Malnad College of Engineering

(An autonomous Institution under Visvesvaraya Technological University,Belgaum)

PB#21, Hassan -573202, Karnataka, India

Full Stack Development (23IS505)

TITLE: Event Management System Using Full Stack Development

Group number: 06

Submitted by

Name	USN
Banu Prasad JK	4MC23IS015
Chandan R	4MC23IS022
Jeevan KM	4MC23IS050
Hemanth Kumar HA	4MC22IS132

Under the guidance of

Mr. Krishna Swaroop A

Assistant Professor,

Department of ISE

Department of Information Science & Engineering

Malnad College of Engineering

Date of submission: 28-11-2025

Table of Content

2. ABSTRACT

Event management systems have evolved significantly with the rise of modern web technologies, enabling organizers to plan, coordinate, and execute events more efficiently. This project focuses on developing a comprehensive full-stack event management application that streamlines tasks such as event creation, registration, scheduling, communication, and real-time updates. Using a full-stack development approach, the system integrates a dynamic and responsive front-end interface, a robust back-end server, and a scalable database to ensure seamless data flow and user interaction. Core features include user authentication, event listings, attendee management, automated notifications, and administrative dashboards. The project highlights the importance of modern frameworks and APIs in improving usability, enhancing automation, and supporting multi-user collaboration. Overall, this full-stack event management system demonstrates how integrated web technologies can transform traditional event planning into a more organized, interactive, and efficient digital experience.

3. INTRODUCTION

Background of the Problem

Traditional event management relies on manual processes like paper registration, spreadsheets, and fragmented communication, which often cause errors, delays, and poor coordination. As events grow larger, these methods become inefficient and fail to meet the need for real-time information and smooth communication. Without a centralized digital system, managing attendees, schedules, and updates becomes difficult, highlighting the need for an automated, full-stack solution to streamline the entire event process.Traditional event management often relies on manual, paper-based processes that are time-consuming and prone to errors.

- Manual event processes lead to errors and delays.
- Spreadsheets and emails make coordination difficult.
- Hard to manage large events using traditional methods.
- Lack of real-time updates affects organizers and attendees.
- No centralized system for storing and accessing event data.
- Need for a digital, automated full-stack solution.

Why This Domain Was Chosen

This domain was chosen because event management often faces challenges like manual work, poor coordination, and lack of automation. A full-stack solution can simplify event planning by providing a centralized, easy-to-use digital system. It also offers a practical way to apply both front-end and back-end development skills to solve real-world problems and improve the overall event experience.

- Event management needs digital automation.
- Manual processes create delays and confusion.
- A full-stack system can streamline event tasks.
- The domain allows practical use of web development skills.
- Helps solve real-world organizational problems.

Real-World Scenario Description

In real-world situations, organizing events such as college fests, seminars, workshops, or corporate meetings often becomes challenging due to manual coordination and scattered information. For example, when a college hosts an annual cultural event, students need to register for multiple activities, organizers must track registrations, schedule events, assign volunteers, and update participants about changes. Without a centralized digital system, this leads to long queues, miscommunication, and delayed updates. A full-stack event management platform solves this problem by offering online registration, automated data storage, real-time announcements, and dashboards for organizers—making the entire event smoother, faster, and more organized for everyone involved.

Problems arise when:

- Registrations are handled manually.
- Organizers track data using paper or spreadsheets.
- Participants do not receive timely updates.
- Large events become difficult to manage manually.
- There is no centralized system for storing event information.
- Communication between organizers and attendees is unclear.

Issues in the Existing System:

- Registration is slow and manually managed.
- Data can be inaccurate or easily lost.
- Information is scattered across multiple sources.
- No real-time updates for attendees.
- Difficult for organizers to track participants and schedules.

- Communication is slow and unorganized.
- Overall event coordination becomes inefficient.

How the Proposed Solution Helps:

- Automates registration and data management.
- Provides a centralized platform for organizers and attendees.
- Ensures real-time updates and notifications.
- Reduces errors and improves accuracy.
- Makes scheduling and tracking easier for organizers.
- Enhances communication and coordination.
- Improves overall event efficiency and user experience.

