px;
    box-shadow: 0 24px 80px rgba(3, 8, 12, 0.75);
    color: #f0eae3
}

.login-card .brand {
    font-weight: 700;
    color: #ffb386;
    margin-bottom: 6px;
    text-transform: lowercase
}

.login-card h2 {
    margin: 6px 0 4px 0;
    text-align: center;
}

.muted {
    color: #c9bfb6
}

.small {
    font-size: 13px
}

.input-label {
    display: block;
    margin-bottom: 4px;
    color: #c9bfb6;
    font-size: 13px;
}

.email,
.password {
    width: 100%;
    padding: 10px 12px;
    height: 44px;
    border-radius: 8px;
    border: 1px solid rgba(255, 255, 255, 0.04);
    background: rgba(255, 255, 255, 0.02);
    color: white;
    margin-top: 8px;
    box-sizing: border-box;
}

.password-wrap {
    position: relative;
    display: flex;
    align-items: center;
}

.password-wrap .password {
    padding-right: 72px;
}

.password.shown {
    color: white;
}

.show-toggle {
    position: absolute;
    right: 12px;
    top: 30px;
    transform: translateY(-50%);
    height: 28px;
    padding: 0 8px;
    display: inline-flex;
    align-items: center;
    justify-content: center;
    background: rgba(255, 179, 134, 0.06);
    border: 1px solid rgba(255, 179, 134, 0.12);
    color: #ffb386;
    border-radius: 6px;
    cursor: pointer;
    font-size: 13px;
    white-space: nowrap;
}

.show-toggle:focus {
    outline: 2px solid rgba(255, 179, 134, 0.25);
}

.actions {
    display: flex;
    align-items: center;
    justify-content: space-between;
    margin-top: 10px
}

.remember {
    color: #c9bfb6
}

.forgot {
    color: #ffb386
}

.btn {
    padding: 10px 12px;
    border-radius: 8px;
    border: 1px solid rgba(255, 255, 255, 0.04);
    background: rgba(255, 255, 255, 0.02);
    color: inherit;
    cursor: pointer;
    text-align: center;
    display: flex;
    align-items: center;
    justify-content: center;
    gap: 10px
```

```
}

.btn.primary {
     background: #ff6b2b;
     color: white;
     border: none;
}

.full {
     width: 100%;
     margin: 12px auto;
}

.signup-link {
     margin-top: 14px;
     color: #c9bfb6;
     text-align: center;
     font-size: 14px;
}

.signup-link a {
     color: #ffb386
}

.username:focus,
.password:focus,
.password.shown:focus {
     border: 1px solid var(--accent) !important;
     outline: none !important;
     box-shadow: none !important;
}

@media(max-width:920px) {
     .login-page {
          flex-direction: column
     }

     .login-left {
          display: none
     }

     .login-right {
          width: 100%;
          padding: 20px
     }
}
```

/frontend/src/SignUp.js

```
<div className="signup-page">
      <aside className="signup-left">
        <div className="signup-left-inner">
          <h1>Get started with Plan Events</h1>
          <p className="lead">Build AI with all your event data in one place</p>

          <ul className="benefits">
            {perks.map((p, i) => (
              <li key={i}><span className="check">✓</span>{p}</li>
            ))}
          </ul>
        </div>
      </aside>

      <main className="signup-right">
        <div className="signup-card">
          <div className="brand">Plan Events</div>
          <h2>Sign up</h2>
          <p className="muted small">Get started for free</p>

          <form onSubmit={handleSubmit}>
            <div className="form-group">
              <label htmlFor="email" className='label-input'>Email</label>
              <input
                id="email"
                className="email"
                placeholder="Email"
                aria-label="Email"
                value={email}
                onChange={(e) => setEmail(e.target.value)}
              />
            </div>
            <div className="form-group">
              <label htmlFor="username" className='label-input'>Username</label>
              <input
                id="username"
                className="username"
                placeholder="Username"
                aria-label="Username"
                value={username}
                onChange={(e) => setUsername(e.target.value)}
              />
            </div>
            <div className="form-group">
              <label htmlFor="password" className='label-input'>Password</label>
              <div className="password-wrap">
                <input
                  id="password"
                  className={`password ${showPassword ? 'shown' : ''}`}
                  type={showPassword ? 'text' : 'password'}
                  placeholder="Password"
                  aria-label="Password"
                  value={password}
                  onChange={(e) => setPassword(e.target.value)}
                />
                <button type="button" className="show-toggle" onClick={toggleShow} aria-pressed={showPassword} aria-label={showPassword ? 'Hide password' : 'Show password'}>{showPassword ? 'Hide' : 'Show'}</button>
              </div>
            </div>

            <div className="actions">
              <label className="remember">
                <input type="checkbox" /> Remember me
              </label>
            </div>

            <button className="btn primary full" type="submit" disabled={loading}>
              {loading ? 'Signing up...' : 'Sign up'}
            </button>
          </form>
          <div className="footer-link">Already using Plan Events? <a href="/login">Log
in</a></div>
        </div>
      </main>
    </div>
```

/frontend/src/SignUp.css

```
.signup-page {
    --left-bg-1: #2a2740;
    --left-bg-2: #213843;
    --left-highlight: rgba(255, 140, 60, 0.06);
    --card-bg-1: #0b0b0b;
    --card-bg-2: #081012;
    --accent: #ffb386;
    --cta: #ff6b2b;
    --check: #ff8a34;
    display: flex;
    min-height: 88vh;
    font-family: Inter, system-ui, -apple-system, 'Segoe UI', Roboto, 'Helvetica Neue', Arial;
}

.signup-left {
    flex: 1;
    padding: 64px 72px;
    display: flex;
    text-align: center;
    align-items: center;
    background: radial-gradient(600px 300px at 25% 30%, rgba(255, 140, 60, 0.04), transparent
25%), linear-gradient(90deg, #2a2740 0%, #213843 100%);
    color: #f1eee9
}

.signup-right {
    width: 420px;
    display: flex;
    align-items: center;
    justify-content: center;
    background: linear-gradient(180deg, rgba(0, 0, 0, 0.45), rgba(0, 0, 0, 0.55));
    padding: 48px
}

.signup-left-inner {
    max-width: 560px
}

.signup-left h1 {
    font-size: 36px;
    margin: 0 0 12px;
    color: #fff9f6;
}

.signup-left .lead {
    color: #e6dbd3;
    margin-bottom: 22px
}

.benefits {
    list-style: none;
    padding: 0;
```

```css
    margin: 0 0 30px 0;
}

.benefits li {
    display: flex;
    align-items: center;
    gap: 8px;
    margin: 12px 0;
    color: #f3e9e1;
}

.benefits .check {
    display: inline-flex;
    align-items: center;
    justify-content: center;
    width: 28px;
    height: 28px;
    background: var(--check);
    color: white;
    border-radius: 6px;
    font-weight: 700;
}

.trusted {
    margin-top: 36px;
    color: #ccbbb2
}

.trusted-logos {
    display: flex;
    gap: 14px;
    margin-top: 12px;
    align-items: center
}

.trusted-logos img {
    height: 28px;
    filter: brightness(0) invert(1);
    opacity: 0.85
}

.logo-placeholder {
    background: rgba(255, 255, 255, 0.03);
    padding: 6px 10px;
    border-radius: 6px;
    color: #ccbbb2;
    font-size: 12px;
}

.signup-card {
    width: 100%;
    background: linear-gradient(180deg, var(--card-bg-1) 0%, var(--card-bg-2) 100%);
    border-radius: 12px;
    padding: 28px;
    box-shadow: 0 24px 80px rgba(3, 8, 12, 0.75);
    color: #f0eae3;
}

.signup-card .brand {
    font-weight: 700;
    color: var(--accent);
    margin-bottom: 6px;
    text-transform: lowercase;
}

.signup-card h2 {
    margin: 6px 0 4px 0;
    text-align: center;
}

.muted {
    color: #c9bfb6
}

.small {
    font-size: 13px;
}

.tiny {
    font-size: 12px;
}

.auth-buttons {
    display: flex;
    flex-direction: column;
    gap: 10px;
    margin: 14px 0
}

.btn.primary {
    background: var(--cta);
    color: white;
    border: none;
}

.full {
    width: 100%;
    margin-top: 12px
}

.divider {
    display: flex;
    align-items: center;
    margin: 14px 0
}

.divider:before,
.divider:after {
    content: '';
    flex: 1;
    height: 1px;
    background: rgba(255, 255, 255, 0.03)
}

.divider span {
    padding: 0 10px;
    color: #b59a8f
}

.username {
    width: 100%;
    padding: 10px;
```

```css
    border-radius: 8px;
    border: none !important;
    outline: none !important;
    box-shadow: none !important;
    background: rgba(255, 255, 255, 0.02);
    color: inherit;
}

.email {
    width: 100%;
    padding: 10px;
    border-radius: 8px;
    border: none !important;
    outline: none !important;
    box-shadow: none !important;
    background: rgba(255, 255, 255, 0.02);
    color: inherit;
}

.password {
    width: 100%;
    padding: 10px;
    border-radius: 8px;
    border: none !important;
    outline: none !important;
    box-shadow: none !important;
    background: rgba(255, 255, 255, 0.02);
    color: inherit;
}

.username:focus,
.email:focus,
.password:focus,
.password.shown:focus {
    border: 1px solid var(--accent) !important;
    outline: none !important;
    box-shadow: none !important;
}

label {
    display: block;
    margin-bottom: 4px;
    font-size: 14px;
    color: #c9bfb6;
    font-weight: 500;
}

.label-input {
    color: #c9bfb6;
}

.remember {
    color: #c9bfb6;
    font-size: 14px;
}

.footer-link {
    margin-top: 14px;
    color: #c9bfb6;
    text-align: center;
    font-size: 14px;
```

```
}

.footer-link a {
    color: #ffb386
}

.form-group {
    margin-bottom: 12px;
}

.password-wrap {
    display: flex;
    align-items: center;
    gap: 8px;
    margin: 0;
}

.show-toggle {
    background: none;
    border: none;
    color: var(--accent);
    cursor: pointer;
    font-size: 14px;
    padding: 0 4px;
    border-radius: 4px;
    transition: background-color 0.2s;
}

.show-toggle:hover {
    background: rgba(255, 179, 134, 0.1);
}

.forgot {
    color: #ffb386;
    text-decoration: none;
}

.forgot:hover {
    text-decoration: underline;
}

@media(max-width:920px) {
    .signup-page {
        flex-direction: column
    }

    .signup-left {
        display: none
    }

    .signup-right {
        width: 100%;
        padding: 20px
    }
}
```

/frontend/src/AdminLogin.js

```
<div className="admin-login-page">
      <aside className="admin-login-left">
        <div className="admin-login-left-inner">
          <h1>Welcome back, Admin</h1>
          <p className="lead">Sign in to manage the platform</p>

          <ul className="benefits">
            <li><span className="check">✓</span>Access admin dashboard</li>
            <li><span className="check">✓</span>Manage users and events</li>
            <li><span className="check">✓</span>Secure admin tools</li>
          </ul>
        </div>
      </aside>

      <main className="admin-login-right">
        <div className="admin-login-card">
          <div className="brand">Plan Events</div>
          <h2>Admin Log in</h2>
          <p className="muted small">Sign in to your admin account</p>

          <form onSubmit={handleSubmit}>
            <label htmlFor="email" className="input-label">Email or Username</label>
            <input
              id="email"
              className="email"
              placeholder="Email or Username"
              aria-label="Email or Username"
              value={email}
              onChange={(e) => setEmail(e.target.value)}
            />
            <label htmlFor="password" className="input-label">Password</label>
            <div className="password-wrap">
              <input
                id="password"
                className={`password ${showPassword ? 'shown' : ''}`}
                type={showPassword ? 'text' : 'password'}
                placeholder="Password"
                aria-label="Password"
                value={password}
                onChange={(e) => setPassword(e.target.value)}
              />
              <button type="button" className="show-toggle" onClick={toggleShow} aria-
pressed={showPassword} aria-label={showPassword ? 'Hide password' : 'Show
password'}>{showPassword ? 'Hide' : 'Show'}</button>
            </div>

            <div className="actions">
              <label className="remember"><input type="checkbox" /> Remember me</label>
              <a className="forgot" href="/forgot">Forgot?</a>
            </div>

            <button className="btn primary full" type="submit" disabled={loading}>
              {loading ? 'Logging in...' : 'Admin Login'}
            </button>
          </form>

          <div className="signup-link">New admin? <a href="/admin/signup">Create an admin
account</a></div>
```

```
        </div>
      </main>
    </div>
```

/frontend/src/AdminLogin.css

```
.admin-login-page {
     display: flex;
     min-height: 88vh;
     font-family: Inter, system-ui, -apple-system, 'Segoe UI', Roboto, Arial
}

.admin-login-left {
     flex: 1;
     padding: 64px 72px;
     display: flex;
     text-align: center;
     align-items: center;
     background: radial-gradient(600px 300px at 25% 30%, rgba(255, 140, 60, 0.04), transparent
25%), linear-gradient(90deg, #2a2740 0%, #213843 100%);
     color: #f1eee9
}

.admin-login-left-inner {
     text-align: left;
     width: 560px;
     padding: 36px 40px;
     border-radius: 8px;
     display: flex;
     flex-direction: column;
     align-items: left;
     justify-content: left;
     gap: 18px;
}

.admin-login-left h1 {
     font-size: 36px;
     margin: 0 0 12px;
     color: #fff9f6
}

.admin-login-left .lead {
     color: #e6dbd3;
     margin-bottom: 22px
}

.benefits {
     list-style: none;
     padding: 0;
     margin: 0;
     display: flex;
     flex-direction: column;
     gap: 12px;
     align-items: flex-start;
}

.benefits li {
     display: flex;
     align-items: center;
     justify-content: center;
```

```css
    gap: 12px;
    margin: 0;
}

.benefits .check {
    display: inline-flex;
    align-items: center;
    justify-content: center;
    width: 28px;
    height: 28px;
    background: #ff8a34;
    color: white;
    border-radius: 6px;
    font-weight: 700
}

.admin-login-right {
    width: 420px;
    display: flex;
    align-items: center;
    justify-content: center;
    background: linear-gradient(180deg, rgba(0, 0, 0, 0.45), rgba(0, 0, 0, 0.55));
    padding: 48px
}

.admin-login-card {
    width: 100%;
    background: linear-gradient(180deg, #0b0b0b 0%, #081012 100%);
    border-radius: 12px;
    padding: 28px;
    box-shadow: 0 24px 80px rgba(3, 8, 12, 0.75);
    color: #f0eae3
}

.admin-login-card .brand {
    font-weight: 700;
    color: #ffb386;
    margin-bottom: 6px;
    text-transform: lowercase
}

.admin-login-card h2 {
    margin: 6px 0 4px 0;
    text-align: center;
}

.muted {
    color: #c9bfb6
}

.small {
    font-size: 13px
}

.input-label {
    display: block;
    margin-bottom: 4px;
    color: #c9bfb6;
    font-size: 13px;
}
```

```
.email,
.password {
    width: 100%;
    padding: 10px 12px;
    height: 44px;
    border-radius: 8px;
    border: 1px solid rgba(255, 255, 255, 0.04);
    background: rgba(255, 255, 255, 0.02);
    color: white;
    margin-top: 8px;
    box-sizing: border-box;
}

.password-wrap {
    position: relative;
    display: flex;
    align-items: center;
}

.password-wrap .password {
    padding-right: 72px;
}

.password.shown {
    color: white;
}

.show-toggle {
    position: absolute;
    right: 12px;
    top: 30px;
    transform: translateY(-50%);
    height: 28px;
    padding: 0 8px;
    display: inline-flex;
    align-items: center;
    justify-content: center;
    background: rgba(255, 179, 134, 0.06);
    border: 1px solid rgba(255, 179, 134, 0.12);
    color: #ffb386;
    border-radius: 6px;
    cursor: pointer;
    font-size: 13px;
    white-space: nowrap;
}

.show-toggle:focus {
    outline: 2px solid rgba(255, 179, 134, 0.25);
}

.actions {
    display: flex;
    align-items: center;
    justify-content: space-between;
    margin-top: 10px
}

.remember {
    color: #c9bfb6
}
```

```css
.forgot {
    color: #ffb386
}

.btn {
    padding: 10px 12px;
    border-radius: 8px;
    border: 1px solid rgba(255, 255, 255, 0.04);
    background: rgba(255, 255, 255, 0.02);
    color: inherit;
    cursor: pointer;
    text-align: center;
    display: flex;
    align-items: center;
    justify-content: center;
    gap: 10px
}

.btn.primary {
    background: #ff6b2b;
    color: white;
    border: none;
}

.full {
    width: 100%;
    margin: 12px auto;
}

.signup-link {
    margin-top: 14px;
    color: #c9bfb6;
    text-align: center;
    font-size: 14px;
}

.signup-link a {
    color: #ffb386
}

.username:focus,
.password:focus,
.password.shown:focus {
    border: 1px solid var(--accent) !important;
    outline: none !important;
    box-shadow: none !important;
}
@media(max-width:920px) {
    .admin-login-page {
        flex-direction: column
    }

    .admin-login-left {
        display: none
    }

    .admin-login-right {
        width: 100%;
        padding: 20px
    }
}
```

/frontend/src/AdminSignUp.js

```
<div className="admin-signup-page">
      <aside className="admin-signup-left">
        <div className="admin-signup-left-inner">
          <h1>Get started as Admin</h1>
          <p className="lead">Manage Plan Events with all admin tools in one place</p>

          <ul className="benefits">
            {perks.map((p, i) => (
              <li key={i}><span className="check">✓</span>{p}</li>
            ))}
          </ul>
        </div>
      </aside>

      <main className="admin-signup-right">
        <div className="admin-signup-card">
          <div className="brand">Plan Events</div>
          <h2>Admin Sign up</h2>
          <p className="muted small">Get started as admin</p>

          <form onSubmit={handleSubmit}>
            <div className="form-group">
              <label htmlFor="email" className='label-input'>Email</label>
              <input
                id="email"
                className="email"
                placeholder="Email"
                aria-label="Email"
                value={email}
                onChange={(e) => setEmail(e.target.value)}
              />
            </div>
            <div className="form-group">
              <label htmlFor="username" className='label-input'>Username</label>
              <input
                id="username"
                className="username"
                placeholder="Username"
                aria-label="Username"
                value={username}
                onChange={(e) => setUsername(e.target.value)}
              />
            </div>
            <div className="form-group">
              <label htmlFor="password" className='label-input'>Password</label>
              <div className="password-wrap">
                <input
                  id="password"
                  className={`password ${showPassword ? 'shown' : ''}`}
                  type={showPassword ? 'text' : 'password'}
                  placeholder="Password"
                  aria-label="Password"
                  value={password}
                  onChange={(e) => setPassword(e.target.value)}
                />
                <button type="button" className="show-toggle" onClick={toggleShow} aria-
pressed={showPassword} aria-label={showPassword ? 'Hide password' : 'Show
password'}>{showPassword ? 'Hide' : 'Show'}</button>
```

```
              </div>
            </div>

            <div className="actions">
              <label className="remember">
                <input type="checkbox" /> Remember me
              </label>
            </div>

            <button className="btn primary full" type="submit" disabled={loading}>
              {loading ? 'Signing up...' : 'Admin Sign up'}
            </button>
          </form>

          <div className="footer-link">Already an admin? <a href="/admin/login">Log
in</a></div>
        </div>
      </main>
    </div>
```

/frontend/src/AdminSignUp.css

```
.admin-signup-page {
     --left-bg-1: #2a2740;
     --left-bg-2: #213843;
     --left-highlight: rgba(255, 140, 60, 0.06);
     --card-bg-1: #0b0b0b;
     --card-bg-2: #081012;
     --accent: #ffb386;
     --cta: #ff6b2b;
     --check: #ff8a34;
     display: flex;
     min-height: 88vh;
     font-family: Inter, system-ui, -apple-system, 'Segoe UI', Roboto, 'Helvetica Neue', Arial;
}

.admin-signup-left {
     flex: 1;
     padding: 64px 72px;
     display: flex;
     text-align: center;
     align-items: center;
     background: radial-gradient(600px 300px at 25% 30%, rgba(255, 140, 60, 0.04), transparent
25%), linear-gradient(90deg, #2a2740 0%, #213843 100%);
     color: #f1eee9
}

.admin-signup-right {
     width: 420px;
     display: flex;
     align-items: center;
     justify-content: center;
     background: linear-gradient(180deg, rgba(0, 0, 0, 0.45), rgba(0, 0, 0, 0.55));
     padding: 48px
}

.admin-signup-left-inner {
     max-width: 560px
}
```

```
.admin-signup-left h1 {
    font-size: 36px;
    margin: 0 0 12px;
    color: #fff9f6;
}

.admin-signup-left .lead {
    color: #e6dbd3;
    margin-bottom: 22px
}

.benefits {
    list-style: none;
    padding: 0;
    margin: 0 0 30px 0;
}

.benefits li {
    display: flex;
    align-items: center;
    gap: 8px;
    margin: 12px 0;
    color: #f3e9e1;
}

.benefits .check {
    display: inline-flex;
    align-items: center;
    justify-content: center;
    width: 28px;
    height: 28px;
    background: var(--check);
    color: white;
    border-radius: 6px;
    font-weight: 700;
}

.trusted {
    margin-top: 36px;
    color: #ccbbb2
}

.trusted-logos {
    display: flex;
    gap: 14px;
    margin-top: 12px;
    align-items: center
}

.trusted-logos img {
    height: 28px;
    filter: brightness(0) invert(1);
    opacity: 0.85
}

.logo-placeholder {
    background: rgba(255, 255, 255, 0.03);
    padding: 6px 10px;
    border-radius: 6px;
    color: #ccbbb2;
    font-size: 12px;
```

```
}

.admin-signup-card {
    width: 100%;
    background: linear-gradient(180deg, var(--card-bg-1) 0%, var(--card-bg-2) 100%);
    border-radius: 12px;
    padding: 28px;
    box-shadow: 0 24px 80px rgba(3, 8, 12, 0.75);
    color: #f0eae3;
}

.admin-signup-card .brand {
    font-weight: 700;
    color: var(--accent);
    margin-bottom: 6px;
    text-transform: lowercase;
}

.admin-signup-card h2 {
    margin: 6px 0 4px 0;
    text-align: center;
}

.muted {
    color: #c9bfb6
}

.small {
    font-size: 13px;
}

.tiny {
    font-size: 12px;
}

.auth-buttons {
    display: flex;
    flex-direction: column;
    gap: 10px;
    margin: 14px 0
}

.btn.primary {
    background: var(--cta);
    color: white;
    border: none;
}

.full {
    width: 100%;
    margin-top: 12px
}

.divider {
    display: flex;
    align-items: center;
    margin: 14px 0
}

.divider:before,
.divider:after {
    content: '';
    flex: 1;
    height: 1px;
    background: rgba(255, 255, 255, 0.03)
}

.divider span {
    padding: 0 10px;
    color: #b59a8f
}

.username {
    width: 100%;
    padding: 10px;
    border-radius: 8px;
    border: none !important;
    outline: none !important;
    box-shadow: none !important;
    background: rgba(255, 255, 255, 0.02);
    color: inherit;
}

.email {
    width: 100%;
    padding: 10px;
    border-radius: 8px;
    border: none !important;
    outline: none !important;
    box-shadow: none !important;
    background: rgba(255, 255, 255, 0.02);
    color: inherit;
}

.password {
    width: 100%;
    padding: 10px;
    border-radius: 8px;
    border: none !important;
    outline: none !important;
    box-shadow: none !important;
    background: rgba(255, 255, 255, 0.02);
    color: inherit;
}

.username:focus,
.email:focus,
.password:focus,
.password.shown:focus {
    border: 1px solid var(--accent) !important;
    outline: none !important;
    box-shadow: none !important;
}

label {
    display: block;
    margin-bottom: 4px;
    font-size: 14px;
    color: #c9bfb6;
    font-weight: 500;
}
```

```css
.label-input {
    color: #c9bfb6;
}

.remember {
    color: #c9bfb6;
    font-size: 14px;
}

.footer-link {
    margin-top: 14px;
    color: #c9bfb6;
    text-align: center;
    font-size: 14px;
}

.footer-link a {
    color: #ffb386
}

.form-group {
    margin-bottom: 12px;
}

.password-wrap {
    display: flex;
    align-items: center;
    gap: 8px;
    margin: 0;
}

.show-toggle {
    background: none;
    border: none;
    color: var(--accent);
    cursor: pointer;
    font-size: 14px;
    padding: 0 4px;
    border-radius: 4px;
    transition: background-color 0.2s;
}

.show-toggle:hover {
    background: rgba(255, 179, 134, 0.1);
}

.forgot {
    color: #ffb386;
    text-decoration: none;
}

.forgot:hover {
    text-decoration: underline;
}

@media(max-width:920px) {
    .admin-signup-page {
        flex-direction: column
    }

    .admin-signup-left {
        display: none
    }

    .admin-signup-right {
        width: 100%;
        padding: 20px
    }
}
```

/frontend/src/SpeakerLogin.js

```
<div className="speaker-login-page">
            <aside className="speaker-login-left">
                <div className="speaker-login-left-inner">
                    <h1>Welcome back, Speaker</h1>
                    <p className="lead">Sign in to manage your presentations</p>

                    <ul className="benefits">
                        <li><span className="check">✓</span>Access speaker
dashboard</li>
                        <li><span className="check">✓</span>Manage your talks and
events</li>
                        <li><span className="check">✓</span>Secure speaker tools</li>
                    </ul>
                </div>
            </aside>

            <main className="speaker-login-right">
                <div className="speaker-login-card">
                    <div className="brand">Plan Events</div>
                    <h2>Speaker Log in</h2>
                    <p className="muted small">Sign in to your speaker account</p>

                    <form onSubmit={handleSubmit}>
                        <label htmlFor="email" className="input-label">Email or
Username</label>
                        <input
                            id="email"
                            className="email"
                            placeholder="Email or Username"
                            aria-label="Email or Username"
                            value={email}
                            onChange={(e) => setEmail(e.target.value)}
                        />
                        <label htmlFor="password" className="input-
label">Password</label>
                        <div className="password-wrap">
                            <input
                                id="password"
                                className={`password ${showPassword ? 'shown' : ''}`}
                                type={showPassword ? 'text' : 'password'}
                                placeholder="Password"
                                aria-label="Password"
                                value={password}
                                onChange={(e) => setPassword(e.target.value)}
                            />
                            <button type="button" className="show-toggle"
onClick={toggleShow} aria-pressed={showPassword} aria-label={showPassword ? 'Hide password' :
'Show password'}>{showPassword ? 'Hide' : 'Show'}</button>
```

```
                              </div>

                              <div className="actions">
                                   <label className="remember"><input type="checkbox" />
Remember me</label>

                                   <a className="forgot" href="/forgot">Forgot?</a>
                              </div>

                              <button className="btn primary full" type="submit"
disabled={loading}>
                                   {loading ? 'Logging in...' : 'Speaker Login'}
                              </button>
                         </form>

                         <div className="signup-link">New speaker? <a
href="/speaker/signup">Create a speaker account</a></div>
                    </div>
               </main>
          </div>
```

/frontend/src/SpeakerLogin.css

```
.speaker-login-page {
     display: flex;
     min-height: 88vh;
     font-family: Inter, system-ui, -apple-system, 'Segoe UI', Roboto, Arial
}

.speaker-login-left {
     flex: 1;
     padding: 64px 72px;
     display: flex;
     text-align: center;
     align-items: center;
     background: radial-gradient(600px 300px at 25% 30%, rgba(255, 140, 60, 0.04), transparent
25%), linear-gradient(90deg, #2a2740 0%, #213843 100%);
     color: #f1eee9
}

.speaker-login-left-inner {
     text-align: left;
     width: 560px;
     padding: 36px 40px;
     border-radius: 8px;
     display: flex;
     flex-direction: column;
     align-items: left;
     justify-content: left;
     gap: 18px;
}

.speaker-login-left h1 {
     font-size: 36px;
     margin: 0 0 12px;
     color: #fff9f6
}

.speaker-login-left .lead {
     color: #e6dbd3;
     margin-bottom: 22px
```

```css
}

.benefits {
    list-style: none;
    padding: 0;
    margin: 0;
    display: flex;
    flex-direction: column;
    gap: 12px;
    align-items: flex-start;
}

.benefits li {
    display: flex;
    align-items: center;
    justify-content: center;
    gap: 12px;
    margin: 0;
}

.benefits .check {
    display: inline-flex;
    align-items: center;
    justify-content: center;
    width: 28px;
    height: 28px;
    background: #ff8a34;
    color: white;
    border-radius: 6px;
    font-weight: 700
}

.speaker-login-right {
    width: 420px;
    display: flex;
    align-items: center;
    justify-content: center;
    background: linear-gradient(180deg, rgba(0, 0, 0, 0.45), rgba(0, 0, 0, 0.55));
    padding: 48px
}

.speaker-login-card {
    width: 100%;
    background: linear-gradient(180deg, #0b0b0b 0%, #081012 100%);
    border-radius: 12px;
    padding: 28px;
    box-shadow: 0 24px 80px rgba(3, 8, 12, 0.75);
    color: #f0eae3
}

.speaker-login-card .brand {
    font-weight: 700;
    color: #ffb386;
    margin-bottom: 6px;
    text-transform: lowercase
}

.speaker-login-card h2 {
    margin: 6px 0 4px 0;
    text-align: center;
}
```

```css
.muted {
    color: #c9bfb6
}

.small {
    font-size: 13px
}

.input-label {
    display: block;
    margin-bottom: 4px;
    color: #c9bfb6;
    font-size: 13px;
}

.email,
.password {
    width: 100%;
    padding: 10px 12px;
    height: 44px;
    border-radius: 8px;
    border: 1px solid rgba(255, 255, 255, 0.04);
    background: rgba(255, 255, 255, 0.02);
    color: white;
    margin-top: 8px;
    box-sizing: border-box;
}

.password-wrap {
    position: relative;
    display: flex;
    align-items: center;
}

.password-wrap .password {
    padding-right: 72px;
}

.password.shown {
    color: white;
}

.show-toggle {
    position: absolute;
    right: 12px;
    top: 30px;
    transform: translateY(-50%);
    height: 28px;
    padding: 0 8px;
    display: inline-flex;
    align-items: center;
    justify-content: center;
    background: rgba(255, 179, 134, 0.06);
    border: 1px solid rgba(255, 179, 134, 0.12);
    color: #ffb386;
    border-radius: 6px;
    cursor: pointer;
    font-size: 13px;
    white-space: nowrap;
}
```

```
.show-toggle:focus {
    outline: 2px solid rgba(255, 179, 134, 0.25);
}

.actions {
    display: flex;
    align-items: center;
    justify-content: space-between;
    margin-top: 10px
}

.remember {
    color: #c9bfb6
}

.forgot {
    color: #ffb386
}

.btn {
    padding: 10px 12px;
    border-radius: 8px;
    border: 1px solid rgba(255, 255, 255, 0.04);
    background: rgba(255, 255, 255, 0.02);
    color: inherit;
    cursor: pointer;
    text-align: center;
    display: flex;
    align-items: center;
    justify-content: center;
    gap: 10px
}

.btn.primary {
    background: #ff6b2b;
    color: white;
    border: none;
}

.full {
    width: 100%;
    margin: 12px auto;
}

.signup-link {
    margin-top: 14px;
    color: #c9bfb6;
    text-align: center;
    font-size: 14px;
}

.signup-link a {
    color: #ffb386
}

.username:focus,
.password:focus,
.password.shown:focus {
    border: 1px solid var(--accent) !important;
    outline: none !important;
```

```
     box-shadow: none !important;
}

@media(max-width:920px) {
     .speaker-login-page {
          flex-direction: column
     }

     .speaker-login-left {
          display: none
     }

     .speaker-login-right {
          width: 100%;
          padding: 20px
     }
}
```

/frontend/src/SpeakerSignUp.js

```
<div className="speaker-signup-page">
      <aside className="speaker-signup-left">
        <div className="speaker-signup-left-inner">
          <h1>Get started as Speaker</h1>
          <p className="lead">Manage your talks with all speaker tools in one place</p>

          <ul className="benefits">
            {perks.map((p, i) => (
              <li key={i}><span className="check">✓</span>{p}</li>
            ))}
          </ul>
        </div>
      </aside>

      <main className="speaker-signup-right">
        <div className="speaker-signup-card">
          <div className="brand">Plan Events</div>
          <h2>Speaker Sign up</h2>
          <p className="muted small">Get started as speaker</p>

          <form onSubmit={handleSubmit}>
            <div className="form-group">
              <label htmlFor="email" className='label-input'>Email</label>
              <input
                id="email"
                className="email"
                placeholder="Email"
                aria-label="Email"
                value={email}
                onChange={(e) => setEmail(e.target.value)}
              />
            </div>
            <div className="form-group">
              <label htmlFor="username" className='label-input'>Username</label>
              <input
                id="username"
                className="username"
                placeholder="Username"
                aria-label="Username"
                value={username}
                onChange={(e) => setUsername(e.target.value)}
              />
            </div>
            <div className="form-group">
              <label htmlFor="password" className='label-input'>Password</label>
              <div className="password-wrap">
                <input
                  id="password"
                  className={`password ${showPassword ? 'shown' : ''}`}
                  type={showPassword ? 'text' : 'password'}
                  placeholder="Password"
                  aria-label="Password"
                  value={password}
                  onChange={(e) => setPassword(e.target.value)}
                />
                <button type="button" className="show-toggle" onClick={toggleShow} aria-
pressed={showPassword} aria-label={showPassword ? 'Hide password' : 'Show
password'}>{showPassword ? 'Hide' : 'Show'}</button>
              </div>
            </div>

            <div className="actions">
              <label className="remember">
                <input type="checkbox" /> Remember me
              </label>
              <a className="forgot" href="/forgot-password">Forgot?</a>
            </div>

            <button className="btn primary full" type="submit" disabled={loading}>
              {loading ? 'Signing up...' : 'Speaker Sign up'}
            </button>
          </form>

          <div className="footer-link">Already a speaker? <a href="/speaker/login">Log
in</a></div>
        </div>
      </main>
    </div>
```

/frontend/src/SpeakerSignUp.css

```
.speaker-signup-page {
     --left-bg-1: #2a2740;
     --left-bg-2: #213843;
     --left-highlight: rgba(255, 140, 60, 0.06);
     --card-bg-1: #0b0b0b;
     --card-bg-2: #081012;
     --accent: #ffb386;
     --cta: #ff6b2b;
     --check: #ff8a34;
     display: flex;
     min-height: 88vh;
     font-family: Inter, system-ui, -apple-system, 'Segoe UI', Roboto, 'Helvetica Neue', Arial;
}

.speaker-signup-left {
     flex: 1;
     padding: 64px 72px;
     display: flex;
```

```
    text-align: center;
    align-items: center;
    background: radial-gradient(600px 300px at 25% 30%, rgba(255, 140, 60, 0.04), transparent
25%), linear-gradient(90deg, #2a2740 0%, #213843 100%);
    color: #f1eee9
}

.speaker-signup-right {
    width: 420px;
    display: flex;
    align-items: center;
    justify-content: center;
    background: linear-gradient(180deg, rgba(0, 0, 0, 0.45), rgba(0, 0, 0, 0.55));
    padding: 48px
}

.speaker-signup-left-inner {
    max-width: 560px
}

.speaker-signup-left h1 {
    font-size: 36px;
    margin: 0 0 12px;
    color: #fff9f6;
}

.speaker-signup-left .lead {
    color: #e6dbd3;
    margin-bottom: 22px
}

.benefits {
    list-style: none;
    padding: 0;
    margin: 0 0 30px 0;
}

.benefits li {
    display: flex;
    align-items: center;
    gap: 8px;
    margin: 12px 0;
    color: #f3e9e1;
}

.benefits .check {
    display: inline-flex;
    align-items: center;
    justify-content: center;
    width: 28px;
    height: 28px;
    background: var(--check);
    color: white;
    border-radius: 6px;
    font-weight: 700;
}

.trusted {
    margin-top: 36px;
    color: #ccbbb2
```

```
}
.trusted-logos {
    display: flex;
    gap: 14px;
    margin-top: 12px;
    align-items: center
}

.trusted-logos img {
    height: 28px;
    filter: brightness(0) invert(1);
    opacity: 0.85
}

.logo-placeholder {
    background: rgba(255, 255, 255, 0.03);
    padding: 6px 10px;
    border-radius: 6px;
    color: #ccbbb2;
    font-size: 12px;
}

.speaker-signup-card {
    width: 100%;
    background: linear-gradient(180deg, var(--card-bg-1) 0%, var(--card-bg-2) 100%);
    border-radius: 12px;
    padding: 28px;
    box-shadow: 0 24px 80px rgba(3, 8, 12, 0.75);
    color: #f0eae3;
}

.speaker-signup-card .brand {
    font-weight: 700;
    color: var(--accent);
    margin-bottom: 6px;
    text-transform: lowercase;
}

.speaker-signup-card h2 {
    margin: 6px 0 4px 0;
    text-align: center;
}

.muted {
    color: #c9bfb6
}

.small {
    font-size: 13px;
}

.tiny {
    font-size: 12px;
}

.auth-buttons {
    display: flex;
    flex-direction: column;
    gap: 10px;
    margin: 14px 0
}
```

```css
.btn.primary {
     background: var(--cta);
     color: white;
     border: none;
}

.full {
     width: 100%;
     margin-top: 12px
}

.divider {
     display: flex;
     align-items: center;
     margin: 14px 0
}

.divider:before,
.divider:after {
     content: '';
     flex: 1;
     height: 1px;
     background: rgba(255, 255, 255, 0.03)
}

.divider span {
     padding: 0 10px;
     color: #b59a8f
}

.username {
     width: 100%;
     padding: 10px;
     border-radius: 8px;
     border: none !important;
     outline: none !important;
     box-shadow: none !important;
     background: rgba(255, 255, 255, 0.02);
     color: inherit;
}

.email {
     width: 100%;
     padding: 10px;
     border-radius: 8px;
     border: none !important;
     outline: none !important;
     box-shadow: none !important;
     background: rgba(255, 255, 255, 0.02);
     color: inherit;
}

.password {
     width: 100%;
     padding: 10px;
     border-radius: 8px;
     border: none !important;
     outline: none !important;
     box-shadow: none !important;
     background: rgba(255, 255, 255, 0.02);
     color: inherit;
}

.username:focus,
.email:focus,
.password:focus,
.password.shown:focus {
     border: 1px solid var(--accent) !important;
     outline: none !important;
     box-shadow: none !important;
}

label {
     display: block;
     margin-bottom: 4px;
     font-size: 14px;
     color: #c9bfb6;
     font-weight: 500;
}

.label-input {
     color: #c9bfb6;
}

.remember {
     color: #c9bfb6;
     font-size: 14px;
}

.footer-link {
     margin-top: 14px;
     color: #c9bfb6;
     text-align: center;
     font-size: 14px;
}

.footer-link a {
     color: #ffb386
}

.form-group {
     margin-bottom: 12px;
}

.password-wrap {
     display: flex;
     align-items: center;
     gap: 8px;
     margin: 0;
}

.show-toggle {
     background: none;
     border: none;
     color: var(--accent);
     cursor: pointer;
     font-size: 14px;
     padding: 0 4px;
     border-radius: 4px;
     transition: background-color 0.2s;
}
```

```
.show-toggle:hover {
     background: rgba(255, 179, 134, 0.1);
}

.forgot {
     color: #ffb386;
     text-decoration: none;
}

.forgot:hover {
     text-decoration: underline;
}

@media(max-width:920px) {
     .speaker-signup-page {
          flex-direction: column
     }

     .speaker-signup-left {
          display: none
     }

     .speaker-signup-right {
          width: 100%;
          padding: 20px
     }
}
```

/frontend/src/ForgotPassword.js

```
<div className="forgot-password-page">
               <aside className="forgot-left">
                    <div className="forgot-left-inner">
                         <h1>Welcome back</h1>
                         <ul className="welcome-bullets">
                              <li><span className="check">✓</span>Quick access to
dashboard</li>
                              <li><span className="check">✓</span>Manage attendees and
tickets</li>
                              <li><span className="check">✓</span>Secure event tools</li>
                         </ul>
                    </div>
               </aside>

               <main className="forgot-right">
                    <div className="forgot-card">
                         <div className="brand">Plan Events</div>
                         <h2>Forgot Password</h2>
                         <p className="muted small">Enter your email to receive a reset
link</p>

                         <form onSubmit={handleSubmit}>
                              <label className="input-label">Email</label>
                              <input
                                   type="email"
                                   className="email"
                                   placeholder="Email"
                                   aria-label="Email"
```

```
                                  value={email}
                                  onChange={(e) => setEmail(e.target.value)}
                                  required
                             />

                             <button type="submit" className="btn primary full">Send Reset
Link</button>
                        </form>

                        {message && <p className="success">{message}</p>}
                        {error && <p className="error">{error}</p>}

                        <div className="back-link">
                             <a href="/login">Back to Login</a>
                        </div>
                   </div>
              </main>
         </div>
```

/frontend/src/ForgotPassword.css

```
.forgot-password-page {
    display: flex;
    min-height: 88vh;
    font-family: Inter, system-ui, -apple-system, 'Segoe UI', Roboto, Arial
}

.forgot-left {
    flex: 1;
    padding: 64px 72px;
    display: flex;
    text-align: center;
    align-items: center;
    background: radial-gradient(600px 300px at 25% 30%, rgba(255, 140, 60, 0.04), transparent
25%), linear-gradient(90deg, #2a2740 0%, #213843 100%);
    color: #f1eee9;
}

.forgot-left-inner {
    text-align: left;
    width: 560px;
    padding: 36px 40px;
    border-radius: 8px;
    display: flex;
    flex-direction: column;
    align-items: flex-start;
    justify-content: flex-start;
    gap: 18px;
}

.forgot-left h1 {
    font-size: 36px;
    margin: 0 0 12px;
    color: #fff9f6
}

.welcome-bullets {
    list-style: none;
    padding: 0;
    margin: 0;
    display: flex;
    flex-direction: column;
    gap: 12px;
    align-items: flex-start;
}

.welcome-bullets li {
    display: flex;
    align-items: center;
    justify-content: flex-start;
    gap: 12px;
    margin: 0;
}

.welcome-bullets .check {
    display: inline-flex;
    align-items: center;
    justify-content: center;
    width: 28px;
    height: 28px;
    background: #ff8a34;
    color: white;
    border-radius: 6px;
    font-weight: 700
}

.forgot-right {
    width: 420px;
    display: flex;
    align-items: center;
    justify-content: center;
    background: linear-gradient(180deg, rgba(0, 0, 0, 0.45), rgba(0, 0, 0, 0.55));
    padding: 48px
}

.forgot-card {
    width: 100%;
    background: linear-gradient(180deg, #0b0b0b 0%, #081012 100%);
    border-radius: 12px;
    padding: 28px;
    box-shadow: 0 24px 80px rgba(3, 8, 12, 0.75);
    color: #f0eae3
}

.forgot-card .brand {
    font-weight: 700;
    color: #ffb386;
    margin-bottom: 6px;
}

.forgot-card h2 {
    margin: 6px 0 12px 0;
    text-align: center;
}

.muted {
    color: #c9bfb6;
    margin-bottom: 16px;
}

.small {
    font-size: 13px
}

.input-label {
    display: block;
    margin-bottom: 4px;
    color: #c9bfb6;
    font-size: 13px;
}

.email,
.password {
    width: 100%;
    padding: 10px 12px;
    height: 44px;
    border-radius: 8px;
    border: 1px solid rgba(255, 255, 255, 0.04);
    background: rgba(255, 255, 255, 0.02);
    color: white;
    margin-top: 8px;
    box-sizing: border-box;
}

.password-wrap {
    position: relative;
    display: flex;
    align-items: center;
}

.password-wrap .password {
    padding-right: 72px;
}

.password.shown {
    color: white;
}

.show-toggle {
    position: absolute;
    right: 12px;
    top: 30px;
    transform: translateY(-50%);
    height: 28px;
    padding: 0 8px;
    display: inline-flex;
    align-items: center;
    justify-content: center;
    background: rgba(255, 179, 134, 0.06);
    border: 1px solid rgba(255, 179, 134, 0.12);
    color: #ffb386;
    border-radius: 6px;
    cursor: pointer;
    font-size: 13px;
    white-space: nowrap;
}

.show-toggle:focus {
    outline: 2px solid rgba(255, 179, 134, 0.25);
}

.actions {
    display: flex;
    align-items: center;
    justify-content: space-between;
    margin-top: 10px
}

.remember {
    color: #c9bfb6
}

.forgot {
    color: #ffb386
}

.btn {
    padding: 10px 12px;
    border-radius: 8px;
    border: 1px solid rgba(255, 255, 255, 0.04);
    background: rgba(255, 255, 255, 0.02);
    color: inherit;
    cursor: pointer;
    text-align: center;
    display: flex;
    align-items: center;
    justify-content: center;
    gap: 10px
}

.btn.primary {
    background: #ff6b2b;
    color: white;
    border: none;
}

.full {
    width: 100%;
    margin: 12px auto;
}

.signup-link {
    margin-top: 14px;
    color: #c9bfb6;
    text-align: center;
    font-size: 14px;
}

.signup-link a {
    color: #ffb386
}

.username:focus,
.password:focus,
.password.shown:focus {
    border: 1px solid var(--accent) !important;
    outline: none !important;
    box-shadow: none !important;
}

.success {
    color: #28a745;
    text-align: center;
    margin-top: 1rem;
}

.error {
    color: #dc3545;
    text-align: center;
    margin-top: 1rem;
}

.back-link {
    margin-top: 20px;
    color: #c9bfb6;
    text-align: center;
    font-size: 14px;
}