4. OBJECTIVES OF THE PROJECT

The main objective of this project is to develop a full-stack event management system that simplifies the planning, coordination, and execution of events. The system aims to automate registration, scheduling, and communication processes while providing a centralized platform for organizers and participants. It seeks to improve data accuracy, enable real-time updates, enhance user experience, and reduce the workload on organizers. Additionally, the project focuses on creating a scalable, interactive, and user-friendly solution that can be applied across different types of events.

- To automate event registration and attendee management.
- To provide a centralized platform for organizers and participants.
- To ensure accurate storage and management of event data.
- To enable real-time updates and notifications for users.
- To improve overall coordination and communication during events.
- To reduce manual effort and minimize errors in event management.
- To create a scalable and user-friendly system suitable for various events.

5. SYSTEM REQUIREMENTS

5.1 Software Requirements

The Examination Management & Scheduling System requires the following software components for development, execution, and maintenance:

• Python Version

- Python3.10orabove
  Recommended for maximum compatibility with Django 4.x and to ensure stable performance during development.

• Django Version

- Django4.x(4.0/4.1/4.2)
  Used for implementing authentication, URL routing, ORM-based database interactions, and template rendering.

• Integrated Development Environment (IDE)

Any of the following development environments may be used:

- Visual Studio Code (VS Code) recommended for its extensions and lightweight performance
- PyCharm Community/Professional Edition
- Sublime Text, Atom, or any basic text editor (optional)

• Database Used

- SQLite3(defaultDjangodevelopmentdatabase)
  Lightweight, serverless, and automatically integrated with Django.

Optional production-ready alternatives (if required):

- MySQL

- PostgreSQL

• Other Dependencies / Technologies

- HTML5, CSS3, JavaScript for frontend structure, styling, and basic interactivity
- Bootstrap or Tailwind CSS for responsive user interface design
- Django Template Engine for rendering dynamic HTML pages
- Pillow if the project is extended to include file uploads or media handling

Python Virtual Environment (venv) for isolating project dependencies.