.back-link a {
    color: #ffb386
}

@media(max-width:920px) {
    .forgot-password-page {
        flex-direction: column
    }

    .forgot-left {
        display: none
    }

    .forgot-right {
        width: 100%;
        padding: 20px
    }
}
```

/backend/app.py

```
from flask import Flask, jsonify, request, send_from_directory
from flask_cors import CORS
import sqlite3
import os
from dotenv import load_dotenv
from werkzeug.utils import secure_filename

load_dotenv()

app = Flask(__name__)
CORS(app)

DATABASE = os.getenv('DATABASE_URL') or 'events.db'

def get_db_connection():
    conn = sqlite3.connect(DATABASE)
    conn.row_factory = sqlite3.Row
    return conn

def log_action(action, details, admin_user=None):
    conn = get_db_connection()
    cursor = conn.cursor()
    cursor.execute("""
        INSERT INTO audit_logs (action, details, admin_user)
        VALUES (?, ?, ?)
    """, (action, details, admin_user))
    conn.commit()
    conn.close()

def create_table():
    conn = get_db_connection()
    cursor = conn.cursor()
    cursor.execute("""
        CREATE TABLE IF NOT EXISTS users (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            username TEXT NOT NULL UNIQUE,
            email TEXT NOT NULL UNIQUE,
            password TEXT NOT NULL,
            phone TEXT,
            created_at DATETIME DEFAULT CURRENT_TIMESTAMP,
            role TEXT DEFAULT 'user'
        )
    """)
    cursor.execute("""
        CREATE TABLE IF NOT EXISTS events (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            banner TEXT,
            title TEXT NOT NULL,
            description TEXT,
            date TEXT,
            start_time TEXT,
            end_time TEXT,
            location TEXT,
            category TEXT,
            organizer TEXT,
            event_type TEXT,
            requirements TEXT,
            speakers TEXT,
            agenda TEXT,
            registration_link TEXT,
            ticket_info TEXT
        )
    """)
    cursor.execute("""
        CREATE TABLE IF NOT EXISTS registrations (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            event_id INTEGER NOT NULL,
            name TEXT NOT NULL,
            email TEXT NOT NULL,
            phone TEXT,
            ticket_type TEXT NOT NULL,
            quantity INTEGER NOT NULL,
            payment_status TEXT DEFAULT 'pending',
            checked_in BOOLEAN DEFAULT FALSE,
            created_at DATETIME DEFAULT CURRENT_TIMESTAMP,
            FOREIGN KEY (event_id) REFERENCES events (id)
        )
    """)
    cursor.execute("""
        CREATE TABLE IF NOT EXISTS audit_logs (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            action TEXT NOT NULL,
            details TEXT,
            admin_user TEXT,
            timestamp DATETIME DEFAULT CURRENT_TIMESTAMP
        )
    """)
    cursor.execute("""
        CREATE TABLE IF NOT EXISTS admin_settings (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            key TEXT UNIQUE NOT NULL,
            value TEXT NOT NULL,
            updated_at DATETIME DEFAULT CURRENT_TIMESTAMP
        )
    """)
    conn.commit()
    print("Tables ensured (created if missing)")

    # Ensure missing columns (safe migration for older DBs)
    cursor.execute("PRAGMA table_info(events)")
    existing_cols = [row['name'] for row in cursor.fetchall()]

    # Check for duplicate columns and migrate if necessary
    has_duplicates = 'eventType' in existing_cols or 'registrationLink' in existing_cols or
'additionalDetails' in existing_cols
    if has_duplicates:
        print("Duplicate columns detected. Migrating table...")
        # Create new table with correct schema
        cursor.execute("""
        CREATE TABLE events_new (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            banner TEXT,
            title TEXT NOT NULL,
            description TEXT,
            date TEXT,
            start_time TEXT,
            end_time TEXT,
            location TEXT,
            category TEXT,
            organizer TEXT,
            event_type TEXT,
            speakers TEXT,
            agenda TEXT,
            registration_link TEXT,
            ticket_info TEXT,
            requirements TEXT
        )
        """)
        # Copy data, preferring underscored columns
        cursor.execute("""
        INSERT INTO events_new (id, banner, title, description, date, start_time, end_time,
location, category, organizer, event_type, speakers, agenda, registration_link, ticket_info,
requirements)
        SELECT id, banner, title, description, date, start_time, end_time, location, category,
organizer,
               COALESCE(event_type, eventType, 'Event') as event_type,
               speakers,
               agenda,
               COALESCE(registration_link, registrationLink, 'https://plan-events.com/register')
as registration_link,
               ticket_info,
               '' as requirements
        FROM events
        """)
        # Drop old table
        cursor.execute("DROP TABLE events")
        # Rename new table
        cursor.execute("ALTER TABLE events_new RENAME TO events")
        conn.commit()
        print("Table migrated. Duplicates removed.")
        # Re-fetch existing_cols after migration
        cursor.execute("PRAGMA table_info(events)")
        existing_cols = [row['name'] for row in cursor.fetchall()]

    # Add any missing columns that the code expects
    if 'category' not in existing_cols:
        cursor.execute("ALTER TABLE events ADD COLUMN category TEXT")
        conn.commit()
        print("Added missing column: category")
    if 'organizer' not in existing_cols:
        cursor.execute("ALTER TABLE events ADD COLUMN organizer TEXT")
        conn.commit()
        print("Added missing column: organizer")
    if 'venue' in existing_cols and 'location' not in existing_cols:
        cursor.execute("ALTER TABLE events RENAME COLUMN venue TO location")
        conn.commit()
        print("Renamed venue to location")
    if 'banner' not in existing_cols:
        cursor.execute("ALTER TABLE events ADD COLUMN banner TEXT")
        conn.commit()
        print("Added missing column: banner")

    if 'event_type' not in existing_cols:
        cursor.execute("ALTER TABLE events ADD COLUMN event_type TEXT")
        conn.commit()
        print("Added missing column: event_type")
    if 'requirements' not in existing_cols:
        cursor.execute("ALTER TABLE events ADD COLUMN requirements TEXT")
        conn.commit()
        print("Added missing column: requirements")
    if 'speakers' not in existing_cols:
        cursor.execute("ALTER TABLE events ADD COLUMN speakers TEXT")
        conn.commit()
        print("Added missing column: speakers")
    if 'agenda' not in existing_cols:
        cursor.execute("ALTER TABLE events ADD COLUMN agenda TEXT")
        conn.commit()
        print("Added missing column: agenda")
    if 'registration_link' not in existing_cols:
        cursor.execute("ALTER TABLE events ADD COLUMN registration_link TEXT")
        conn.commit()
        print("Added missing column: registration_link")
    if 'ticket_info' not in existing_cols:
        cursor.execute("ALTER TABLE events ADD COLUMN ticket_info TEXT")
        conn.commit()
        print("Added missing column: ticket_info")

    # Check users table for phone column
    cursor.execute("PRAGMA table_info(users)")
    user_cols = [row['name'] for row in cursor.fetchall()]
    if 'phone' not in user_cols:
        cursor.execute("ALTER TABLE users ADD COLUMN phone TEXT")
        conn.commit()
        print("Added missing column: phone to users")
    # Check registrations table for created_at column and migrate if necessary
    cursor.execute("PRAGMA table_info(registrations)")
    reg_cols = [row['name'] for row in cursor.fetchall()]
    if 'created_at' not in reg_cols:
        print("Missing created_at column in registrations. Migrating table...")
        # Create new table with correct schema including created_at
        cursor.execute("""
        CREATE TABLE registrations_new (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            event_id INTEGER NOT NULL,
            name TEXT NOT NULL,
            email TEXT NOT NULL,
            phone TEXT,
            ticket_type TEXT NOT NULL,
            quantity INTEGER NOT NULL,
            payment_status TEXT DEFAULT 'pending',
            checked_in BOOLEAN DEFAULT FALSE,
            created_at DATETIME DEFAULT CURRENT_TIMESTAMP,
            FOREIGN KEY (event_id) REFERENCES events (id)
        )
        """)
        # Copy data from old table, setting created_at to NULL for existing rows
        cursor.execute("""
        INSERT INTO registrations_new (id, event_id, name, email, phone, ticket_type, quantity,
payment_status, checked_in)
        SELECT id, event_id, name, email, phone, ticket_type, quantity, payment_status,
checked_in
        FROM registrations
        """)
        # Drop old table
        cursor.execute("DROP TABLE registrations")
        # Rename new table
        cursor.execute("ALTER TABLE registrations_new RENAME TO registrations")
        conn.commit()
        print("Registrations table migrated with created_at column.")

    # Check audit_logs table for column name migration
    cursor.execute("PRAGMA table_info(audit_logs)")
    audit_cols = [row['name'] for row in cursor.fetchall()]
    if 'user' in audit_cols and 'admin_user' not in audit_cols:
        print("Renaming column 'user' to 'admin_user' in audit_logs table...")
        cursor.execute("ALTER TABLE audit_logs RENAME COLUMN user TO admin_user")
        conn.commit()
        print("Column renamed successfully.")

    # Insert initial data only if table is empty
    cursor.execute("SELECT COUNT(*) as cnt FROM events")
    count = cursor.fetchone()[0]
    if count == 0:
        cursor.execute("""
            INSERT INTO events (title, description, date, start_time, end_time, location,
category, organizer, banner, event_type, speakers, agenda, registration_link, ticket_info,
requirements)
            VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)
        """, (
            'Tech Conference 2024',
            'A premier conference showcasing the latest advancements in technology.',
            '2024-09-15',
            '09:00',
            '17:00',
            'San Francisco, CA',
            'Technology',
            'Tech Innovations Inc.',
            'https://via.placeholder.com/600x200?text=Tech+Conference',
            'Conference',
            '[{"name": "John Doe", "bio": "Software Engineer"}]',
            '[{"start_time": "09:00", "end_time": "10:00", "description": "Keynote"}]',
            'https://example.com/register',
            'Early bird tickets available',
            'No specific requirements'
        ))
        cursor.execute("""
            INSERT INTO events (title, description, date, start_time, end_time, location,
category, organizer, banner, event_type, speakers, agenda, registration_link, ticket_info,
requirements)
            VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)
        """, (
            'Music Festival',
            'An electrifying music festival with performances from top artists.',
            '2024-10-05',
            '18:00',
            '22:00',
            'Austin, TX',
            'Entertainment',
            'Austin Music Group',
            'https://via.placeholder.com/600x200?text=Music+Festival',
            'Festival',
            '[{"name": "Jane Smith", "bio": "Singer"}]',
            '[{"start_time": "18:00", "end_time": "20:00", "description": "Main Stage"}]',
            'https://example.com/tickets',
            'VIP tickets available',
            'Bring your dancing shoes'
        ))

        # Insert sample registrations only if table is empty
        cursor.execute("SELECT COUNT(*) as cnt FROM registrations")
        reg_count = cursor.fetchone()[0]
        if reg_count == 0:
            cursor.execute("""
                INSERT INTO registrations (event_id, name, email, phone, ticket_type, quantity,
payment_status)
                VALUES (?, ?, ?, ?, ?, ?, ?)
            """, (1, 'Alice Johnson', 'alice@example.com', '123-456-7890', 'General', 2,
'paid'))
            cursor.execute("""
                INSERT INTO registrations (event_id, name, email, phone, ticket_type, quantity,
payment_status)
                VALUES (?, ?, ?, ?, ?, ?, ?)
            """, (1, 'Bob Smith', 'bob@example.com', None, 'VIP', 1, 'pending'))
            cursor.execute("""
                INSERT INTO registrations (event_id, name, email, phone, ticket_type, quantity,
payment_status)
                VALUES (?, ?, ?, ?, ?, ?, ?)
            """, (2, 'Carol Davis', 'carol@example.com', '098-765-4321', 'General', 1, 'paid'))
            cursor.execute("""
                INSERT INTO registrations (event_id, name, email, phone, ticket_type, quantity,
payment_status)
                VALUES (?, ?, ?, ?, ?, ?, ?)
            """, (2, 'David Wilson', 'david@example.com', None, 'VIP', 3, 'paid'))
            print("Sample registrations inserted")
        # Insert sample users only if table is empty
        cursor.execute("SELECT COUNT(*) as cnt FROM users")
        user_count = cursor.fetchone()[0]
        if user_count == 0:
            cursor.execute("""
                INSERT INTO users (username, email, password, phone, role)
                VALUES (?, ?, ?, ?, ?)
            """, ('admin', 'admin@example.com', 'adminpass', '000-000-0000', 'admin'))
            cursor.execute("""
                INSERT INTO users (username, email, password, phone, role)
                VALUES (?, ?, ?, ?, ?)
            """, ('user1', 'user1@example.com', 'pass123', '111-222-3333', 'user'))
            cursor.execute("""
                INSERT INTO users (username, email, password, phone, role)
                VALUES (?, ?, ?, ?, ?)
            """, ('user2', 'user2@example.com', 'pass456', None, 'user'))
            print("Sample users inserted")

        conn.commit()
        print("Initial data inserted")

    conn.close()

create_table()

@app.route('/', methods=['GET'])
def index():
    return "Server is running"

@app.route('/events', methods=['GET', 'POST'])
def events():
    print("Events endpoint hit")
    conn = get_db_connection()
    cursor = conn.cursor()

    if request.method == 'GET':
        cursor.execute("SELECT * FROM events")
        rows = cursor.fetchall()
        events = [dict(row) for row in rows]
        conn.close()
        return jsonify(events)

    if request.method == 'POST':
        if request.content_type.startswith('multipart/form-data'):
            # Handle file upload
            title = request.form['title']
            description = request.form.get('description')
            date = request.form.get('date')
            start_time = request.form.get('start_time')
            end_time = request.form.get('end_time')
            location = request.form.get('location')
            category = request.form.get('category')
            organizer = request.form.get('organizer')
            event_type = request.form.get('event_type')
            speakers = request.form.get('speakers')
            agenda = request.form.get('agenda')
            registration_link = request.form.get('registration_link')
            ticket_info = request.form.get('ticket_info')
            requirements = request.form.get('requirements')
            banner = None
            if 'banner' in request.files:
                file = request.files['banner']
                if file.filename != '':
                    filename = secure_filename(file.filename)
                    file.save(os.path.join('uploads', filename))
                    banner = filename
        else:
            # Handle JSON
            data = request.get_json()
            title = data['title']
            description = data.get('description')
            date = data.get('date')
            start_time = data.get('start_time')
            end_time = data.get('end_time')
            location = data.get('location')
            category = data.get('category')
            organizer = data.get('organizer')
            event_type = data.get('event_type')
            speakers = data.get('speakers')
            agenda = data.get('agenda')
            registration_link = data.get('registration_link')
            ticket_info = data.get('ticket_info')
            banner = data.get('banner')
            requirements = data.get('requirements')

        cursor.execute("""
            INSERT INTO events (title, description, date, start_time, end_time, location,
category, organizer, banner, event_type, speakers, agenda, registration_link, ticket_info,
requirements)
            VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)
        """, (title, description, date, start_time, end_time, location, category, organizer,
banner, event_type, speakers, agenda, registration_link, ticket_info, requirements))
        conn.commit()
        log_action('Event Created', f"Event '{title}' created", 'admin@example.com')  #
Assuming admin email
        conn.close()
        return jsonify({'message': 'Event created successfully'}), 201

@app.route('/events/<int:id>', methods=['GET', 'PUT', 'DELETE'])
def event(id):
    print("Event by ID endpoint hit")
    conn = get_db_connection()
    cursor = conn.cursor()

    if request.method == 'GET':
        cursor.execute("SELECT * FROM events WHERE id = ?", (id,))
        row = cursor.fetchone()
        if row:
            event = dict(row)
            conn.close()
            return jsonify(event)
        else:
            conn.close()
            return jsonify({'message': 'Event not found'}), 404

    if request.method == 'PUT':
        if request.content_type.startswith('multipart/form-data'):
            # Handle file upload
            title = request.form['title']
            description = request.form.get('description')
            date = request.form.get('date')
            start_time = request.form.get('start_time')
            end_time = request.form.get('end_time')
            location = request.form.get('location')
            category = request.form.get('category')
            organizer = request.form.get('organizer')
            event_type = request.form.get('event_type')
            speakers = request.form.get('speakers')
            agenda = request.form.get('agenda')
            registration_link = request.form.get('registration_link')
            ticket_info = request.form.get('ticket_info')
            requirements = request.form.get('requirements')
            banner = None
            if 'banner' in request.files:
                file = request.files['banner']
                if file.filename != '':
                    filename = secure_filename(file.filename)
                    file.save(os.path.join('uploads', filename))
                    banner = filename
        else:
            # Handle JSON
            data = request.get_json()
            title = data['title']
            description = data.get('description')
            date = data.get('date')
            start_time = data.get('start_time')
            end_time = data.get('end_time')
            location = data.get('location')
            category = data.get('category')
            organizer = data.get('organizer')
            event_type = data.get('event_type')
            speakers = data.get('speakers')
            agenda = data.get('agenda')
            registration_link = data.get('registration_link')
            ticket_info = data.get('ticket_info')
            banner = data.get('banner')
            requirements = data.get('requirements')

        cursor.execute("""
            UPDATE events SET title = ?, description = ?, date = ?, start_time = ?, end_time
= ?, location = ?, category = ?, organizer = ?, banner = ?, event_type = ?, speakers = ?,
agenda = ?, registration_link = ?, ticket_info = ?, requirements = ?
            WHERE id = ?
        """, (title, description, date, start_time, end_time, location, category, organizer,
banner, event_type, speakers, agenda, registration_link, ticket_info, requirements, id))
        conn.commit()
        log_action('Event Updated', f"Event '{title}' (ID: {id}) updated", 'admin@example.com')
        conn.close()
        return jsonify({'message': 'Event updated successfully'})

    if request.method == 'DELETE':
        cursor.execute("DELETE FROM events WHERE id = ?", (id,))
        if cursor.rowcount > 0:
            conn.commit()
            log_action('Event Deleted', f"Event ID: {id} deleted", 'admin@example.com')
            conn.close()
            return jsonify({'message': 'Event deleted successfully'})
        else:
            conn.close()
            return jsonify({'message': 'Event not found'}), 404

@app.route('/events/<int:id>/register', methods=['POST'])
def register_event(id):
    print("Register endpoint hit")
    data = request.get_json()
    if not data:
        return jsonify({'error': 'No data provided'}), 400

    name = data.get('name')
    email = data.get('email')
    phone = data.get('phone')
    ticket_type = data.get('ticket_type')
    quantity = data.get('quantity', 1)  # Default to 1 if not provided

    if not all([name, email, ticket_type]):
        return jsonify({'error': 'Missing required fields: name, email, ticket_type'}), 400

    print(f"Registering for event {id} with email: '{email}'")

    conn = get_db_connection()
    cursor = conn.cursor()
    cursor.execute("""
        INSERT INTO registrations (event_id, name, email, phone, ticket_type, quantity,
payment_status)
        VALUES (?, ?, ?, ?, ?, ?, 'paid')
    """, (id, name, email, phone, ticket_type, quantity))
    conn.commit()
    conn.close()
    return jsonify({'message': 'Registration successful'}), 201

@app.route('/events/<int:id>/checkin', methods=['POST'])
def checkin_event(id):
    print("Checkin endpoint hit")
    data = request.get_json()
    if not data:
        return jsonify({'error': 'No data provided'}), 400

    email = data.get('email')
    print(f"Email received: '{email}' for event {id}")
    if not email:
        return jsonify({'error': 'Email is required'}), 400

    # Trim whitespace from email
    email = email.strip()

    conn = get_db_connection()
    cursor = conn.cursor()

    # First, check if registration exists (case-insensitive and trimmed)
    cursor.execute("""
        SELECT id FROM registrations
        WHERE event_id = ? AND LOWER(TRIM(email)) = LOWER(?)
    """, (id, email))
    registration = cursor.fetchone()
    print(f"Registration found: {registration}")

    if not registration:
        conn.close()
        return jsonify({'error': 'No registration found for this email and event'}), 404

    # Update checked_in (case-insensitive and trimmed)
    cursor.execute("""
        UPDATE registrations
        SET checked_in = TRUE
        WHERE event_id = ? AND LOWER(TRIM(email)) = LOWER(?)
    """, (id, email))
    print(f"Rowcount after update: {cursor.rowcount}")
    conn.commit()
    conn.close()
    return jsonify({'message': 'Check-in successful'}), 200

@app.route('/user/registrations', methods=['POST'])
def get_user_registrations():
    print("User registrations endpoint hit")
    data = request.get_json()
    if not data:
        return jsonify({'error': 'No data provided'}), 400

    email = data.get('email')
    if not email:
        return jsonify({'error': 'Email is required'}), 400

    # Trim whitespace from email
    email = email.strip()

    conn = get_db_connection()
    cursor = conn.cursor()

    # Join registrations and events to get full event details with check-in status
    cursor.execute("""
        SELECT r.id as registration_id, r.checked_in,
               e.id, e.banner, e.title, e.description, e.date, e.start_time, e.end_time,
               e.location, e.category, e.organizer, e.event_type, e.speakers, e.agenda,
               e.registration_link, e.ticket_info, e.requirements
        FROM registrations r
        JOIN events e ON r.event_id = e.id
        WHERE LOWER(TRIM(r.email)) = LOWER(?)
        ORDER BY e.date DESC
    """, (email,))
    rows = cursor.fetchall()
    conn.close()

    # Format as array of event objects with checked_in
    registrations = []
    for row in rows:
        event = dict(row)
        # Remove registration_id from top level, keep checked_in
        event.pop('registration_id', None)
        registrations.append(event)

    return jsonify(registrations), 200

@app.route('/user/profile', methods=['PUT'])
def update_user_profile():
    print("Update user profile endpoint hit")
    data = request.get_json()
    if not data:
        return jsonify({'error': 'No data provided'}), 400

    email = data.get('email')
    name = data.get('name')
    phone = data.get('phone')

    if not email:
        return jsonify({'error': 'Email is required'}), 400
    # Trim whitespace
    email = email.strip()

    conn = get_db_connection()
    cursor = conn.cursor()

    # Update user profile
    cursor.execute("""
        UPDATE users SET username = ?, phone = ? WHERE LOWER(TRIM(email)) = LOWER(?)
    """, (name, phone, email))
    conn.commit()
    conn.close()

    return jsonify({'message': 'Profile updated successfully'}), 200

@app.route('/api/stats', methods=['GET'])
def get_stats():
    print("Stats endpoint hit")
    conn = get_db_connection()
    cursor = conn.cursor()

    # Total Events
    cursor.execute("SELECT COUNT(*) as totalEvents FROM events")
    totalEvents = cursor.fetchone()['totalEvents']

    # Total Users
    cursor.execute("SELECT COUNT(*) as totalUsers FROM users")
    totalUsers = cursor.fetchone()['totalUsers']

    # Total Active Users (distinct emails with paid registrations)
    cursor.execute("SELECT COUNT(DISTINCT email) as totalActiveUsers FROM registrations WHERE
payment_status = 'paid'")
    totalActiveUsersRow = cursor.fetchone()
    totalActiveUsers = totalActiveUsersRow['totalActiveUsers'] if
totalActiveUsersRow['totalActiveUsers'] else 0

    # Total Tickets (sum of quantities for paid registrations)
    cursor.execute("SELECT SUM(quantity) as totalTickets FROM registrations WHERE
payment_status = 'paid'")
    totalTicketsRow = cursor.fetchone()
    totalTickets = totalTicketsRow['totalTickets'] if totalTicketsRow['totalTickets'] else 0

    # Total Upcoming Events
    cursor.execute("SELECT COUNT(*) as totalUpcomingEvents FROM events WHERE date >=
date('now')")
    totalUpcomingEvents = cursor.fetchone()['totalUpcomingEvents']

    # Total Revenue (assuming $50 per ticket)
    totalRevenue = totalTickets * 50

    conn.close()
    return jsonify({
        'totalEvents': totalEvents,
        'totalUsers': totalUsers,
        'totalActiveUsers': totalActiveUsers,
        'totalTickets': totalTickets,
        'totalUpcomingEvents': totalUpcomingEvents,
        'totalRevenue': totalRevenue
    })
```

```
@app.route('/api/recent-activities', methods=['GET'])
def get_recent_activities():
    print("Recent activities endpoint hit")
    conn = get_db_connection()
    cursor = conn.cursor()

    # Fetch recent registrations as activities (last 10)
    cursor.execute("""
        SELECT r.id, r.name, r.email, r.ticket_type, r.quantity, r.payment_status, e.title as
event_title, r.created_at
        FROM registrations r
        JOIN events e ON r.event_id = e.id
        ORDER BY r.id DESC
        LIMIT 10
    """)
    rows = cursor.fetchall()
    activities = []
    for row in rows:
        activity = {
            'id': row['id'],
            'action': 'Ticket Purchased' if row['payment_status'] == 'paid' else 'Registration
Pending',
            'details': f"{row['name']} ({row['email']}) registered for '{row['event_title']}'
- {row['quantity']} {row['ticket_type']} ticket(s)",
            'timestamp': row['created_at'] or 'N/A'
        }
        activities.append(activity)

    conn.close()
    return jsonify(activities)

@app.route('/api/users', methods=['GET'])
def get_users():
    print("Users endpoint hit")
    conn = get_db_connection()
    cursor = conn.cursor()
    cursor.execute("SELECT id, username, email, phone, role, created_at FROM users")
    rows = cursor.fetchall()
    users = [dict(row) for row in rows]
    conn.close()
    return jsonify(users)

@app.route('/api/users/<int:id>', methods=['DELETE'])
def delete_user(id):
    print(f"Delete user endpoint hit for id: {id}")
    conn = get_db_connection()
    cursor = conn.cursor()
    cursor.execute("SELECT username FROM users WHERE id = ?", (id,))
    user = cursor.fetchone()
    if not user:
        conn.close()
        return jsonify({'error': 'User not found'}), 404
    username = user['username']
    cursor.execute("DELETE FROM users WHERE id = ?", (id,))
    conn.commit()
    log_action('User Deleted', f"User '{username}' (ID: {id}) deleted", 'admin@example.com')
    conn.close()
    return jsonify({'message': 'User deleted successfully'})

@app.route('/api/users/<int:id>', methods=['PUT'])
def update_user(id):
    print(f"Update user endpoint hit for id: {id}")
    data = request.get_json()
    if not data:
        return jsonify({'error': 'No data provided'}), 400

    username = data.get('username')
    email = data.get('email')
    phone = data.get('phone')
    role = data.get('role')

    if not all([username, email, role]):
        return jsonify({'error': 'Missing required fields: username, email, role'}), 400

    conn = get_db_connection()
    cursor = conn.cursor()
    cursor.execute("""
        UPDATE users SET username = ?, email = ?, phone = ?, role = ?
        WHERE id = ?
    """, (username, email, phone, role, id))
    if cursor.rowcount == 0:
        conn.close()
        return jsonify({'error': 'User not found'}), 404
    conn.commit()
    log_action('User Updated', f"User '{username}' (ID: {id}) updated", 'admin@example.com')
    conn.close()
    return jsonify({'message': 'User updated successfully'})

@app.route('/api/tickets', methods=['GET'])
def get_tickets():
    print("Tickets endpoint hit")
    conn = get_db_connection()
    cursor = conn.cursor()

    # Fetch all registrations as tickets, joining with events for event details
    cursor.execute("""
        SELECT r.id, r.event_id, r.name, r.email, r.phone, r.ticket_type, r.quantity,
r.payment_status, r.checked_in, r.created_at,
               e.title as event_name
        FROM registrations r
        JOIN events e ON r.event_id = e.id
        ORDER BY r.created_at DESC
    """)
    rows = cursor.fetchall()
    tickets = []
    for row in rows:
        ticket = {
            'id': row['id'],
            'event_name': row['event_name'],
            'user_email': row['email'],
            'quantity': row['quantity'],
            'status': 'active' if row['payment_status'] == 'paid' else 'cancelled',
            'purchase_date': row['created_at']
        }
        tickets.append(ticket)

    conn.close()
    return jsonify(tickets)

@app.route('/api/audit-logs', methods=['GET'])
def get_audit_logs():
    print("Audit logs endpoint hit")
    conn = get_db_connection()
    cursor = conn.cursor()
    cursor.execute("SELECT id, action, details, admin_user, timestamp FROM audit_logs ORDER BY 
timestamp DESC LIMIT 50")
    rows = cursor.fetchall()
    logs = [dict(row) for row in rows]
    conn.close()
    return jsonify(logs)

@app.route('/api/speaker/events', methods=['GET'])
def get_speaker_events():
    print("Speaker events endpoint hit")
    # TODO: Implement proper authentication and filter by speaker ID
    # For now, return all events (can be filtered client-side)
    conn = get_db_connection()
    cursor = conn.cursor()
    cursor.execute("SELECT * FROM events")
    rows = cursor.fetchall()
    events = [dict(row) for row in rows]
    conn.close()
    return jsonify(events)

@app.route('/api/admin/settings', methods=['GET', 'POST'])
def admin_settings():
    print("Admin settings endpoint hit")
    # Basic admin check (in production, use JWT token verification)
    # For now, assume request is from admin (handled by frontend ProtectedAdminRoute)
    conn = get_db_connection()
    cursor = conn.cursor()

    if request.method == 'GET':
        cursor.execute("SELECT key, value FROM admin_settings")
        rows = cursor.fetchall()
        settings = {row['key']: row['value'] for row in rows}
        conn.close()
        return jsonify(settings)

    if request.method == 'POST':
        data = request.get_json()
        if not data:
            return jsonify({'error': 'No data provided'}), 400

        for key, value in data.items():
            cursor.execute("""
                INSERT OR REPLACE INTO admin_settings (key, value)
                VALUES (?, ?)
            """, (key, str(value)))
        conn.commit()
        log_action('Admin Settings Updated', f"Settings updated: {list(data.keys())}", 
'admin@example.com')
        conn.close()
        return jsonify({'message': 'Settings updated successfully'}), 200

@app.route('/uploads/<filename>')
def uploaded_file(filename):
    return send_from_directory('uploads', filename)

if __name__ == '__main__':
    app.run(debug=True)
```

/backend-auth/server.js

```javascript
const express = require('express');
const sqlite3 = require('sqlite3').verbose();
const jwt = require('jsonwebtoken');
const cors = require('cors');
const path = require('path');
require('dotenv').config();

const app = express();
const PORT = process.env.PORT || 3001;
const JWT_SECRET = process.env.JWT_SECRET || 'your-secret-key';

app.use(cors());
app.use(express.json());

// Database setup
const dbPath = path.join(__dirname, '..', 'backend', 'events.db');
const db = new sqlite3.Database(dbPath, (err) => {
  if (err) {
    console.error('Error opening database:', err.message);
  } else {
    console.log('Connected to the SQLite database.');
  }
});

// Create tables if they don't exist
db.serialize(() => {
  db.run(`
    CREATE TABLE IF NOT EXISTS users (
      id INTEGER PRIMARY KEY AUTOINCREMENT,
      username TEXT UNIQUE NOT NULL,
      email TEXT UNIQUE NOT NULL,
      password TEXT NOT NULL,
      phone TEXT,
      role TEXT DEFAULT 'user',
      created_at DATETIME DEFAULT CURRENT_TIMESTAMP
    )
  `);

  // Ensure role and phone columns exist in users table for existing DB
  db.all("PRAGMA table_info(users)", (err, rows) => {
    if (err) {
      console.error('Error checking table info:', err);
    } else {
      if (!rows || !rows.some(row => row.name === 'role')) {
        db.run("ALTER TABLE users ADD COLUMN role TEXT DEFAULT 'user'", (err) => {
          if (err) {
            console.error('Error adding role column:', err);
          } else {
            console.log('Added role column to users table');
          }
        });
      }
      if (!rows || !rows.some(row => row.name === 'phone')) {
        db.run("ALTER TABLE users ADD COLUMN phone TEXT", (err) => {
          if (err) {
            console.error('Error adding phone column:', err);
          } else {
            console.log('Added phone column to users table');
          }
        });
      }
    }
  });

  db.run(`
    CREATE TABLE IF NOT EXISTS admins (
      id INTEGER PRIMARY KEY AUTOINCREMENT,
      username TEXT UNIQUE NOT NULL,
      email TEXT UNIQUE NOT NULL,
      password TEXT NOT NULL,
      phone TEXT,
      created_at DATETIME DEFAULT CURRENT_TIMESTAMP
    )
  `);

  db.run(`
    CREATE TABLE IF NOT EXISTS speakers (
      id INTEGER PRIMARY KEY AUTOINCREMENT,
      username TEXT UNIQUE NOT NULL,
      email TEXT UNIQUE NOT NULL,
      password TEXT NOT NULL,
      bio TEXT,
      created_at DATETIME DEFAULT CURRENT_TIMESTAMP
    )
  `);

  db.run(`
    CREATE TABLE IF NOT EXISTS otps (
      email TEXT PRIMARY KEY,
      otp TEXT NOT NULL,
      expires_at DATETIME NOT NULL
    )
  `);

  db.run(`
    CREATE TABLE IF NOT EXISTS audit_logs (
      id INTEGER PRIMARY KEY AUTOINCREMENT,
      action TEXT NOT NULL,
      admin_user TEXT,
      timestamp DATETIME DEFAULT CURRENT_TIMESTAMP,
      details TEXT
    )
  `);

  // Ensure admin_user column exists in audit_logs table for existing DB
  db.all("PRAGMA table_info(audit_logs)", (err, rows) => {
    if (err) {
      console.error('Error checking audit_logs table info:', err);
    } else {
      const hasUser = rows.some(row => row.name === 'user');
      const hasAdminUser = rows.some(row => row.name === 'admin_user');
      if (hasUser && !hasAdminUser) {
        // Rename 'user' to 'admin_user'
        db.run("ALTER TABLE audit_logs ADD COLUMN admin_user_temp TEXT", (err) => {
          if (err) {
            console.error('Error adding temp column:', err);
          } else {
            db.run("UPDATE audit_logs SET admin_user_temp = user", (err) => {
              if (err) {
                console.error('Error copying data:', err);
              } else {
```

```
        db.run("ALTER TABLE audit_logs DROP COLUMN user", (err) => {
         if (err) {
          console.error('Error dropping old column:', err);
         } else {
          db.run("ALTER TABLE audit_logs RENAME COLUMN admin_user_temp TO admin_user",
(err) => {
           if (err) {
            console.error('Error renaming column:', err);
           } else {
            console.log('Renamed user column to admin_user in audit_logs table');
           }
          });
         }
        });
       }
      });
     }
    });
   }
  }
 });

 // Ensure phone column exists in admins table for existing DB
 db.all("PRAGMA table_info(admins)", (err, rows) => {
  if (err) {
   console.error('Error checking admins table info:', err);
  } else {
   if (!rows || !rows.some(row => row.name === 'phone')) {
    db.run("ALTER TABLE admins ADD COLUMN phone TEXT", (err) => {
     if (err) {
      console.error('Error adding phone column to admins:', err);
     } else {
      console.log('Added phone column to admins table');
     }
    });
   }
  }
 });
});

// Middleware to verify JWT
const verifyToken = (req, res, next) => {
 const token = req.headers['authorization']?.split(' ')[1];
 if (!token) return res.status(401).json({ error: 'Access denied' });

 jwt.verify(token, JWT_SECRET, (err, decoded) => {
  if (err) return res.status(403).json({ error: 'Invalid token' });
  req.user = decoded;
  next();
 });
};

// Signup endpoint
app.post('/signup', (req, res) => {
 const { username, email, password, role = 'user' } = req.body;

 if (!username || !email || !password) {
  return res.status(400).json({ error: 'Username, email, and password are required' });
 }

 let table;
```

```
  if (role === 'admin') {
    table = 'admins';
  } else if (role === 'speaker') {
    table = 'speakers';
  } else {
    table = 'users';
  }

  db.run(
    `INSERT INTO ${table} (username, email, password) VALUES (?, ?, ?)`,
    [username, email, password],
    function(err) {
      if (err) {
        if (err.message.includes('UNIQUE constraint failed')) {
          return res.status(409).json({ error: 'Username or email already exists' });
        }
        console.error('Signup error:', err);
        return res.status(500).json({ error: 'Internal server error' });
      }
      const action = role === 'admin' ? 'Admin Signup' : role === 'speaker' ? 'Speaker
Signup' : 'User Signup';
      logAuditEvent(action, username, `New ${role} account created: ${username} (${email})`);
      res.status(201).json({ message: 'User registered successfully' });
    }
  );
});

// Login endpoint
app.post('/login', (req, res) => {
  const { username, password, role = 'user' } = req.body;

  if (!username || !password) {
    return res.status(400).json({ error: 'Username and password are required' });
  }

  const table = role === 'admin' ? 'admins' : 'users';

  db.get(
    `SELECT * FROM ${table} WHERE username = ? OR email = ?`,
    [username, username],
    (err, user) => {
      if (err) {
        console.error('Login error:', err);
        return res.status(500).json({ error: 'Internal server error' });
      }
      if (!user) return res.status(401).json({ error: 'Invalid username or password' });

      if (password !== user.password) return res.status(401).json({ error: 'Invalid username
or password' });

      const token = jwt.sign(
        { id: user.id, username: user.username, email: user.email, role },
        JWT_SECRET,
        { expiresIn: '24h' }
      );

      const action = role === 'admin' ? 'Admin Login' : 'User Login';
      logAuditEvent(action, username, `Successful login as ${role}`);
      res.json({
        message: 'Login successful',
        token,
        user: { id: user.id, username: user.username, email: user.email, role }
      });
    }
  );
});

// Speaker login endpoint
app.post('/speaker/login', (req, res) => {
  const { username, password } = req.body;
  const role = 'speaker';

  if (!username || !password) {
    return res.status(400).json({ error: 'Username and password are required' });
  }

  const table = 'speakers';

  db.get(
    `SELECT * FROM ${table} WHERE username = ? OR email = ?`,
    [username, username],
    (err, user) => {
      if (err) {
        console.error('Speaker login error:', err);
        return res.status(500).json({ error: 'Internal server error' });
      }
      if (!user) return res.status(401).json({ error: 'Invalid username or password' });

      if (password !== user.password) return res.status(401).json({ error: 'Invalid username
or password' });

      const token = jwt.sign(
        { id: user.id, username: user.username, email: user.email, role },
        JWT_SECRET,
        { expiresIn: '24h' }
      );

      logAuditEvent('Speaker Login', username, `Successful login as speaker`);
      res.json({
        message: 'Login successful',
        token,
        user: { id: user.id, username: user.username, email: user.email, role }
      });
    }
  );
});

// Forgot password endpoint with OTP
app.post('/forgot-password', (req, res) => {
  const { email } = req.body;

  if (!email) {
    return res.status(400).json({ error: 'Email is required' });
  }

  db.get(`SELECT 1 as exists FROM (SELECT email FROM users WHERE email = ? UNION SELECT email
FROM admins WHERE email = ?)`, [email, email], (err, row) => {
    if (err) {
      console.error('Forgot password DB error:', err);
      return res.status(500).json({ error: 'Internal server error' });
    }
    if (row) {
      // Generate OTP
      const otp = Math.floor(100000 + Math.random() * 900000).toString();

      db.run(`INSERT OR REPLACE INTO otps (email, otp, expires_at) VALUES (?, ?,
datetime('now', '+10 minutes'))`, [email, otp], (err) => {
        if (err) {
          console.error('Forgot password DB insert error:', err);
          return res.status(500).json({ error: 'Failed to generate OTP' });
        }

        console.log(`OTP for ${email}: ${otp}`);
        res.json({ message: 'If an account with that email exists, OTP has been sent.' });
      });
    } else {
      // Generic response for security
      res.json({ message: 'If an account with that email exists, OTP has been sent.' });
    }
  });
});

// Get profile
app.get('/profile', verifyToken, (req, res) => {
  const { id, role } = req.user;
  const table = role === 'admin' ? 'admins' : role === 'speaker' ? 'speakers' : 'users';

  let query;
  if (role === 'admin') {
    query = `SELECT id, username, email, phone, created_at FROM ${table} WHERE id = ?`;
  } else if (role === 'speaker') {
    query = `SELECT id, username, email, bio, created_at FROM ${table} WHERE id = ?`;
  } else {
    query = `SELECT id, username, email, phone, role, created_at FROM ${table} WHERE id = ?`;
  }

  db.get(query, [id], (err, user) => {
    if (err) {
      console.error('Profile fetch error:', err);
      return res.status(500).json({ error: 'Internal server error' });
    }
    if (!user) return res.status(404).json({ error: 'User not found' });

    // Format created_at to ISO string for consistent frontend parsing
    if (user.created_at) {
      user.created_at = new Date(user.created_at).toISOString();
    }

    // Manually set role for admins and speakers since it's not in the table
    if (role === 'admin') {
      user.role = 'admin';
    } else if (role === 'speaker') {
      user.role = 'speaker';
    }

    res.json({ user });
  });
});

// Update profile
app.put('/profile', verifyToken, (req, res) => {
  const { id, role } = req.user;
  const { username, email, phone, bio, currentPassword, newPassword } = req.body;
  const table = role === 'admin' ? 'admins' : role === 'speaker' ? 'speakers' : 'users';

  // Check for uniqueness if username or email is being changed
  let checkQuery = '';
  let checkParams = [];
  let hasUsernameOrEmailChange = false;
  if (username !== undefined) {
    checkQuery += 'username = ?';
    checkParams.push(username);
    hasUsernameOrEmailChange = true;
  }
  if (email !== undefined) {
    if (checkQuery) checkQuery += ' OR ';
    checkQuery += 'email = ?';
    checkParams.push(email);
    hasUsernameOrEmailChange = true;
  }

  const performUniquenessCheck = (callback) => {
    if (!hasUsernameOrEmailChange) {
      callback(null, null); // No check needed
      return;
    }
    db.get(`SELECT id FROM ${table} WHERE (${checkQuery}) AND id != ?`, [...checkParams, id],
(err, existing) => {
      callback(err, existing);
    });
  };

  performUniquenessCheck((err, existing) => {
    if (err) {
      console.error('Uniqueness check error:', err);
      return res.status(500).json({ error: 'Internal server error' });
    }
    if (existing) {
      return res.status(409).json({ error: 'Username or email already exists' });
    }

    // If newPassword is provided, require and verify currentPassword
    if (newPassword !== undefined && newPassword !== '') {
      if (!currentPassword) {
        return res.status(400).json({ error: 'Current password is required to change
password' });
      }
      db.get(`SELECT password FROM ${table} WHERE id = ?`, [id], (err, user) => {
        if (err) {
          console.error('Password verification error:', err);
          return res.status(500).json({ error: 'Internal server error' });
        }
        if (!user || user.password !== currentPassword) {
          return res.status(401).json({ error: 'Invalid current password' });
        }
        performUpdate();
      });
    } else {
      // No password change, proceed directly
      performUpdate();
    }
```

```
    function performUpdate() {
      let updateQuery = '';
      let updateParams = [];
      let hasChanges = false;

      if (username !== undefined) {
        updateQuery += 'username = ?, ';
        updateParams.push(username);
        hasChanges = true;
      }
      if (email !== undefined) {
        updateQuery += 'email = ?, ';
        updateParams.push(email);
        hasChanges = true;
      }
      if (phone !== undefined && phone !== '') {
        updateQuery += 'phone = ?, ';
        updateParams.push(phone);
        hasChanges = true;
      } else if (phone === '') {
        // Explicitly set to null/empty if cleared
        updateQuery += 'phone = ?, ';
        updateParams.push(null);
        hasChanges = true;
      }
      if (bio !== undefined && bio !== '') {
        updateQuery += 'bio = ?, ';
        updateParams.push(bio);
        hasChanges = true;
      } else if (bio === '') {
        // Explicitly set to null/empty if cleared
        updateQuery += 'bio = ?, ';
        updateParams.push(null);
        hasChanges = true;
      }
      if (newPassword !== undefined && newPassword !== '') {
        updateQuery += 'password = ?, ';
        updateParams.push(newPassword);
        hasChanges = true;
      }

      if (!hasChanges) {
        return res.json({ message: 'No changes made' });
      }

      if (updateQuery) {
        updateQuery = updateQuery.slice(0, -2); // Remove trailing comma and space
      }

      db.run(`UPDATE ${table} SET ${updateQuery} WHERE id = ?`, [...updateParams, id],
function(err) {
        if (err) {
          console.error('Profile update error:', err);
          return res.status(500).json({ error: 'Internal server error' });
        }
        if (this.changes === 0) {
          return res.json({ message: 'No changes made' });
        }
        logAuditEvent('Profile Update', req.user.username, `Updated profile information`);
        res.json({ message: 'Profile updated successfully' });
      });
    }
```

```
  });
});

// Delete profile
app.delete('/profile', verifyToken, (req, res) => {
  const { id, role } = req.user;
  const { confirm } = req.body;
  const table = role === 'admin' ? 'admins' : role === 'speaker' ? 'speakers' : 'users';

  if (confirm !== 'DELETE') {
    return res.status(400).json({ error: 'Confirmation required. Type "DELETE" to confirm.' });
  }

  db.run(`DELETE FROM ${table} WHERE id = ?`, [id], function(err) {
    if (err) {
      console.error('Profile delete error:', err);
      return res.status(500).json({ error: 'Internal server error' });
    }
    if (this.changes === 0) {
      return res.status(404).json({ error: 'User not found' });
    }
    logAuditEvent('Account Deletion', req.user.username, `Account deleted by user`);
    res.json({ message: 'Account deleted successfully' });
  });
});