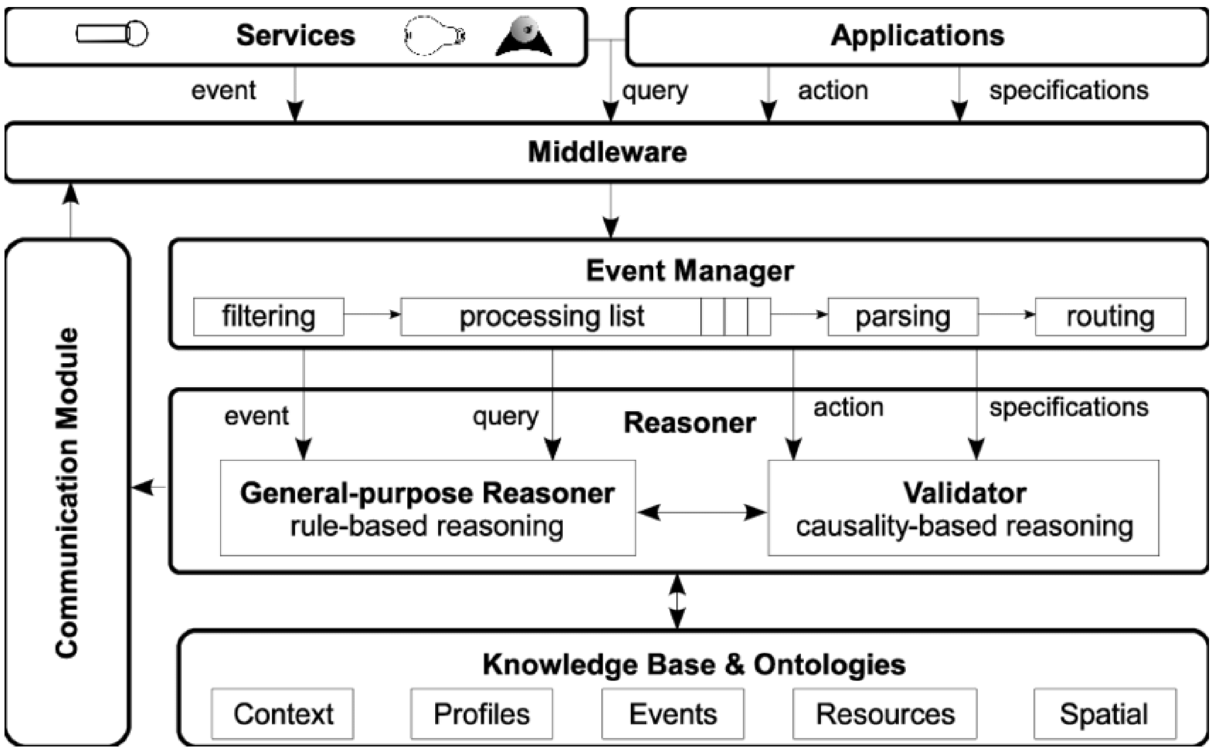
5.2 Hardware Requirements

Minimum System Configuration

- **Processor:** Dual Core 2.0 GHz or higher
- **RAM:** Minimum 4 GB (8 GB recommended)
- **Storage:** Minimum 10 GB free disk space
- **Device:** Laptop / Desktop capable of running Python environment.

6. SYSTEM DESIGN

6.1 Architecture Diagram



6.2 Explanation of Architecture

The architecture of the full-stack event management system follows a three-tier structure: Front-End (Client), Back-End (Server), and Database (Data Storage). Each layer plays a specific role and interacts seamlessly to provide a smooth event management experience.

1. Front-End (Client Side)

- The front-end is the interface that users interact with.
- It includes pages for event registration, login, dashboards, schedules, and notifications.
- Technologies such as HTML, CSS, JavaScript, and frameworks like React.js or Angular are used to create a responsive and user-friendly interface.

- The front-end communicates with the back-end through APIs to fetch or send data.

2. Back-End (Server Side)

- The back-end handles business logic, processes client requests, and manages authentication and authorization.
- Built using Node.js with Express.js (or equivalent), it validates requests, processes data, and interacts with the database.
- The back-end also manages security, such as login verification and role-based access control.

3. Database (Data Storage)

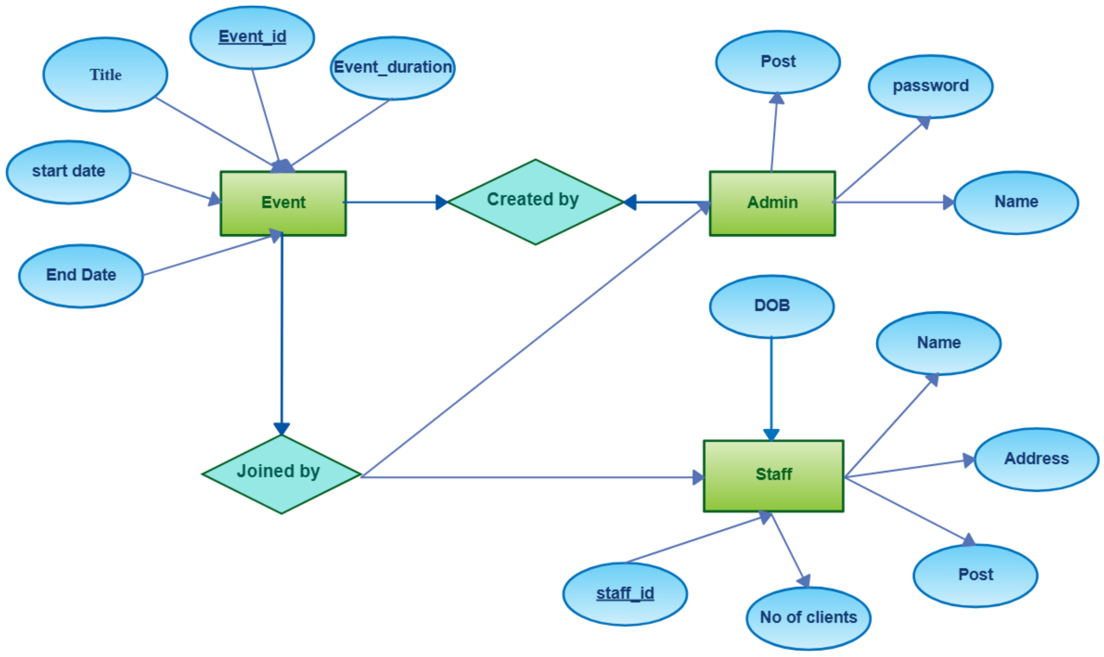
- The database stores critical data, including user profiles, event details, registrations, schedules, and notifications.
- Options include MySQL, PostgreSQL, or MongoDB.
- The back-end interacts with the database using queries or ORMs (Object Relational Mappers) to perform CRUD operations efficiently.