// Admin endpoints
app.get('/admin/users', verifyToken, (req, res) => {
  if (req.user.role !== 'admin') {
    return res.status(403).json({ error: 'Admin access required' });
  }

  db.all(`SELECT id, username, email, phone, role, created_at FROM users`, (err, users) => {
    if (err) {
      console.error('Error fetching users:', err);
      return res.status(500).json({ error: 'Internal server error' });
    }
    res.json(users);
  });
});

app.get('/admin/total-users', verifyToken, (req, res) => {
  if (req.user.role !== 'admin') {
    return res.status(403).json({ error: 'Admin access required' });
  }

  const query = `
    SELECT
      (SELECT COUNT(*) FROM users) +
      (SELECT COUNT(*) FROM admins) as total_users
  `;

  db.get(query, (err, row) => {
    if (err) {
      console.error('Error fetching total users:', err);
      return res.status(500).json({ error: 'Internal server error' });
    }
    res.json({ total_users: row.total_users });
  });
});
```

```
app.get('/admin/registrations', verifyToken, (req, res) => {
  if (req.user.role !== 'admin') {
    return res.status(403).json({ error: 'Admin access required' });
  }

  const query = `
    SELECT r.id, r.name, r.email, r.phone, r.ticket_type, r.checked_in,
           e.title as event_title, e.date as event_date
    FROM registrations r
    JOIN events e ON r.event_id = e.id
    ORDER BY e.date DESC, r.name ASC
  `;

  db.all(query, (err, registrations) => {
    if (err) {
      console.error('Error fetching registrations:', err);
      return res.status(500).json({ error: 'Internal server error' });
    }
    res.json(registrations);
  });
});

// Admin endpoint to get single user by ID
app.get('/admin/users/:id', verifyToken, (req, res) => {
  if (req.user.role !== 'admin') {
    return res.status(403).json({ error: 'Admin access required' });
  }

  const { id } = req.params;

  db.get(`SELECT id, username, email, phone, role, created_at FROM users WHERE id = ?`, [id],
(err, user) => {
    if (err) {
      console.error('Error fetching user:', err);
      return res.status(500).json({ error: 'Internal server error' });
    }
    if (!user) {
      return res.status(404).json({ error: 'User not found' });
    }
    res.json(user);
  });
});

// Admin endpoint to update user
app.put('/admin/users/:id', verifyToken, (req, res) => {
  if (req.user.role !== 'admin') {
    return res.status(403).json({ error: 'Admin access required' });
  }

  const { id } = req.params;
  const { username, email, phone, role } = req.body;

  // Check for uniqueness if username or email is being changed
  let checkQuery = '';
  let checkParams = [];
  let hasUsernameOrEmailChange = false;
  if (username !== undefined) {
    checkQuery += 'username = ?';
    checkParams.push(username);
    hasUsernameOrEmailChange = true;
  }
  if (email !== undefined) {
    if (checkQuery) checkQuery += ' OR ';
    checkQuery += 'email = ?';
    checkParams.push(email);
    hasUsernameOrEmailChange = true;
  }

  const performUniquenessCheck = (callback) => {
    if (!hasUsernameOrEmailChange) {
      callback(null, null);
      return;
    }
    db.get(`SELECT id FROM users WHERE (${checkQuery}) AND id != ?`, [...checkParams, id],
(err, existing) => {
      callback(err, existing);
    });
  };

  performUniquenessCheck((err, existing) => {
    if (err) {
      console.error('Uniqueness check error:', err);
      return res.status(500).json({ error: 'Internal server error' });
    }
    if (existing) {
      return res.status(409).json({ error: 'Username or email already exists' });
    }

    // Build update query
    let updateQuery = '';
    let updateParams = [];
    let hasChanges = false;

    if (username !== undefined) {
      updateQuery += 'username = ?, ';
      updateParams.push(username);
      hasChanges = true;
    }
    if (email !== undefined) {
      updateQuery += 'email = ?, ';
      updateParams.push(email);
      hasChanges = true;
    }
    if (phone !== undefined) {
      updateQuery += 'phone = ?, ';
      updateParams.push(phone || null);
      hasChanges = true;
    }
    if (role !== undefined) {
      updateQuery += 'role = ?, ';
      updateParams.push(role);
      hasChanges = true;
    }

    if (!hasChanges) {
      return res.json({ message: 'No changes made' });
    }

    updateQuery = updateQuery.slice(0, -2); // Remove trailing comma and space
```

```
    db.run(`UPDATE users SET ${updateQuery} WHERE id = ?`, [...updateParams, id], function(err)
{
      if (err) {
        console.error('User update error:', err);
        return res.status(500).json({ error: 'Internal server error' });
      }
      if (this.changes === 0) {
        return res.status(404).json({ error: 'User not found' });
      }
      logAuditEvent('Admin User Update', req.user.username, `Updated user ID ${id}: ${username
|| email || phone || role}`);
      res.json({ message: 'User updated successfully' });
    });
  });
});

// Admin endpoint to delete user
app.delete('/admin/users/:id', verifyToken, (req, res) => {
  if (req.user.role !== 'admin') {
    return res.status(403).json({ error: 'Admin access required' });
  }

  const { id } = req.params;
  const { confirm } = req.body;

  if (confirm !== 'DELETE') {
    return res.status(400).json({ error: 'Confirmation required. Type "DELETE" to confirm.' });
  }

  // Prevent deleting self (basic check, assuming admin is in admins table)
  if (parseInt(id) === req.user.id) {
    return res.status(400).json({ error: 'Cannot delete your own account' });
  }

  db.run(`DELETE FROM users WHERE id = ?`, [id], function(err) {
    if (err) {
      console.error('User delete error:', err);
      return res.status(500).json({ error: 'Internal server error' });
    }
    if (this.changes === 0) {
      return res.status(404).json({ error: 'User not found' });
    }
    logAuditEvent('Admin User Deletion', req.user.username, `Deleted user ID ${id}`);
    res.json({ message: 'User deleted successfully' });
  });
});

// Admin endpoint to get audit logs
app.get('/admin/audit-logs', verifyToken, (req, res) => {
  if (req.user.role !== 'admin') {
    return res.status(403).json({ error: 'Admin access required' });
  }

  db.all(`SELECT id, action, admin_user, timestamp, details FROM audit_logs ORDER BY timestamp
DESC`, (err, logs) => {
    if (err) {
      console.error('Error fetching audit logs:', err);
      return res.status(500).json({ error: 'Internal server error' });
    }
    res.json(logs);
  });
});
```

```
// Helper function to log audit events
const logAuditEvent = (action, user, details) => {
  const now = new Date();
  const istOffsetMs = 5.5 * 60 * 60 * 1000; // UTC+5:30 in milliseconds
  const istTime = new Date(now.getTime() + istOffsetMs);

  const pad = (num) => num.toString().padStart(2, '0');
  const year = istTime.getFullYear();
  const month = pad(istTime.getMonth() + 1);
  const day = pad(istTime.getDate());
  const hour = pad(istTime.getHours());
  const minute = pad(istTime.getMinutes());
  const second = pad(istTime.getSeconds());

  const istTimestamp = `${year}-${month}-${day} ${hour}:${minute}:${second}`;

  db.run(`INSERT INTO audit_logs (action, admin_user, details, timestamp) VALUES (?, ?, ?, ?)`,
[action, user, details, istTimestamp], (err) => {
    if (err) {
      console.error('Error logging audit event:', err);
    }
  });
};

app.listen(PORT, () => {
  console.log(`Auth server running on port ${PORT}`);
});
```

CHAPTER - 10

TESTING

System testing is a crucial phase in the software development life cycle where the complete integrated system is tested to evaluate its compliance with the specified requirements. The objective of system testing in the **Plan Events Application** is to ensure that all components — frontend, backend, and database — work together as intended, providing a seamless experience for users and administrators.

10.1 Types of Testing Performed

The **Plan Events Application** underwent various types of testing to ensure that all modules and components function accurately and efficiently. Each testing type was performed at different stages of development to identify and resolve issues early, ensuring a stable and reliable system.

Additionally, the testing process was designed to simulate real-world scenarios and user interactions to evaluate the system's performance under practical conditions. Various test cases were created for both functional and non-functional aspects, covering everything from event creation and registration to authentication and data security. Continuous testing and feedback loops were implemented throughout the development cycle to detect defects early, improve overall system quality, and ensure that the final product meets the expectations of users, administrators, and stakeholders effectively.

System testing verified that:

- The application performs as expected under different user roles (Admin, Organizer, Participant).
- Data flows correctly between the frontend (React), backend (Flask/Node.js), and database (SQLite).
- All components interact seamlessly without data loss or functional interruption.
- The system meets non-functional requirements such as performance, security, and scalability.

10.1.1 Unit Testing

Unit testing involves testing the smallest units or individual components of the software to ensure that each part functions correctly. In the Plan Events Application, unit tests were performed on modules such as **User Authentication**, **Event Creation**, **Registration Form Validation**, and **Database CRUD operations**. For example, the backend Python functions were tested using frameworks like unit test and pytest to confirm that:

- User login and signup APIs return expected responses.
- Event details are correctly stored and retrieved from the database.

- Validation errors (e.g., empty fields, invalid emails) are properly handled.

This step ensured that all individual components worked accurately before they were integrated.

10.1.2 Integration Testing

Integration testing focuses on verifying the interfaces and interactions between integrated modules. It ensures that data flows smoothly from one module to another without loss or corruption.
In the Plan Events Application, integration testing was conducted between:

- The **Frontend (React)** and **Backend (Flask)** to confirm API communication and response handling.
- The **Backend** and **Database (SQLite)** to verify proper data storage, updates, and retrieval.
- The **Authentication Server (Node.js)** and **Main Flask Application** to validate user token verification and session management.

Test scenarios included registering a new user, logging in, creating an event, and verifying that all data were correctly linked and displayed on the frontend.

10.1.3 System Testing

System Testing is the process of testing the complete, integrated system to evaluate its compliance with the specified requirements. It ensures that all components, modules, and functionalities of the application work together as a unified whole. The main objective of system testing is to validate the end-to-end flow of the application and to verify that both functional and non-functional requirements are fully met.

In the case of the **Plan Events Application**, system testing was conducted after the successful completion of unit and integration testing phases. It focused on examining the behavior of the entire system, including user authentication, event management, registration processes, payment handling, and administrative controls. The system was tested as a complete entity to ensure that interactions between various subsystems, such as the frontend (React), backend (Flask and Node.js), and database (SQLite), operated smoothly and accurately.

10.1.4 Whitebox Testing

White Box Testing, also known as **structural testing** or **glass-box testing**, is a software testing technique that examines the internal structure, logic, and code implementation of the application. It focuses on verifying the internal operations of individual components to ensure that each function, loop, and conditional statement performs as intended. Unlike black box testing, which evaluates functionality from the user's perspective, white box testing provides insight into the internal workings of the system and allows testers to optimize the efficiency and security of the code.

In the **Plan Events Application**, white box testing was primarily conducted during the **unit testing** and **integration testing** phases. Developers examined the Python and Node.js code to verify that all functions executed correctly, handled inputs properly, and produced accurate outputs. This testing method ensured that data flowed correctly between modules such as user authentication, event creation, participant registration, and database management.

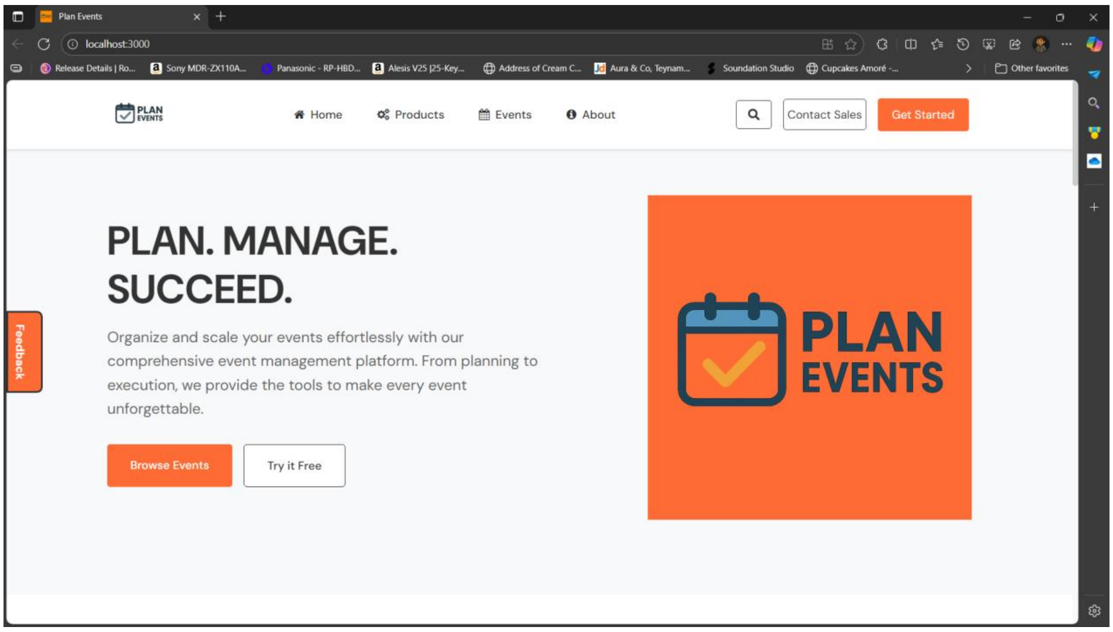
10.1.5 Blackbox Testing

Black Box Testing, also known as **behavioral testing** or **functional testing**, is a software testing technique that evaluates the functionality of an application without examining its internal code structure or logic. In this approach, the tester focuses solely on the input and output of the system — ensuring that it behaves as expected based on the defined requirements and user specifications.

For the **Plan Events Application**, black box testing was conducted to validate whether each feature and module worked according to the user's expectations and the project's functional requirements. Testers interacted with the system through its user interface (frontend) to verify that all actions, such as event creation, user registration, login authentication, participant management, and event notifications, produced the correct and expected results.

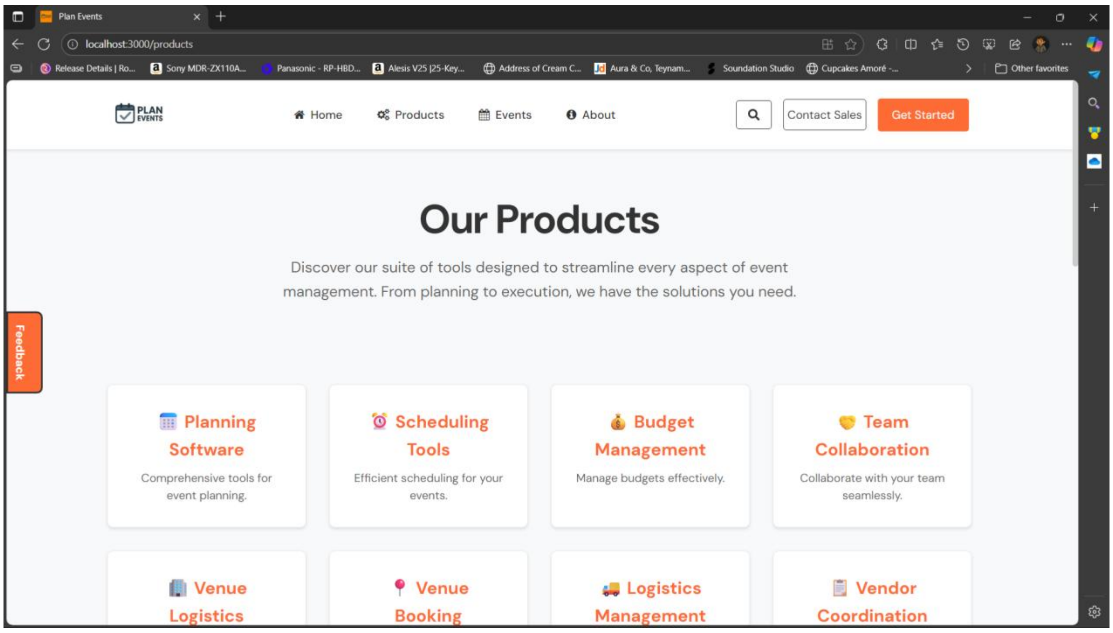
CHAPTER - 11

RESULTS AND SCREENSHOTS



PLAN. MANAGE. SUCCEED.

Organize and scale your events effortlessly with our comprehensive event management platform. From planning to execution, we provide the tools to make every event unforgettable.



Our Products

Discover our suite of tools designed to streamline every aspect of event management. From planning to execution, we have the solutions you need.

Planning Software
Comprehensive tools for event planning.

Scheduling Tools