Data Flow

1. Users perform actions on the front-end (e.g., registering for an event).
2. The front-end sends a request to the back-end server via APIs.
3. The server processes the request, validates data, and interacts with the database.
4. The database returns the requested data to the server.
5. The server sends the response to the front-end, where users see updates in real-time.

7. DATABASE DESIGN

7.1 ER Diagram / Schema Diagram



7.2 Tables with Attributes

1. Users Table

Stores information about the system users (organizers and participants).

Attribute	Data Type	Description
user_id	INT (PK)	Unique ID for each user
name	VARCHAR(100)	Full name of the user
email	VARCHAR(100)	Email address of the user
password	VARCHAR(255)	Hashed password for login
role	ENUM('organizer','participant')	User type
created_at	DATETIME	Account creation timestamp

---

2. Events Table

Stores details about events created by organizers.

Attribute	Data Type	Description
event_id	INT (PK)	Unique ID for each event
event_name	VARCHAR(100)	Name of the event
description	TEXT	Details about the event
location	VARCHAR(150)	Venue of the event
start_date	DATETIME	Event start date and time
end_date	DATETIME	Event end date and time
organizer_id	INT (FK)	ID of the organizer (links to Users)
created_at	DATETIME	Event creation timestamp

---

3. Registrations Table

Keeps track of participants registering for events.

Attribute	Data Type	Description
registration_id	INT (PK)	Unique ID for each registration
event_id	INT (FK)	ID of the registered event
user_id	INT (FK)	ID of the participant
registration_date	DATETIME	Date and time of registration
status	ENUM('confirmed','pending','cancelled')	Registration status

4. Notifications Table

Stores messages or updates sent to users about events.

Attribute	Data Type	Description
notification_id	INT (PK)	Unique ID for each notification
event_id	INT (FK)	Related event ID
user_id	INT (FK)	Recipient user ID
message	TEXT	Notification content
created_at	DATETIME	Timestamp of when notification was sent
status	ENUM('read','unread')	Notification status

8. IMPLEMENTATION

This chapter describes the technical implementation of the system, including models, URL routing, functionalities, and the special logic applied in the backend. The project follows Django’s standard MVC/MVT architecture, where models handle data, views process logic, and templates provide the user interface.

8.1 Models Overview

The project uses a single custom model named Exam, which is responsible for storing all examination-related information for each student.

Exam Model Fields

- user:
  A foreign key linked to Django’s built-in User model.
  It represents the student to whom the exam entry belongs, establishing a one-to-many relationship.
- subject:
  Stores the name of the subject for which the exam is scheduled (e.g., “Mathematics”, “Computer Networks”).
- date:
  The scheduled date of the examination.
- time:
  The starting time of the examination.
- room:
  The allocated examination room number for the student.
- seat:
  The seat number assigned to the student within the examination hall.

Model Usage in the System

- Stores multiple exam entries per student
- Acts as the central data storage component
- Interacts with Django ORM for CRUD operations
- Helps generate personalized exam timetables for each user

The simplicity of the model makes it easy to manage and scale while maintaining integrity through foreign key relationships.

8.2 URL Routing Overview

The system uses Django's URL dispatcher to map browser requests to corresponding views. The main URL routes included in the application are:

Defined Endpoints

- /register/ — Displays the account creation form for new students
- /login/ — Presents the login page and validates user credentials
- /dashboard/ — Displays the personalized exam schedule for the logged-in student
- /logout/ — Logs out the user and ends the session

URL Routing Logic

Each URL pattern is associated with a specific view function that:

- Receives user input
- Processes the request
- Retrieves data from the database
- Returns an output using an HTML template

The routing ensures smooth navigation across all modules of the application.