Efficient scheduling for your events.

Budget Management
Manage budgets effectively.

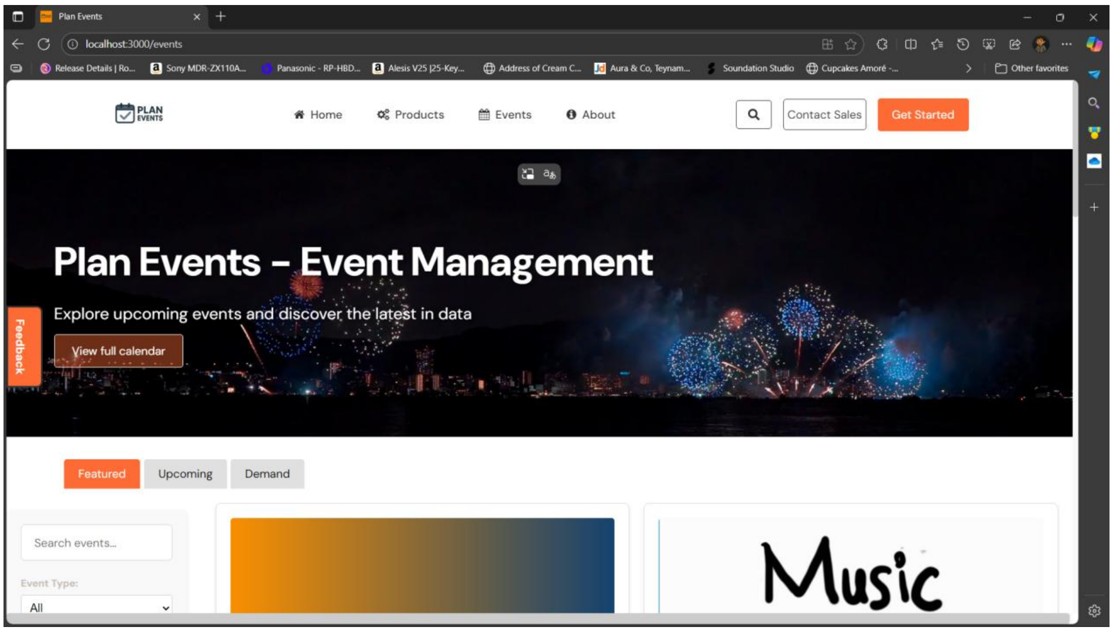
Team Collaboration
Collaborate with your team seamlessly.

Venue Logistics

Venue Booking

Logistics Management

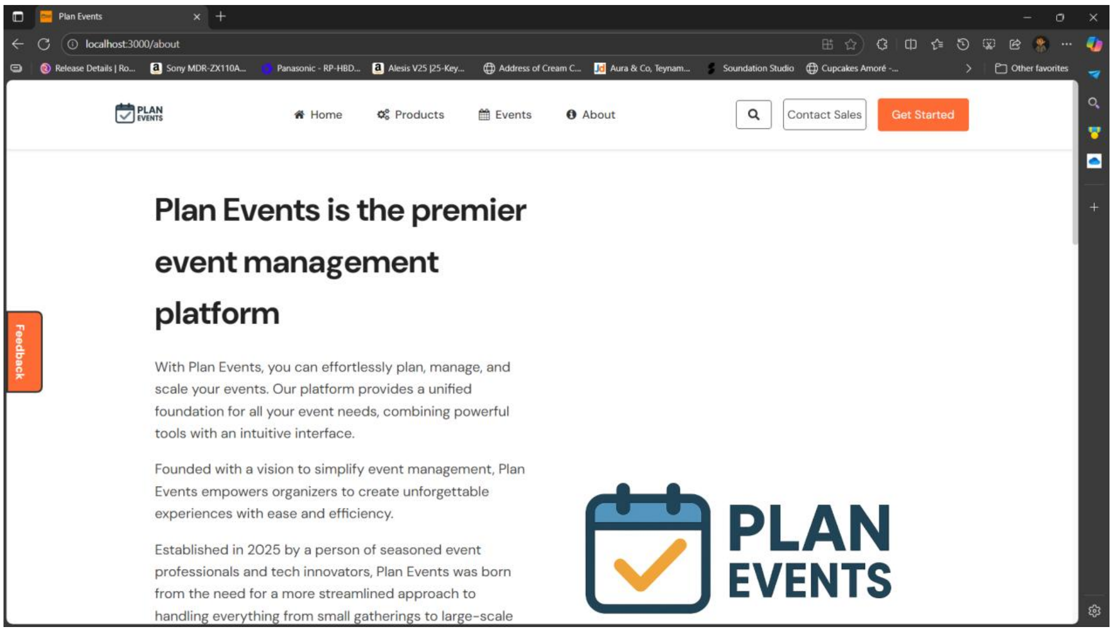
Vendor Coordination



Plan Events – Event Management

Explore upcoming events and discover the latest in data

View full calendar

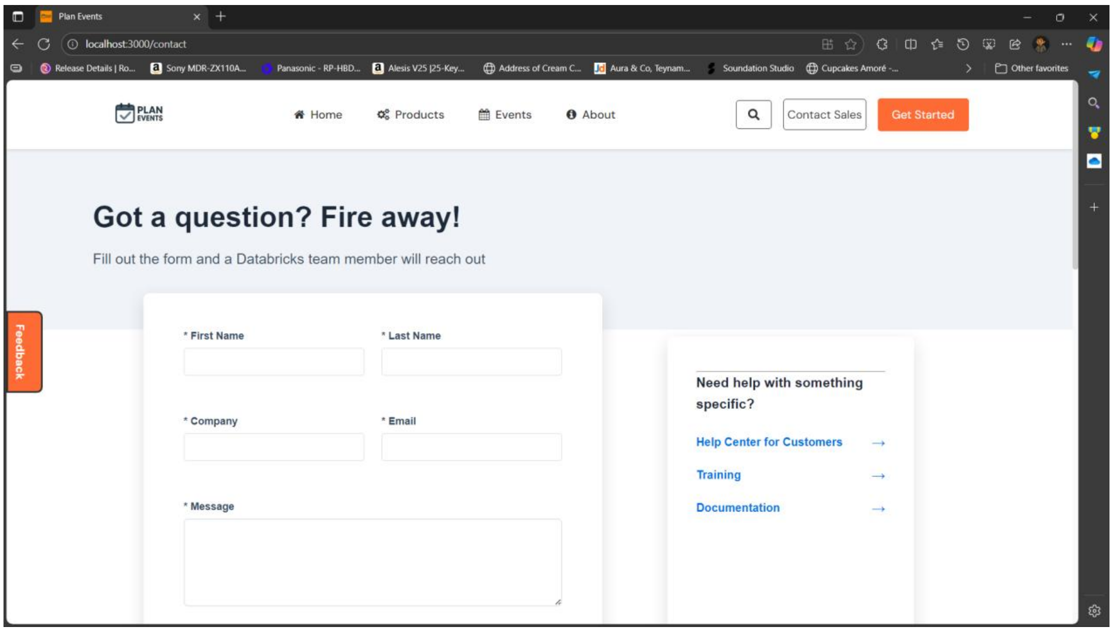


Plan Events is the premier event management platform

With Plan Events, you can effortlessly plan, manage, and scale your events. Our platform provides a unified foundation for all your event needs, combining powerful tools with an intuitive interface.

Founded with a vision to simplify event management, Plan Events empowers organizers to create unforgettable experiences with ease and efficiency.

Established in 2025 by a person of seasoned event professionals and tech innovators, Plan Events was born from the need for a more streamlined approach to handling everything from small gatherings to large-scale



Got a question? Fire away!

Fill out the form and a Databricks team member will reach out

* First Name

* Last Name

* Company

* Email

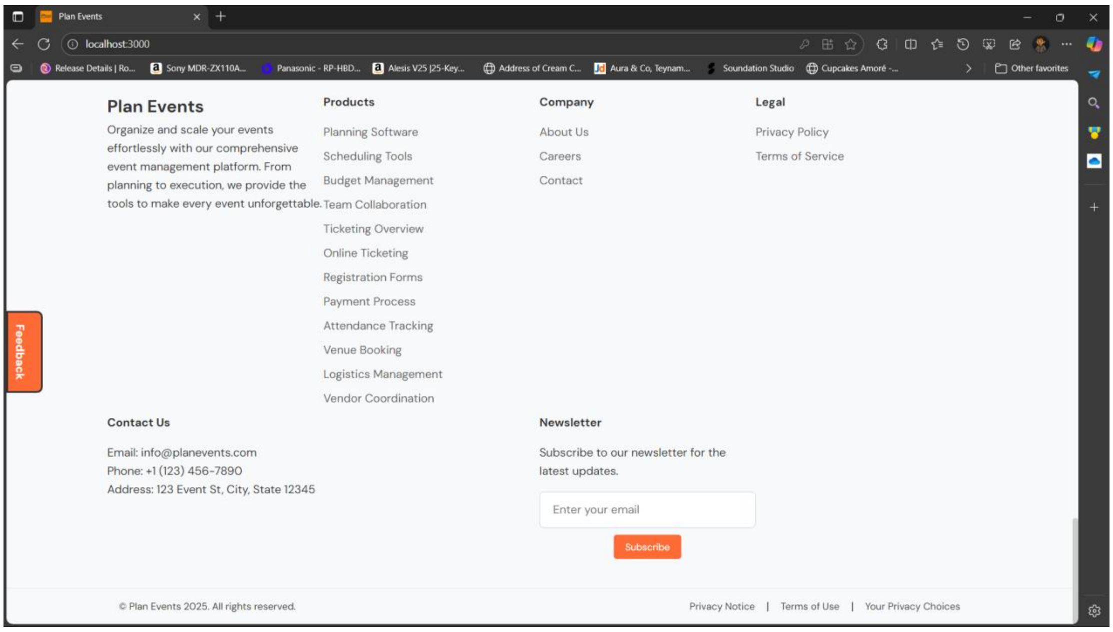
* Message

Need help with something specific?

Help Center for Customers →

Training →

Documentation →



Plan Events

Organize and scale your events effortlessly with our comprehensive event management platform. From planning to execution, we provide the tools to make every event unforgettable.

Products

Planning Software
Scheduling Tools
Budget Management
Team Collaboration
Ticketing Overview
Online Ticketing
Registration Forms
Payment Process
Attendance Tracking
Venue Booking
Logistics Management
Vendor Coordination

Company

About Us
Careers
Contact

Legal

Privacy Policy
Terms of Service

Contact Us

Email: info@planevents.com
Phone: +1 (123) 456-7890
Address: 123 Event St, City, State 12345

Newsletter

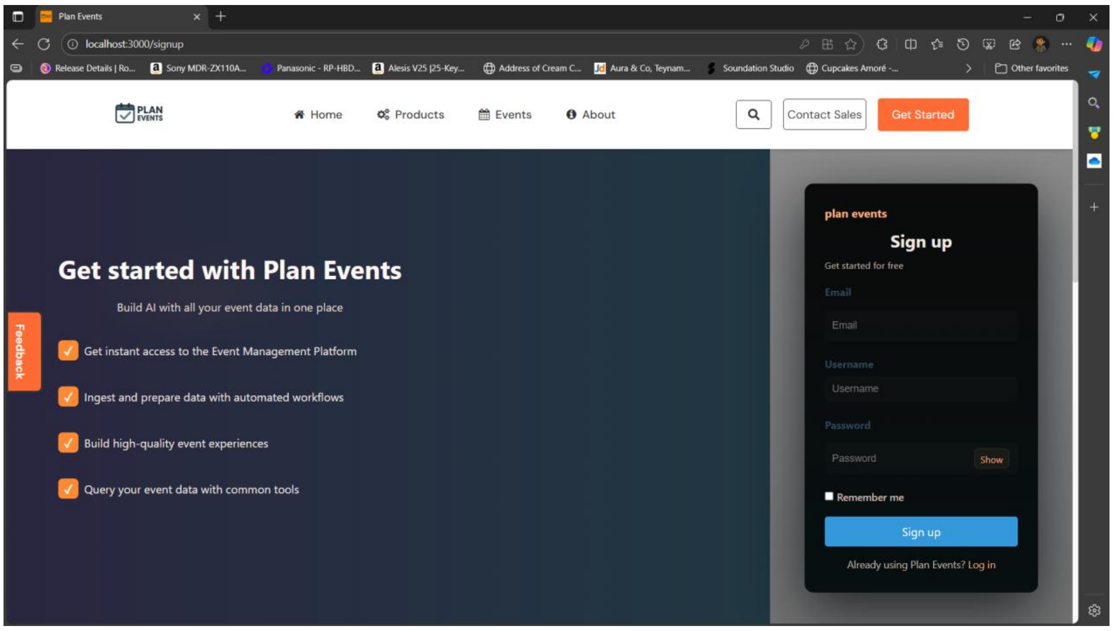
Subscribe to our newsletter for the latest updates.

Enter your email

Subscribe

© Plan Events 2025. All rights reserved.

Privacy Notice | Terms of Use | Your Privacy Choices



Get started with Plan Events

Build AI with all your event data in one place

- ✓ Get instant access to the Event Management Platform
- ✓ Ingest and prepare data with automated workflows
- ✓ Build high-quality event experiences
- ✓ Query your event data with common tools

plan events

Sign up

Get started for free

Email: Email

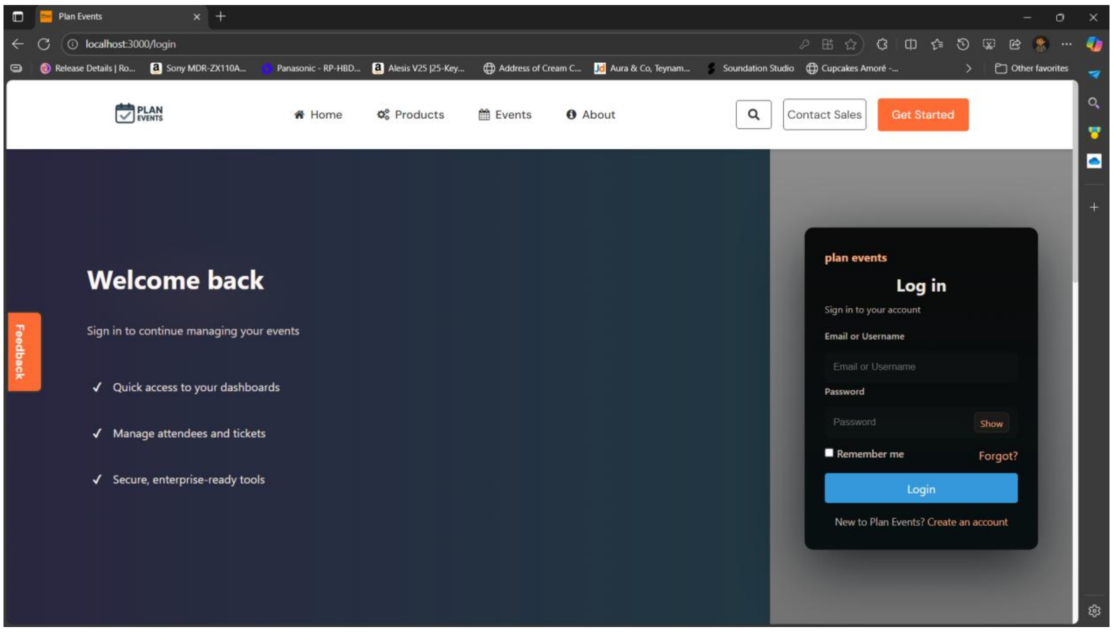
Username: Username

Password: Password Show

Remember me

Sign up

Already using Plan Events? Log in



Welcome back

Sign in to continue managing your events

- ✓ Quick access to your dashboards
- ✓ Manage attendees and tickets
- ✓ Secure, enterprise-ready tools

plan events

Log in

Sign in to your account

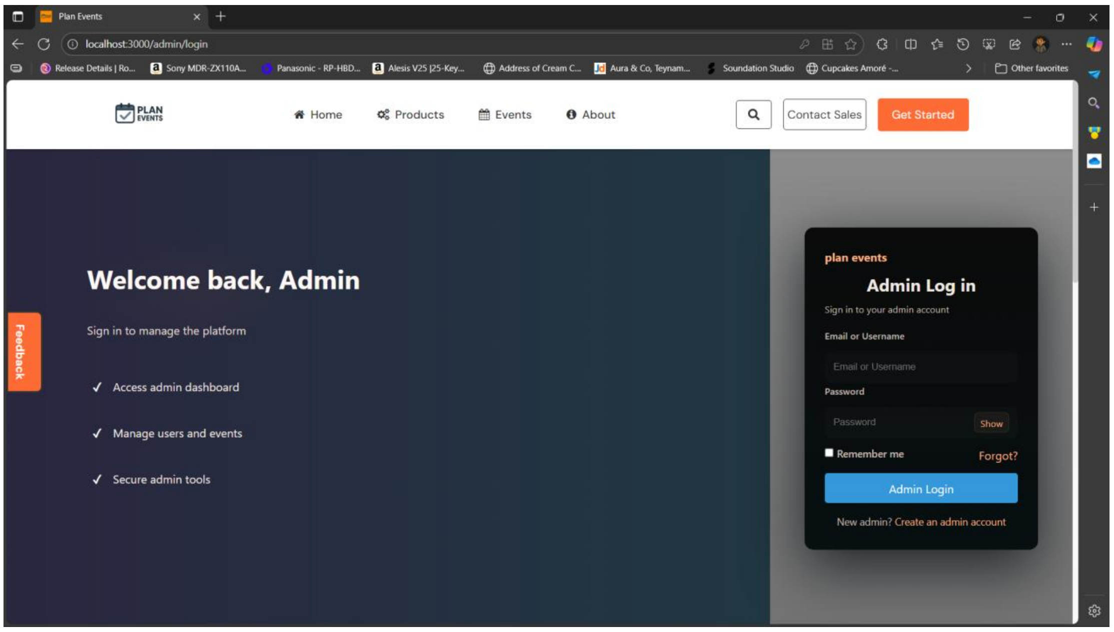
Email or Username: Email or Username

Password: Password Show

Remember me Forgot?

Login

New to Plan Events? Create an account



Welcome back, Admin

Sign in to manage the platform

✓ Access admin dashboard

✓ Manage users and events

✓ Secure admin tools

plan events

Admin Log in

Sign in to your admin account

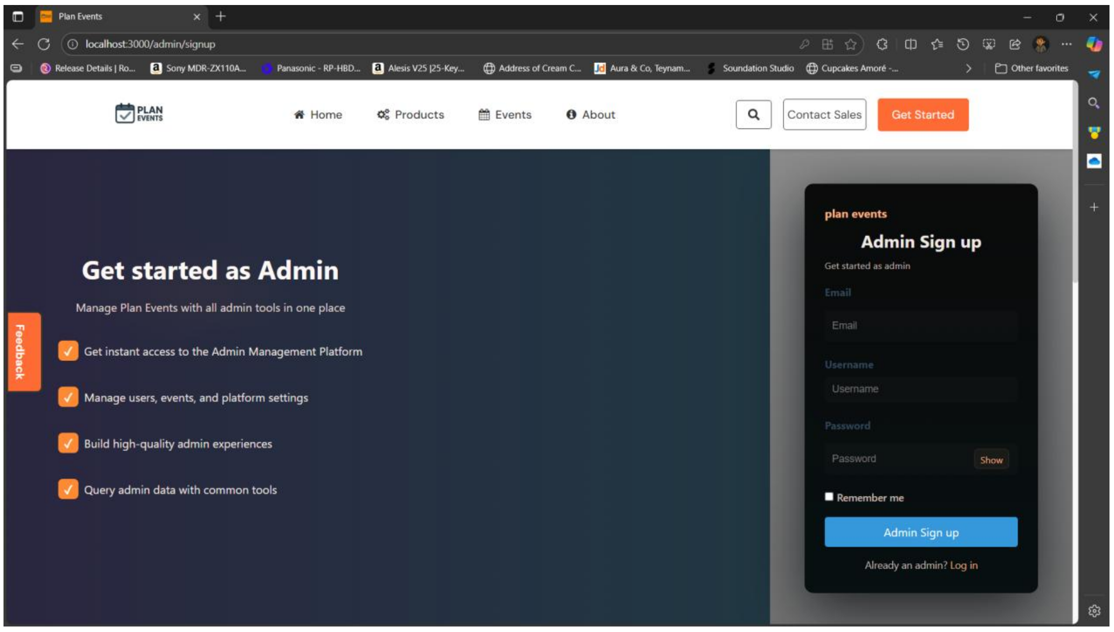
Email or Username: Email or Username

Password: Password Show

Remember me Forgot?

Admin Login

New admin? Create an admin account



Get started as Admin

Manage Plan Events with all admin tools in one place

✓ Get instant access to the Admin Management Platform

✓ Manage users, events, and platform settings

✓ Build high-quality admin experiences

✓ Query admin data with common tools

plan events

Admin Sign up

Get started as admin

Email: Email

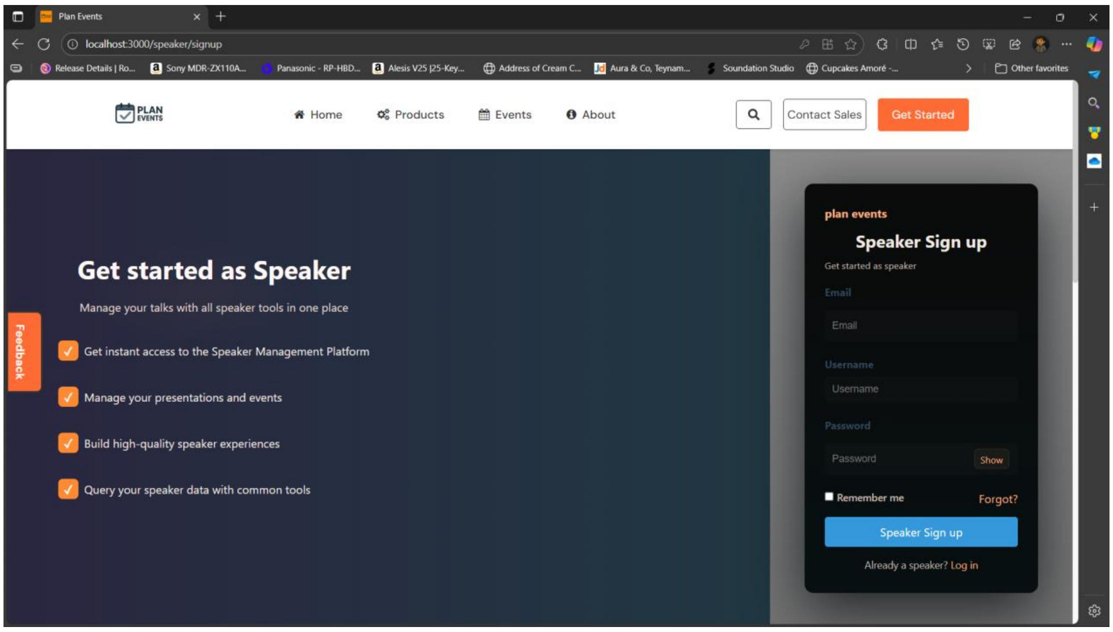
Username: Username

Password: Password Show

Remember me

Admin Sign up

Already an admin? Log in



localhost:3000/speaker/signup

Home Products Events About Contact Sales Get Started

Get started as Speaker

Manage your talks with all speaker tools in one place

- ✓ Get instant access to the Speaker Management Platform
- ✓ Manage your presentations and events
- ✓ Build high-quality speaker experiences
- ✓ Query your speaker data with common tools

plan events

Speaker Sign up

Get started as speaker

Email: Email

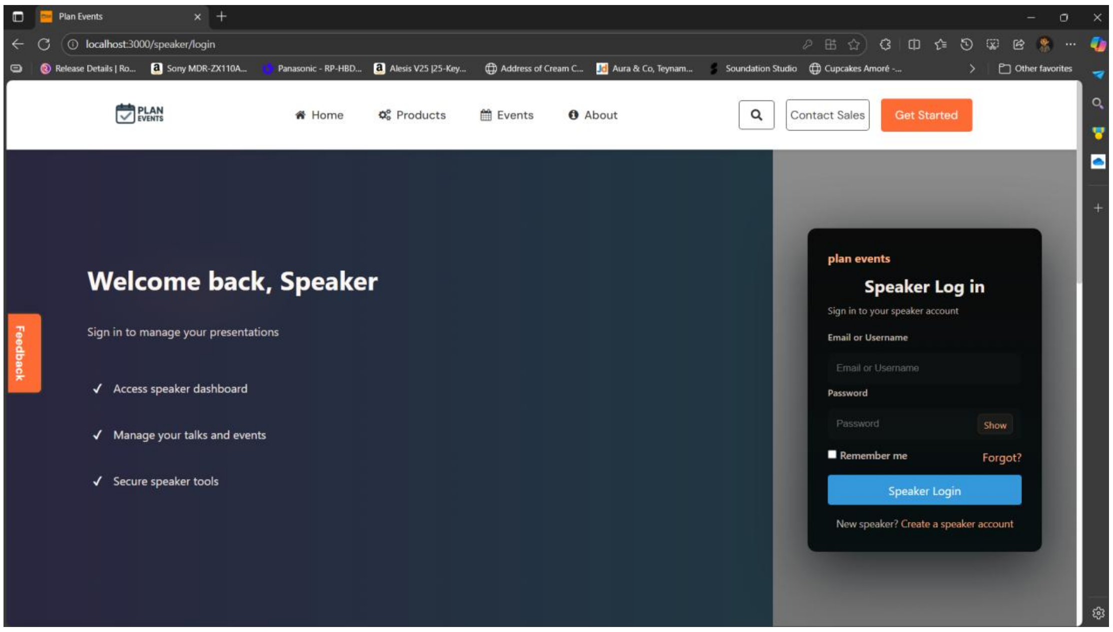
Username: Username

Password: Password (Show)

Remember me Forgot?

Speaker Sign up

Already a speaker? Log in



localhost:3000/speaker/login

Home Products Events About Contact Sales Get Started

Welcome back, Speaker

Sign in to manage your presentations

- ✓ Access speaker dashboard
- ✓ Manage your talks and events
- ✓ Secure speaker tools

plan events

Speaker Log in

Sign in to your speaker account

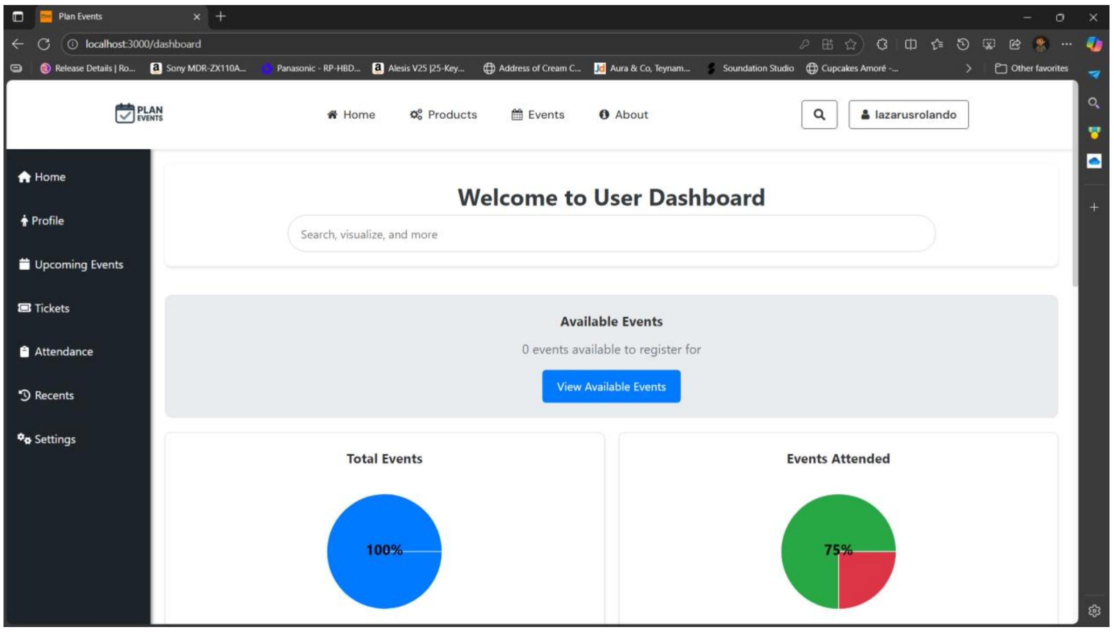
Email or Username: Email or Username

Password: Password (Show)

Remember me Forgot?

Speaker Login

New speaker? Create a speaker account



Welcome to User Dashboard

Search, visualize, and more

Available Events

0 events available to register for

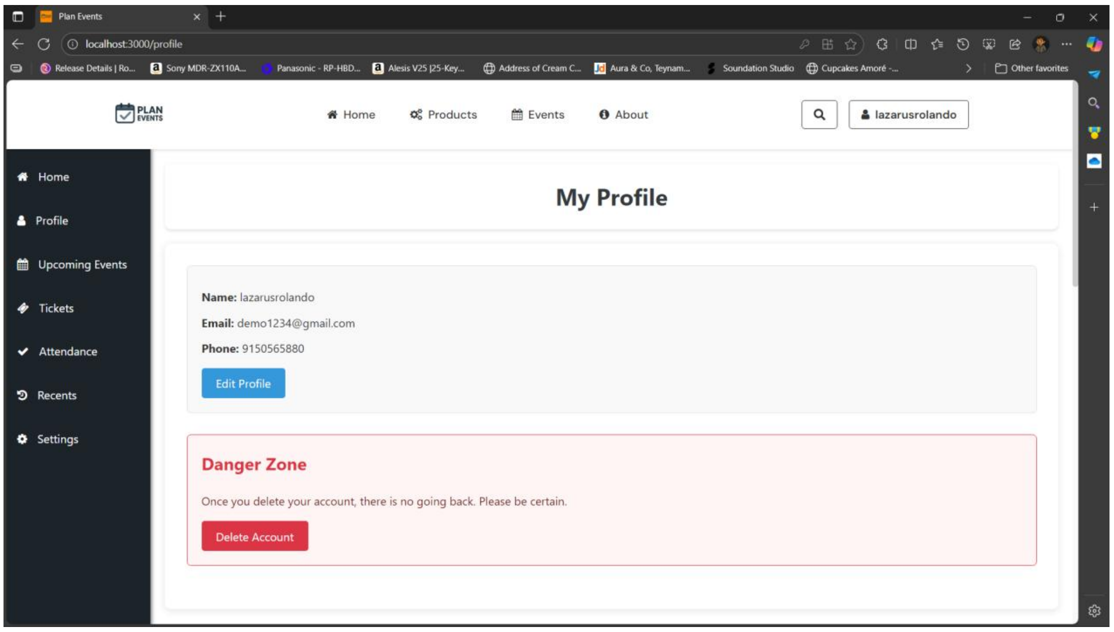
View Available Events

Total Events

100%

Events Attended

75%



My Profile

Name: lazarusrolando

Email: demo1234@gmail.com

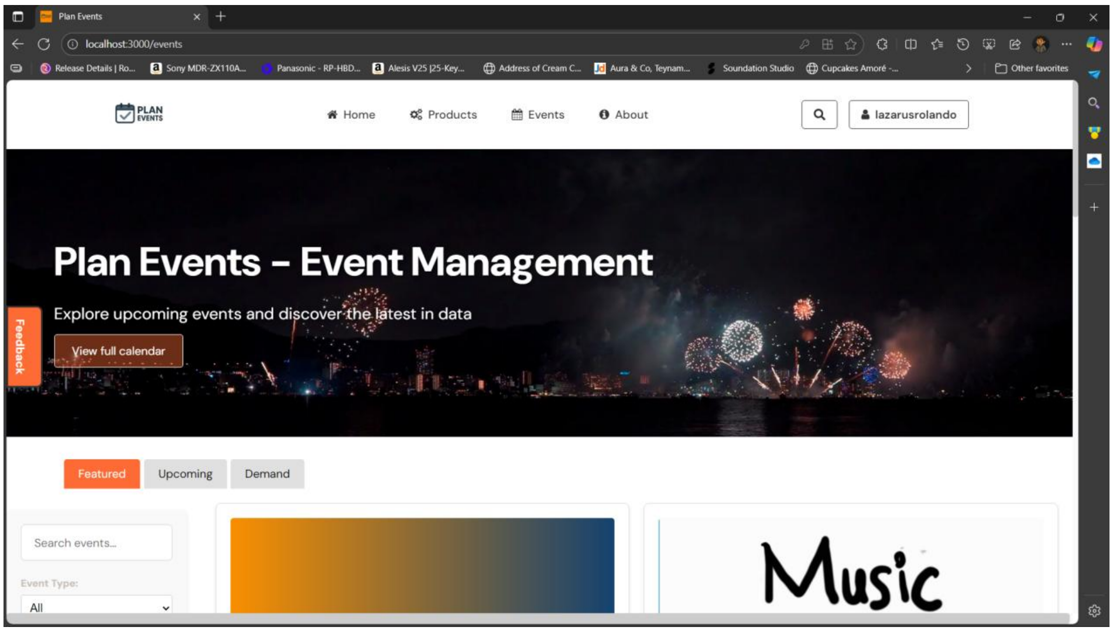
Phone: 9150565880

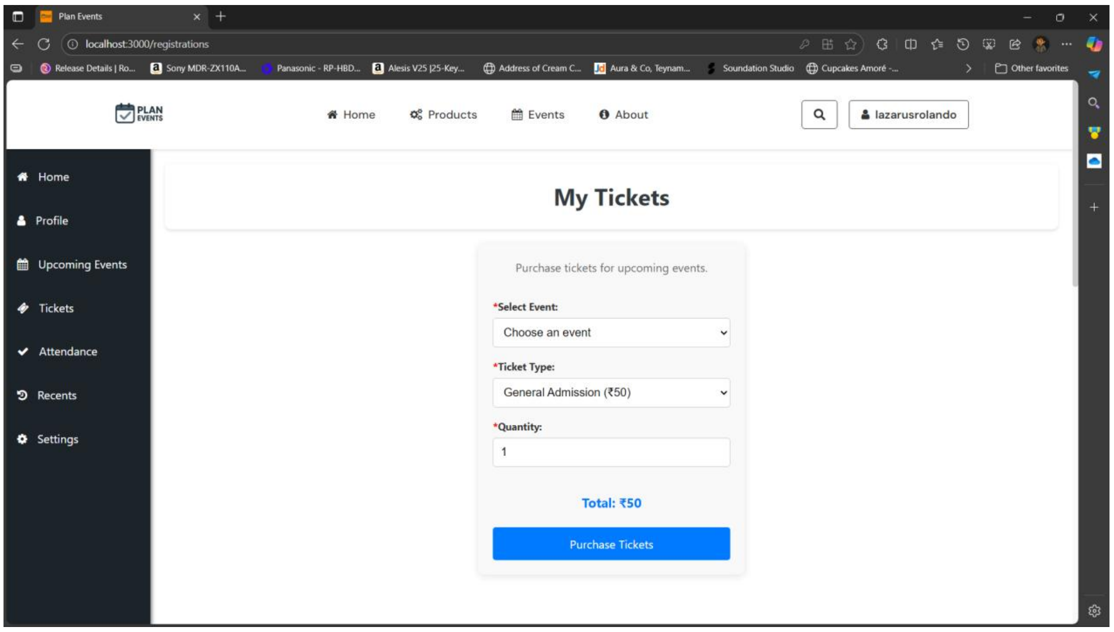
Edit Profile

Danger Zone

Once you delete your account, there is no going back. Please be certain.

Delete Account





My Tickets

Purchase tickets for upcoming events.

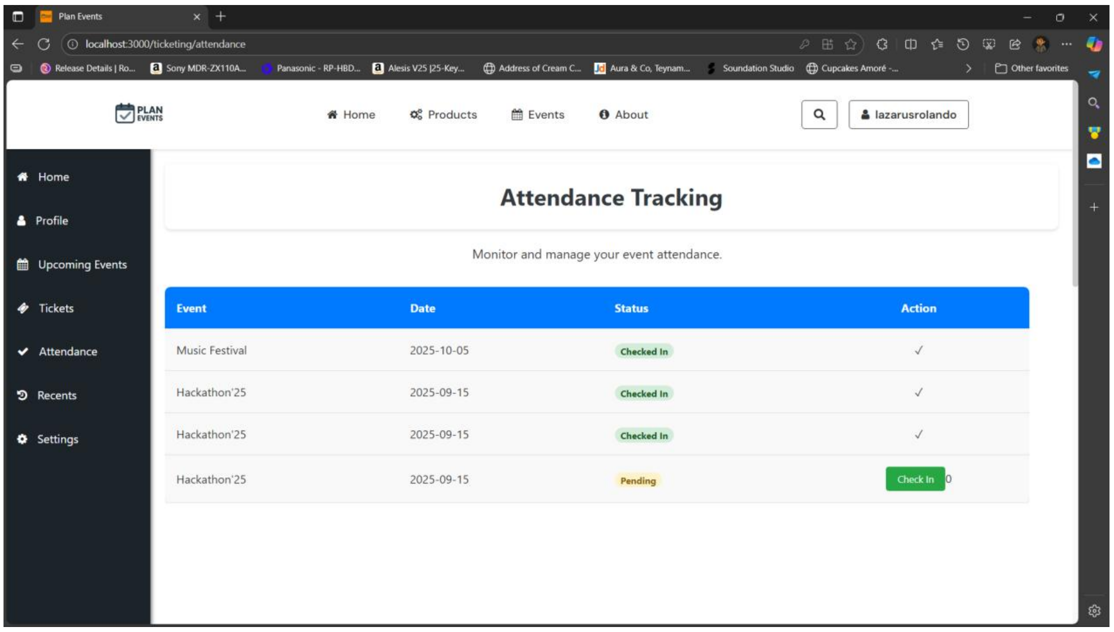
*Select Event: Choose an event

*Ticket Type: General Admission (₹50)

*Quantity: 1

Total: ₹50

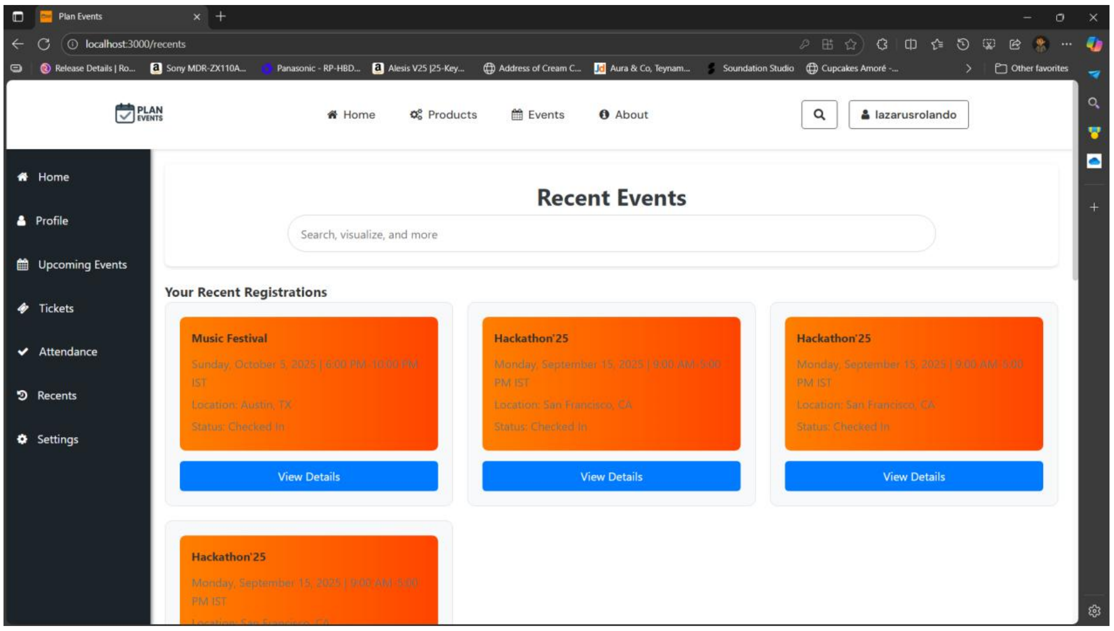
Purchase Tickets



Attendance Tracking

Monitor and manage your event attendance.

Event	Date	Status	Action
Music Festival	2025-10-05	Checked In	✓
Hackathon'25	2025-09-15	Checked In	✓
Hackathon'25	2025-09-15	Checked In	✓
Hackathon'25	2025-09-15	Pending	Check In 0



Recent Events

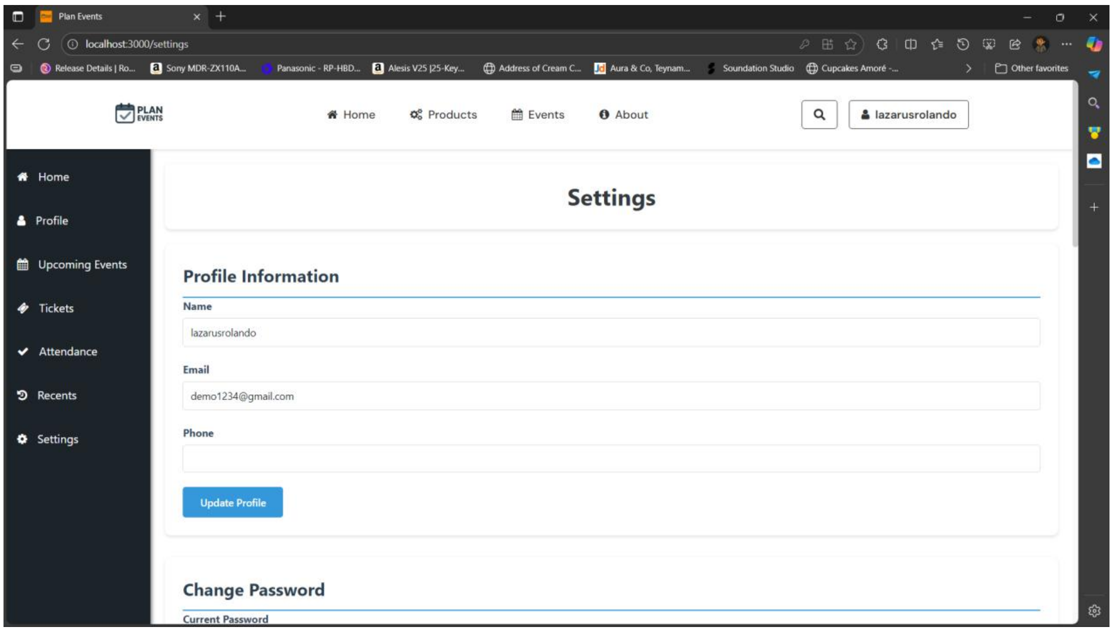
Your Recent Registrations

Music Festival
Sunday, October 5, 2025 | 6:00 PM -10:00 PM IST
Location: Austin, TX
Status: Checked In

Hackathon'25
Monday, September 15, 2025 | 9:00 AM-5:00 PM IST
Location: San Francisco, CA
Status: Checked In

Hackathon'25
Monday, September 15, 2025 | 9:00 AM-5:00 PM IST
Location: San Francisco, CA
Status: Checked In

Hackathon'25
Monday, September 15, 2025 | 9:00 AM-5:00 PM IST
Location: San Francisco, CA



Settings

Profile Information

Name: lazarusrolando

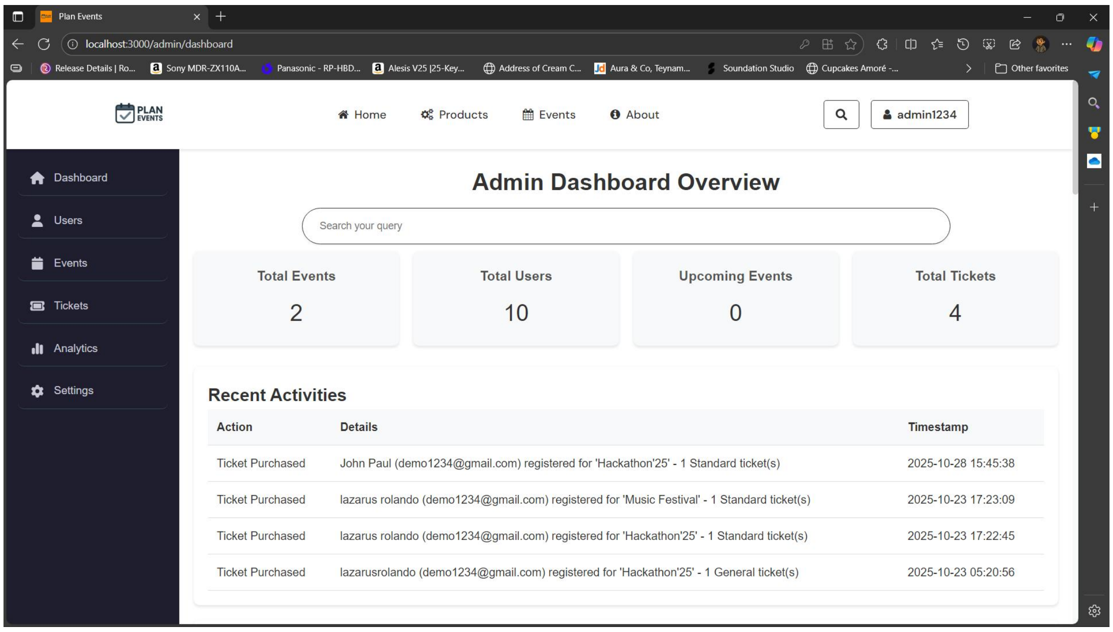
Email: demo1234@gmail.com

Phone:

Update Profile

Change Password

Current Password

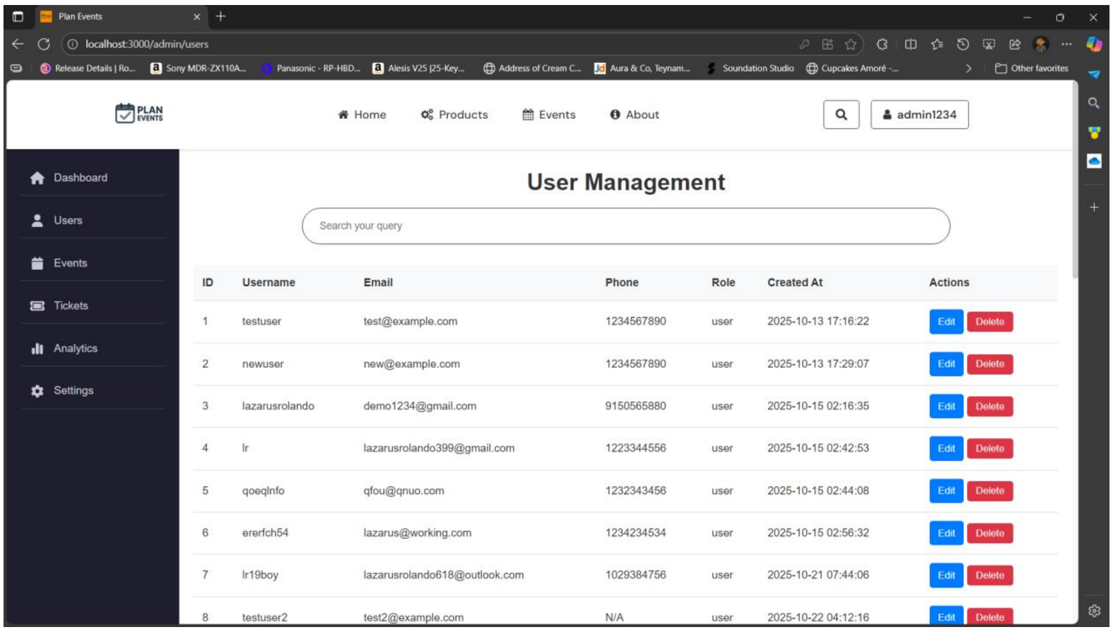


Admin Dashboard Overview

Total Events	Total Users	Upcoming Events	Total Tickets
2	10	0	4

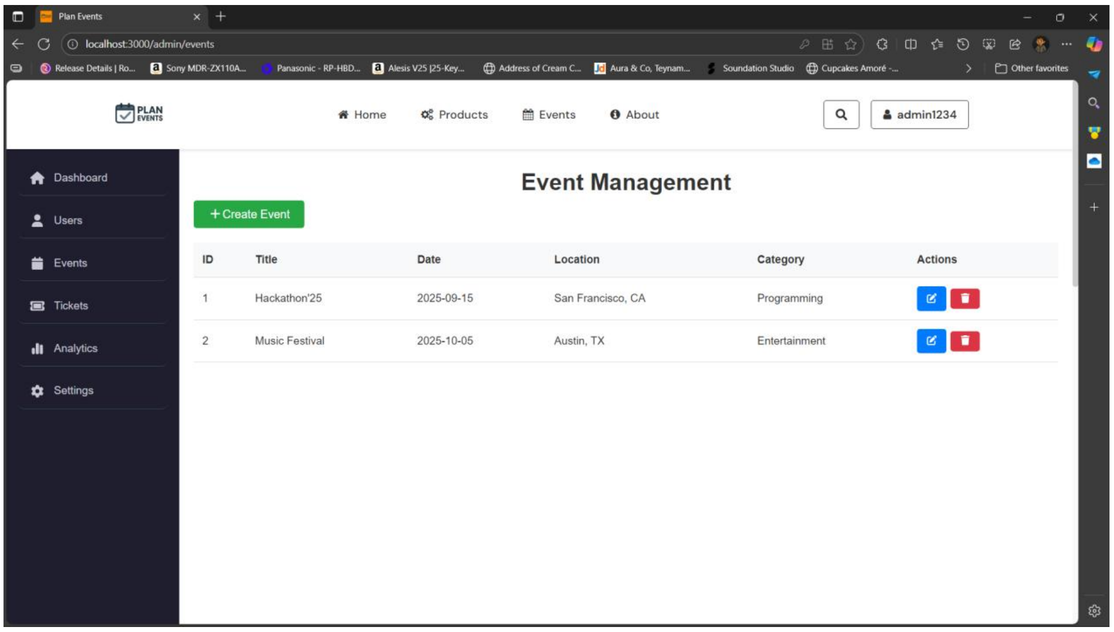
Recent Activities

Action	Details	Timestamp
Ticket Purchased	John Paul (demo1234@gmail.com) registered for 'Hackathon'25' - 1 Standard ticket(s)	2025-10-28 15:45:38