8.3 Important Functionalities

This system implements several essential features to manage examination scheduling efficiently:

1 **User Registration and Authentication**

- Allows organizers and participants to create accounts.
- Secure login using hashed passwords.
- Role-based access control for organizers and participants.

2 **Event Creation and Management**

- Organizers can create, edit, and delete events.
- Add event details such as name, description, location, and schedule.
- Manage multiple events simultaneously.

3 **Event Registration**

- Participants can register online for events.
- Registration confirmation and status tracking.
- Ability to cancel or modify registration before deadlines.

4 **Dashboard and Analytics**

- Organizers can view participant lists and event statistics.
- Monitor registrations, attendance, and other metrics in real-time.

5 **Notifications and Alerts**

- Automated notifications for event updates, reminders, and cancellations.
- Real-time messages to participants and organizers.

6 **Search and Filtering**

- Participants can search for events by name, date, or category.
- Filter events based on location, availability, or type.

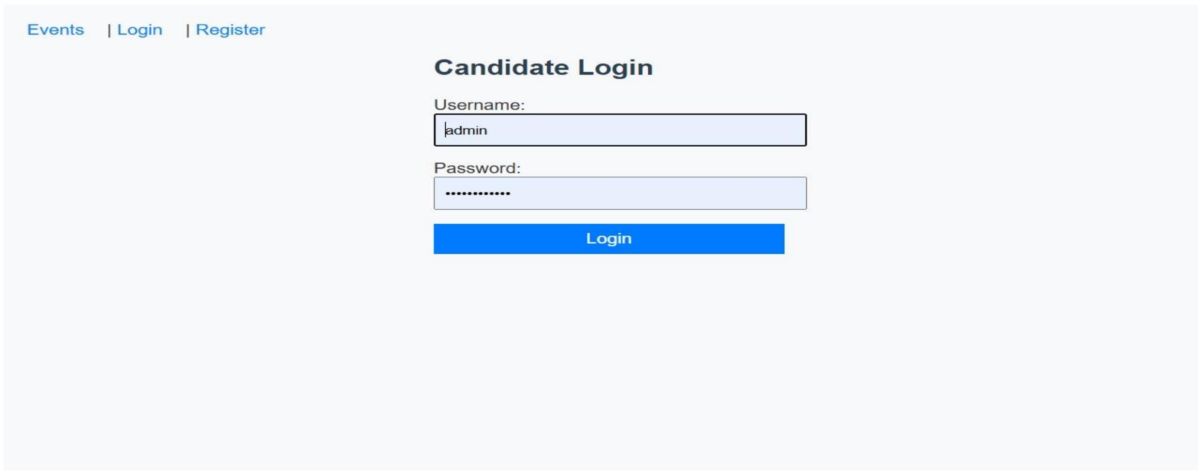
7 **Data Management**

- Centralized storage of users, events, and registrations.
- Easy retrieval and update of records.

8 **Feedback and Reviews (Optional)**

- Participants can provide feedback or ratings for events.
- Organizers can use feedback to improve future events.