Ticket Purchased	lazarus rolando (demo1234@gmail.com) registered for 'Music Festival' - 1 Standard ticket(s)	2025-10-23 17:23:09
Ticket Purchased	lazarus rolando (demo1234@gmail.com) registered for 'Hackathon'25' - 1 Standard ticket(s)	2025-10-23 17:22:45
Ticket Purchased	lazarusrolando (demo1234@gmail.com) registered for 'Hackathon'25' - 1 General ticket(s)	2025-10-23 05:20:56



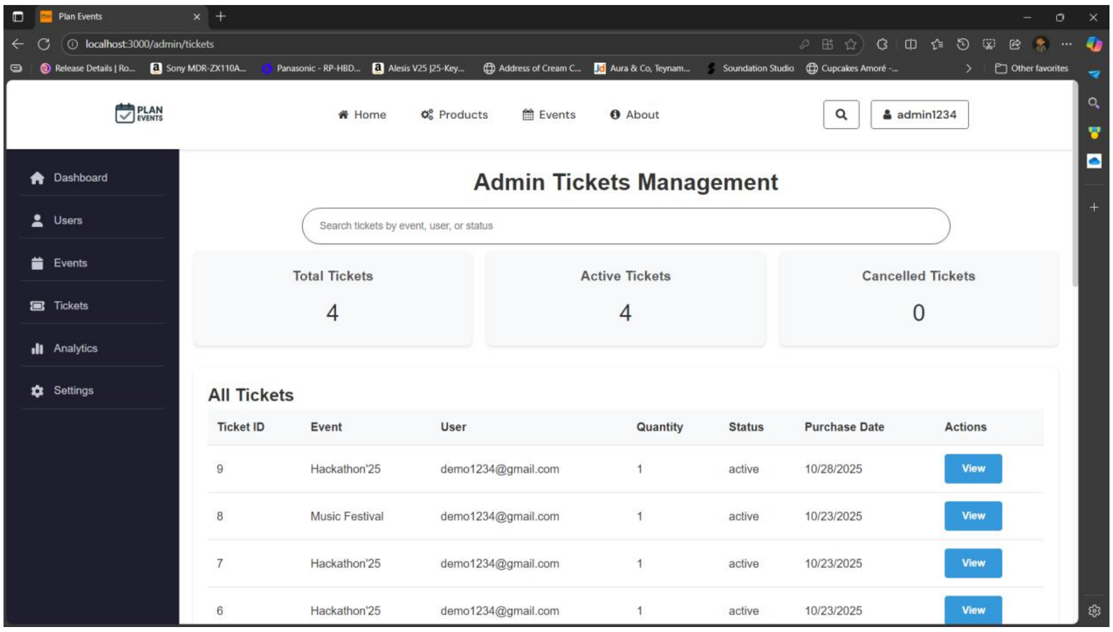
User Management

ID	Username	Email	Phone	Role	Created At	Actions
1	testuser	test@example.com	1234567890	user	2025-10-13 17:16:22	Edit Delete
2	newuser	new@example.com	1234567890	user	2025-10-13 17:29:07	Edit Delete
3	lazarusrolando	demo1234@gmail.com	9150565880	user	2025-10-15 02:16:35	Edit Delete
4	lr	lazarusrolando399@gmail.com	1223344556	user	2025-10-15 02:42:53	Edit Delete
5	qoeqlnfo	qfou@qnuo.com	1232343456	user	2025-10-15 02:44:08	Edit Delete
6	ererfch54	lazarus@working.com	1234234534	user	2025-10-15 02:56:32	Edit Delete
7	lr19boy	lazarusrolando618@outlook.com	1029384756	user	2025-10-21 07:44:06	Edit Delete
8	testuser2	test2@example.com	N/A	user	2025-10-22 04:12:16	Edit Delete



Event Management

ID	Title	Date	Location	Category	Actions
1	Hackathon'25	2025-09-15	San Francisco, CA	Programming	
2	Music Festival	2025-10-05	Austin, TX	Entertainment	



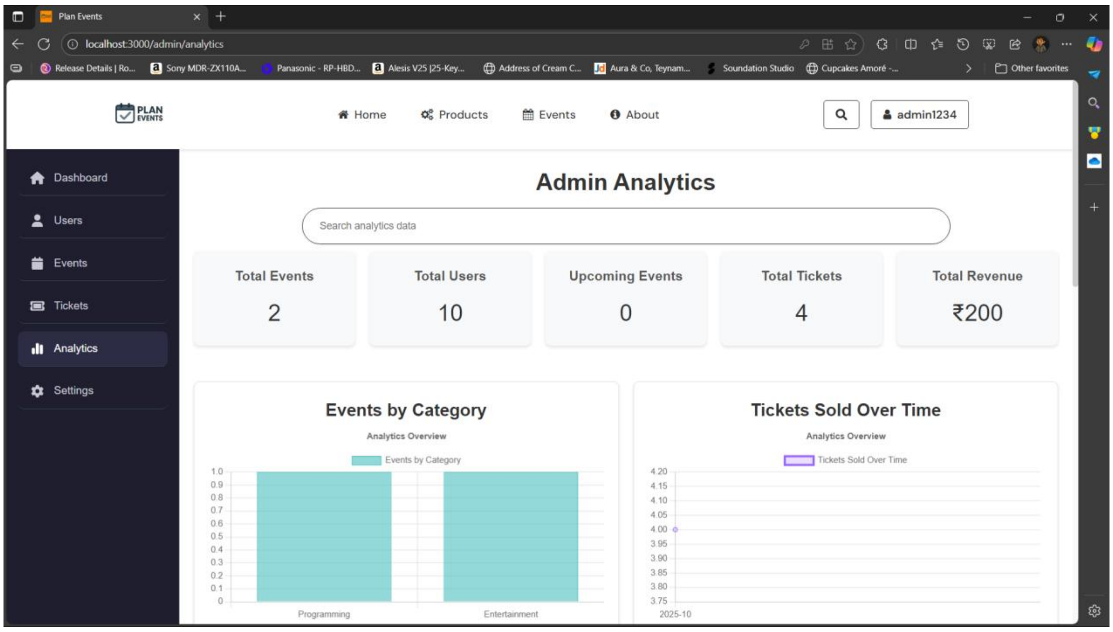
Admin Tickets Management

Search tickets by event, user, or status

Total Tickets	Active Tickets	Cancelled Tickets
4	4	0

All Tickets

Ticket ID	Event	User	Quantity	Status	Purchase Date	Actions
9	Hackathon'25	demo1234@gmail.com	1	active	10/28/2025	View
8	Music Festival	demo1234@gmail.com	1	active	10/23/2025	View
7	Hackathon'25	demo1234@gmail.com	1	active	10/23/2025	View
6	Hackathon'25	demo1234@gmail.com	1	active	10/23/2025	View



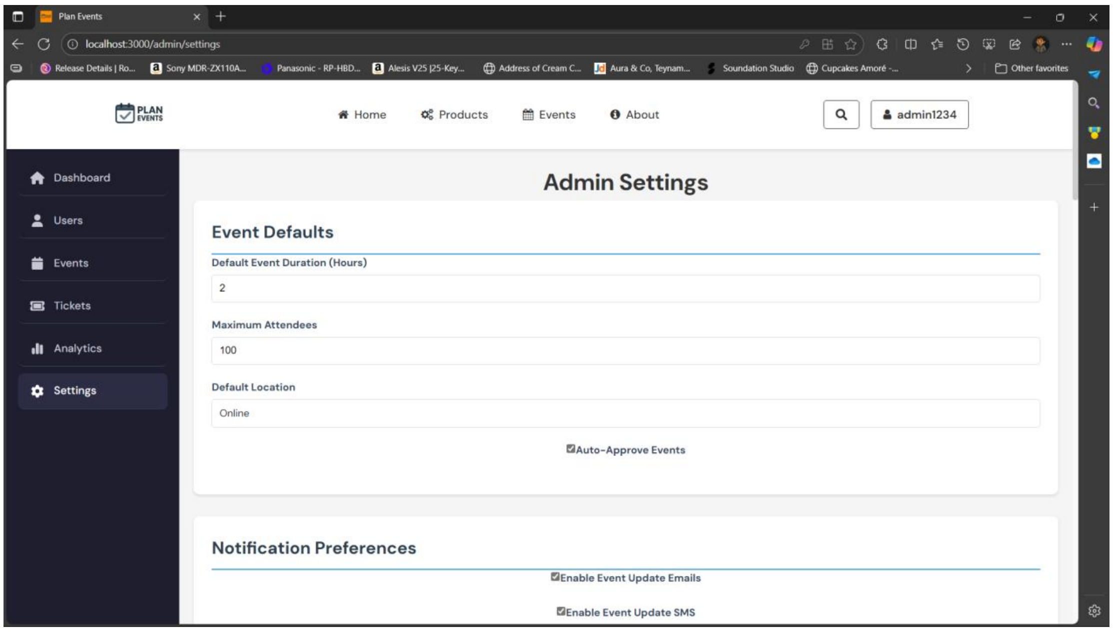
Admin Analytics

Search analytics data

Total Events	Total Users	Upcoming Events	Total Tickets	Total Revenue
2	10	0	4	₹200

Events by Category

Tickets Sold Over Time



Admin Settings

Event Defaults

Default Event Duration (Hours)

2

Maximum Attendees

100

Default Location

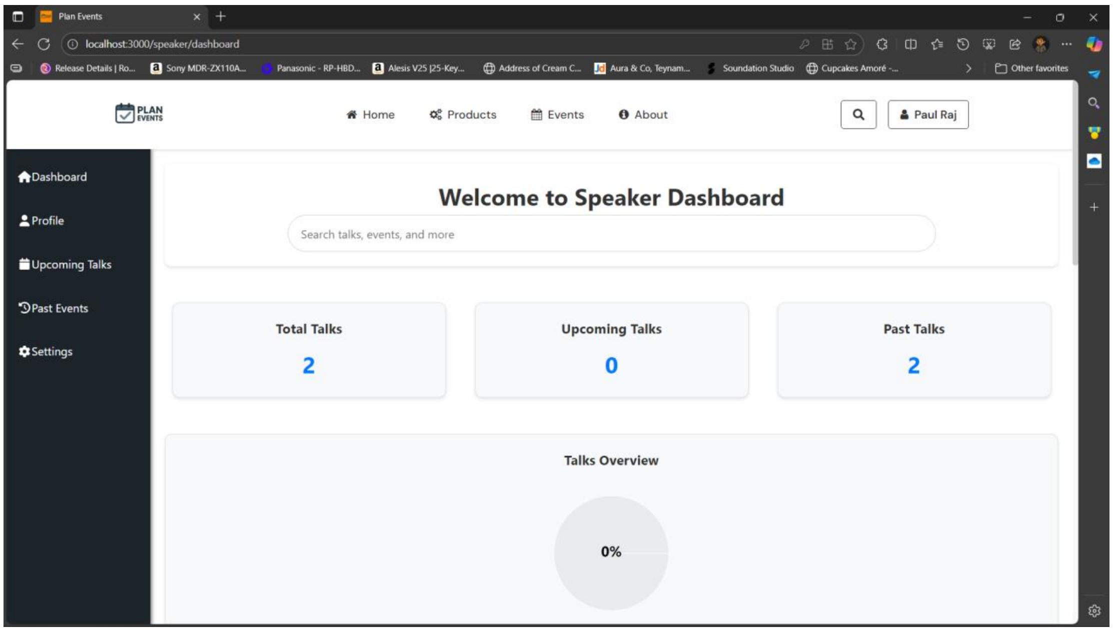
Online

☑Auto-Approve Events

Notification Preferences

☑Enable Event Update Emails

☑Enable Event Update SMS



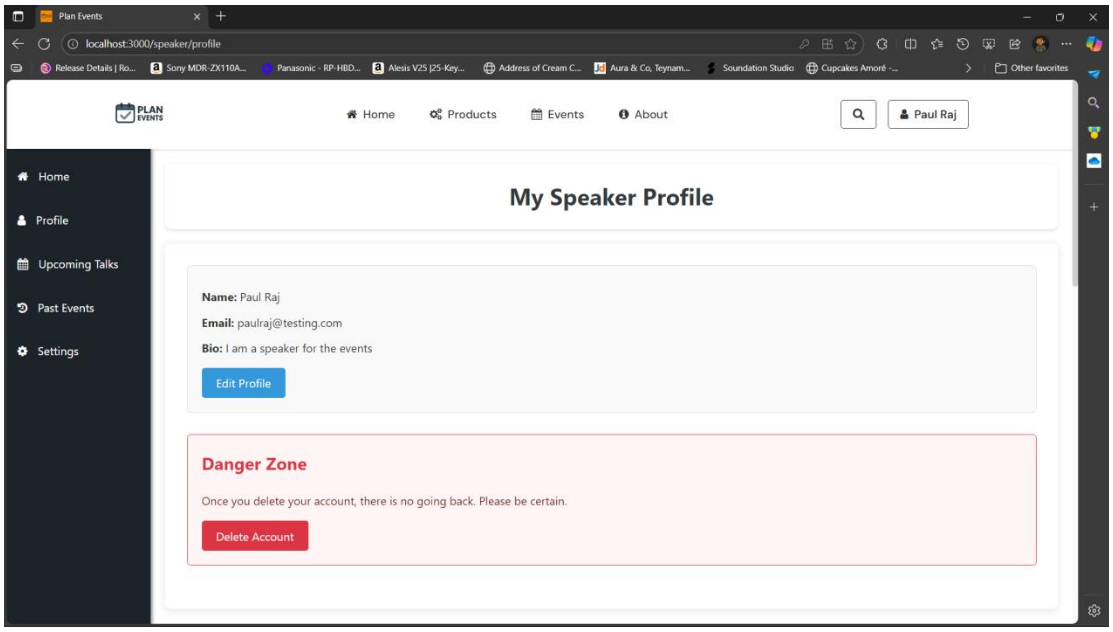
Welcome to Speaker Dashboard

Search talks, events, and more

Total Talks	Upcoming Talks	Past Talks
2	0	2

Talks Overview

0%



My Speaker Profile

Name: Paul Raj

Email: paulraj@testing.com

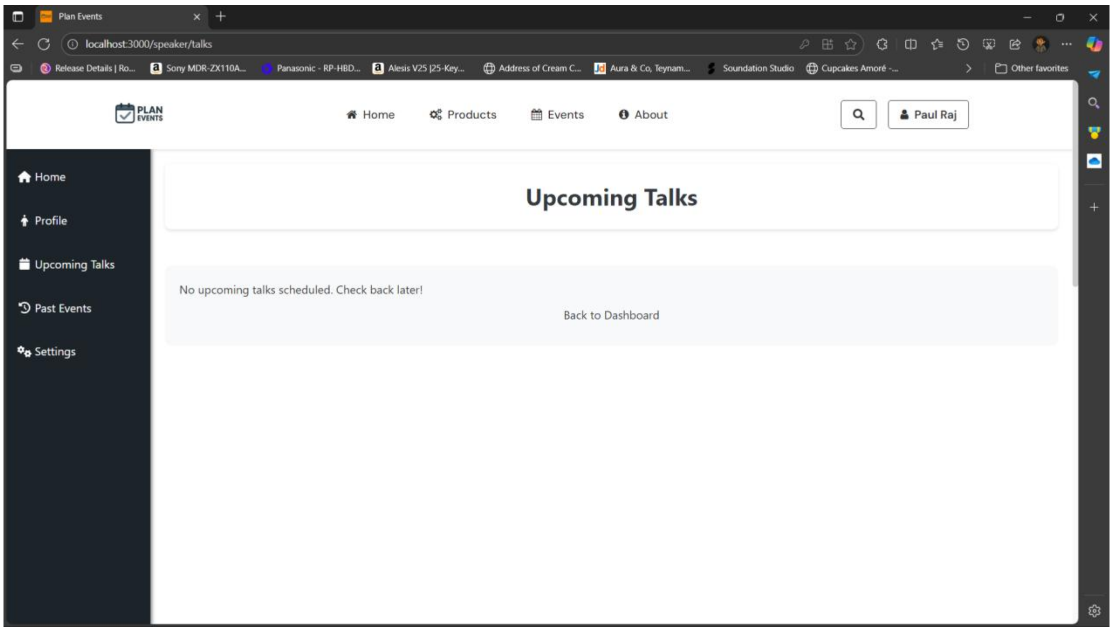
Bio: I am a speaker for the events

Edit Profile

Danger Zone

Once you delete your account, there is no going back. Please be certain.

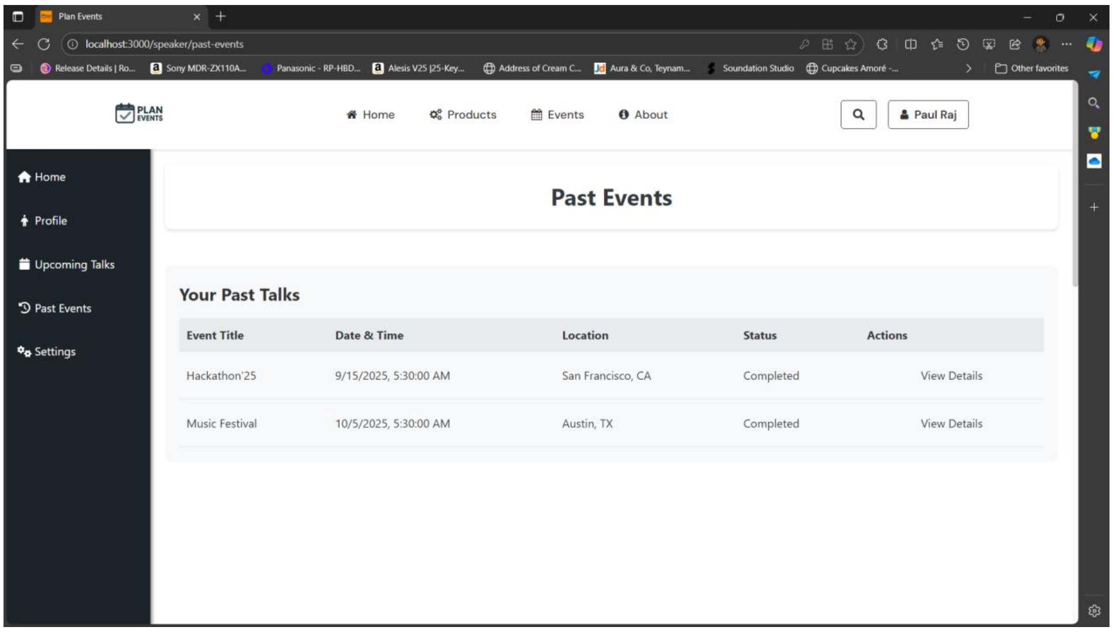
Delete Account



Upcoming Talks

No upcoming talks scheduled. Check back later!

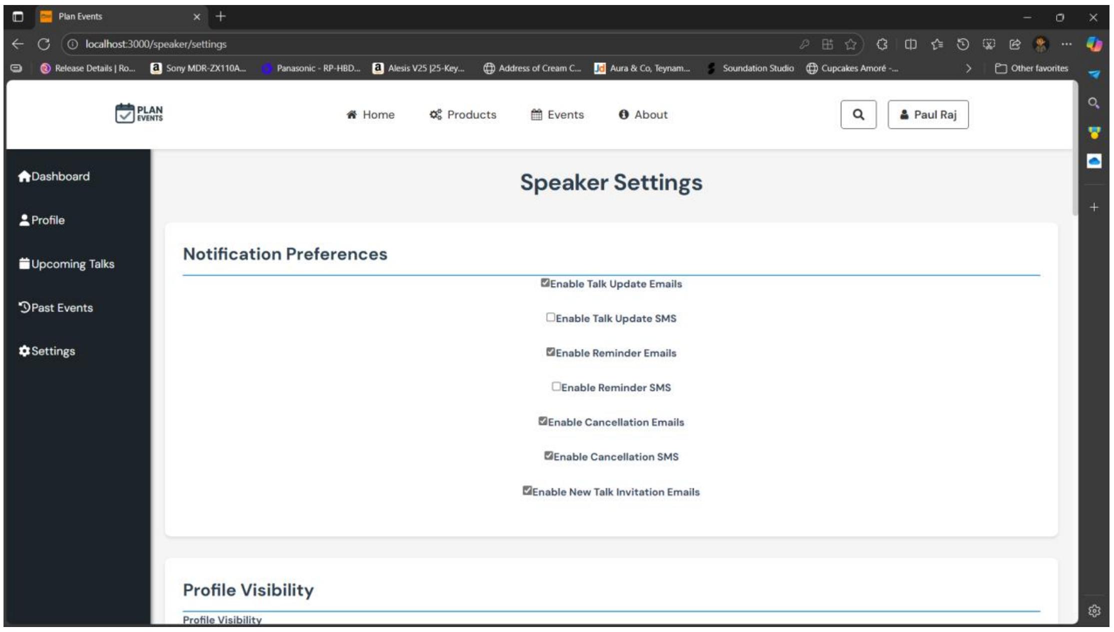
Back to Dashboard



Past Events

Your Past Talks

Event Title	Date & Time	Location	Status	Actions
Hackathon'25	9/15/2025, 5:30:00 AM	San Francisco, CA	Completed	View Details
Music Festival	10/5/2025, 5:30:00 AM	Austin, TX	Completed	View Details



Speaker Settings

Notification Preferences

- [x] Enable Talk Update Emails
- [ ] Enable Talk Update SMS
- [x] Enable Reminder Emails
- [ ] Enable Reminder SMS
- [x] Enable Cancellation Emails
- [x] Enable Cancellation SMS
- [x] Enable New Talk Invitation Emails

Profile Visibility

Profile Visibility

CHAPTER - 12

Conclusion and Future Work

Conclusion

The **Plan Events Application** was successfully developed and tested as a comprehensive platform for managing and organizing events efficiently. The system provides an integrated solution for event creation, participant registration, scheduling, and real-time communication between organizers and attendees. Through the use of modern web technologies and a structured database design, the application ensures reliability, scalability, and ease of use.

Throughout the development process, emphasis was placed on **user experience**, **data integrity**, and **system performance**. Extensive testing — including unit, integration, system, and user acceptance testing — ensured that all modules functioned as expected. The results demonstrated that the Plan Events Application meets the defined functional and non-functional requirements, providing a robust solution for both event organizers and participants.

In addition, the application's architecture was designed with modularity and scalability in mind, allowing for easy maintenance and future enhancements. The integration of secure authentication mechanisms and efficient data flow further strengthened the system's reliability and usability.

Future Work

While the current version of the Plan Events Application fulfills the primary objectives, there are several potential improvements and expansions that could be implemented to enhance its functionality and user engagement:

1. **Mobile Application Development:**
   Creating a dedicated mobile app for Android and iOS platforms to allow users to manage events and registrations on the go.

2. **AI-Based Recommendations:**
   Integrating artificial intelligence to suggest events to users based on their interests, past participation, and location.

3. **Enhanced Notification System:**
   Implementing push notifications and email reminders for upcoming events, registration deadlines, and schedule changes.

4. **Analytics and Reporting:**
   Providing event organizers with detailed analytics dashboards for tracking attendance, engagement, and feedback metrics.

5. **Integration with Third-Party Platforms:**
   Connecting with popular platforms like Google Calendar, PayPal, and social media for seamless event sharing and payment handling.

6. **Feedback and Rating System:**
   Allowing participants to rate events and provide feedback to help organizers improve future sessions.

7. **Multi-Language and Accessibility Support:**
   Expanding the system's usability to a global audience by adding language localization and accessibility features for users with disabilities.

CHAPTER - 13

REFERENCES

BOOK

1. **Node.js**
   Herron, D. (2020). *Node.js Web Development (5th ed.)*. Packt Publishing.

2. **React.js**
   Banks, A., & Porcello, E. (2017). *Learning React: Functional Web Development with React and Redux (2nd ed.)*. O'Reilly Media.
3. **Python**
   Lutz, M. (2013). *Learning Python (5th ed.)*. O'Reilly Media.

4. **Flask**
   Grinberg, M. (2018). *Flask Web Development: Developing Web Applications with Python (2nd ed.)*. O'Reilly Media.

5. **SQLite3**
   Owens, M., & Allen, G. (2010). *The Definitive Guide to SQLite (2nd ed.)*. Apress.

6. **HTML & CSS**
   Duckett, J. (2011). *HTML and CSS: Design and Build Websites*. John Wiley & Sons.

WEBSITES

HTML & CSS: https://www.w3.org/

Node.js: https://nodejs.org/

React.js: https://react.dev/

Python: https://www.python.org/

Flask: https://flask.palletsprojects.com/en/stable/

SQLite: https://sqlite.org/