9 SCREENSHOTS

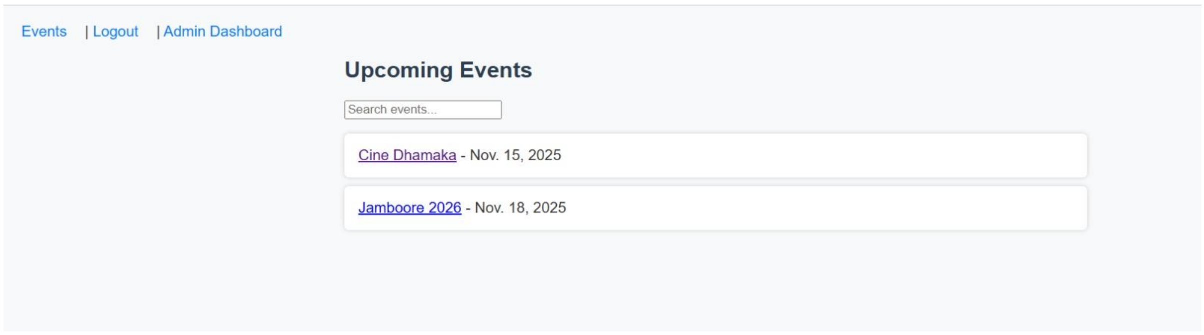

Events | Login | Register

Candidate Login

Username: admin

Password: ············

Login

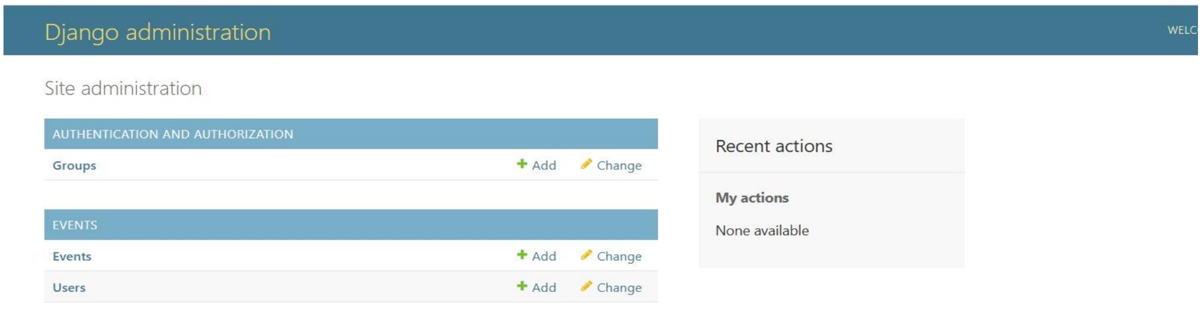

Events | Logout | Admin Dashboard

Upcoming Events

Search events...

Cine Dhamaka - Nov. 15, 2025

Jamboore 2026 - Nov. 18, 2025


Django administration

WELC

Site administration

AUTHENTICATION AND AUTHORIZATION

Groups + Add Change

EVENTS

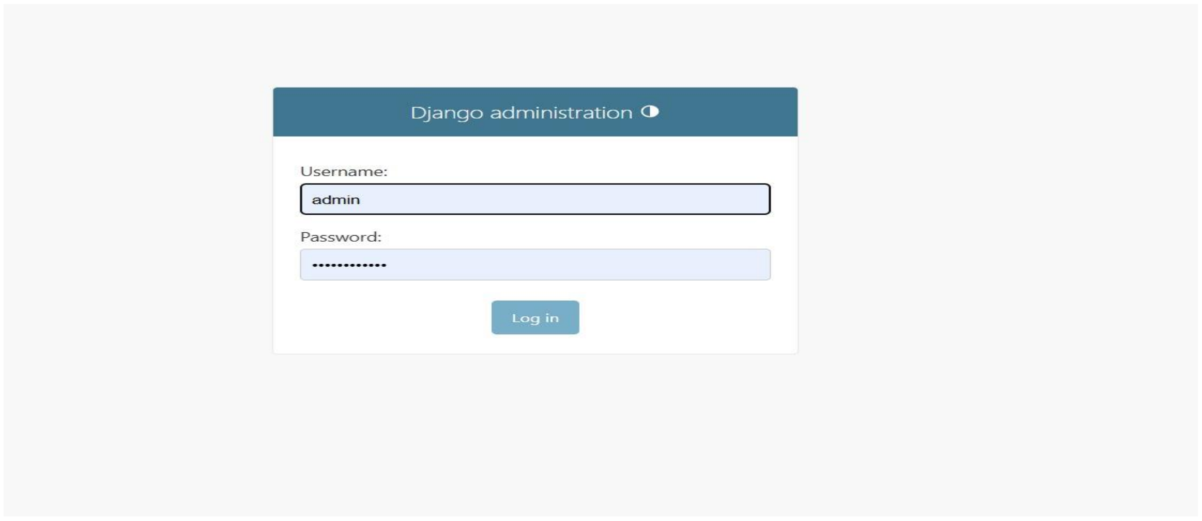
Events + Add Change

Users + Add Change

Recent actions

My actions

None available


Django administration

Username: admin

Password: ············

Log in

10. TESTING

Testing is an essential part of the software development process to ensure that the event management system works correctly, efficiently, and securely. Different types of testing are performed at various stages of the system to identify and fix issues before deployment.

1. Unit Testing

- Tests individual components or functions of the system, such as user registration, event creation, or login functionality.
- Ensures that each module performs as expected independently.
- Example: Verifying that a new user can register successfully and their data is stored in the database.

2. Integration Testing

- Checks how different modules work together, such as the interaction between front-end forms and back-end APIs.
- Ensures that the system components communicate correctly and data flows seamlessly.
- Example: Ensuring that a participant's registration is correctly reflected in the event dashboard.

3. System Testing

- Validates the complete system's functionality against requirements.
- Ensures all features like event creation, registration, notifications, and dashboards work as intended.
- Example: Creating an event, registering multiple participants, and sending notifications successfully.

4. Usability Testing

- Ensures the system is user-friendly and intuitive for organizers and participants.
- Checks interface responsiveness, ease of navigation, and clarity of instructions.

5. Performance Testing

- Evaluates system speed, responsiveness, and stability under load.

- Ensures the system can handle multiple users and large events without crashes or delays.

6. Security Testing

- Ensures the safety of user data, including passwords and personal information.
- Tests for vulnerabilities like SQL injection, unauthorized access, and data breaches.

7. Acceptance Testing

- Conducted by end-users to verify that the system meets real-world requirements.
- Ensures the application is ready for deployment and practical use.

11. RESULTS

The full-stack event management system successfully provides an automated, efficient, and user-friendly solution for organizing events. Organizers can easily create, edit, and manage events, while participants can register online without manual paperwork, ensuring accurate and real-time tracking of registrations. Notifications and alerts are sent automatically, keeping all users informed about updates, cancellations, or reminders. All data, including user profiles, event details, and registrations, is securely stored in a centralized database, allowing easy retrieval and reporting. The front-end interface is intuitive and responsive, enabling users to access schedules, registrations, and notifications seamlessly. Additionally, the system handles multiple users and large events efficiently, with proper security measures such as authentication and role-based access protecting sensitive information. Overall, the system improves efficiency, reduces manual effort, ensures data accuracy, and provides a smooth experience for both organizers and participants, demonstrating the effectiveness of the proposed solution.

12. CONCLUSION

The full-stack event management system successfully addresses the limitations of traditional event management methods by providing a centralized, automated, and user-friendly platform. It streamlines event creation, registration, scheduling, and communication, reducing manual effort and minimizing errors. Real-time updates and notifications ensure that participants and organizers remain informed, while secure data storage maintains accuracy and privacy. The system's responsive interface, scalability, and efficient performance make it suitable for events of varying sizes and types. Overall, the project demonstrates how full-stack development can be effectively applied to solve real-world organizational challenges, enhancing efficiency, coordination, and user experience in event management.

13. FUTURE ENHANCEMENTS

The full-stack event management system can be further enhanced to increase functionality, accessibility, and user engagement. Future improvements may include integrating payment gateways for online ticketing, adding advanced analytics and reporting to monitor participant engagement, and developing mobile applications for iOS and Android for easier access. Additional features like feedback and rating systems, event reminders via email or SMS, and social media integration can improve interaction and communication. Implementing AI-based event recommendations and deploying the system on the cloud can enhance scalability, efficiency, and future readiness. These enhancements will make the system more robust, versatile, and suitable for a wider range of events.

- Integrate payment gateways for online ticketing and registration fees.
- Add advanced analytics and reporting for participant engagement and event tracking.
- Develop mobile applications for iOS and Android for better accessibility.
- Implement feedback and rating systems for events.
- Enable event reminders via email and SMS notifications.
- Integrate with social media platforms for broader communication.
- Provide AI-based recommendations for personalized event suggestions.

14. REFERENCES

- Django Official Documentation: https://docs.djangoproject.com
- Python Official Documentation: https://www.python.org/doc/
- Bootstrap Documentation: https://getbootstrap.com
- MDN Web Docs (HTML, CSS, JS): https://developer.mozilla.org
- Stack Overflow (Community Support)