Engineering and Scientific International Journal (ESIJ)
Volume 13, Issue 2, April – June 2026
ISSN 2394-7187(Online)
ISSN 2394 - 7179 (Print)

Design and Development of a Scalable Event Management System using Django Web Framework

Tholla Sunil[*1], M. Gowthami[2]

[1] *Student, Department of Computer Applications, Viswam Engineering College, Madanapalle, Andhra Pradesh.*
[2] *Assistant Professor, Department of Computer Applications, Viswam Engineering College, Madanapalle, Andhra Pradesh.*

Abstract — The increasing complexity of organizing events in academic, corporate, and community environments necessitates efficient and automated management systems. Traditional event coordination methods, which rely heavily on manual processes such as physical registrations, spreadsheets, and email communication, are prone to inefficiencies, errors, and scalability limitations. This paper presents a comprehensive Event Management System developed as a full-stack web application using the Django framework with SQLite as the backend database. The system provides a unified digital platform that supports two primary user roles: event organizers and participants. Organizers can create, manage, and schedule events, monitor participant registrations, track attendance, and send automated email notifications. Participants can browse available events, register for events, view schedules, and monitor their participation through a personalized dashboard. The system architecture follows the Model-View-Template (MVT) pattern, ensuring modularity, scalability, and maintainability. The application interface is designed using Bootstrap 5.3, providing a responsive and user-friendly experience across devices. The proposed solution effectively automates event coordination processes, reduces administrative overhead, enhances participant engagement, and provides a scalable platform for managing events in modern digital environments.

Keywords: *Event Management System; Django Framework; Web Application; Bootstrap; SQLite; Role-Based Access Control.*

1. Introduction

The rapid growth of digital technologies has changed how events are planned and managed, but many institutions still rely on manual methods like spreadsheets and emails, which are inefficient, error-prone, and not scalable. These traditional approaches often lead to poor coordination, data duplication, and lack of real-time updates, making it difficult for organizers to manage participants, track attendance, and communicate effectively. To solve these issues, a web-based Event Management System is proposed using the Django framework, providing a centralized and automated platform for handling all event-related activities. The system uses Django's MVT architecture with role-based access control to ensure smooth interaction between organizers and participants. It supports key features such as event creation, registration, scheduling, attendance tracking, and automated notifications, offering an efficient and structured solution for complete event lifecycle management.

2. Literature Survey

The development of event management systems has evolved from manual methods to web-based and cloud-based solutions due to the increasing need for efficient digital event coordination. Early manual systems using paper forms, spreadsheets, and emails were error-prone, time-consuming, and lacked centralized data management. Web-based systems improved efficiency by enabling online registration and centralized storage, but many lacked advanced features like role-based access, real-time dashboards, and automated notifications.

Commercial platforms such as Eventbrite, Meetup, and Cvent offer comprehensive features including ticketing, analytics, and communication tools, but they are often costly, less customizable, and dependent on third-party services, limiting their use in academic and small-scale environments. Research has shown that frameworks like Django support scalable and modular development using MVT architecture, while technologies such as Bootstrap and AJAX enhance user experience through responsive and interactive interfaces. The proposed Event Management System addresses existing limitations by offering a lightweight, fully integrated, and customizable solution using open-source technologies, making it suitable for efficient and cost-effective event management.

3. Proposed Methodology

The proposed Event Management System is a full-stack web application designed to centralize and automate the entire event lifecycle. It follows a modular approach where key components such as user authentication, event creation, participant registration, schedule management, attendance tracking, and notifications operate independently but remain interconnected for smooth workflow execution.
Users are divided into two roles, Event Organizers and Participants, where organizers manage and publish events, and participants browse and register for them through a web interface. The process begins with authentication, after which organizers create events with details like title, description, date, time, and location. Participants can register for events, and the system maintains proper tracking of all registrations. During the event lifecycle, attendance is

DOI: **10.30726/esij/v13.i2.2026.1320036**

ISSN 2394-7187(Online)
ISSN 2394 - 7179 (Print)

recorded either by organizers or participants, and automated email notifications are sent as reminders.

3.1 System Architecture

The Event Management System is built using the Model-View-Template (MVT) architectural pattern provided by the Django framework, as illustrated in Fig. 1. The Presentation Layer is developed using HTML, CSS, Bootstrap 5.3, and JavaScript for dynamic interactions. The Application Layer handles business logic including URL routing, view functions, authentication, authorization, and request processing. The Data Layer uses SQLite to store event details, participant information, schedule sessions, and attendance records, with Django ORM facilitating database interaction. A Static and External Services Layer include Bootstrap, Font Awesome for UI enhancement, email notification services, and static files.

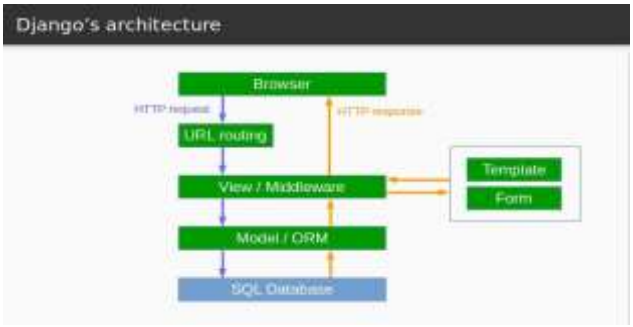

Fig. 1: System Architecture of Event Management System

3.2 System Flow Diagram

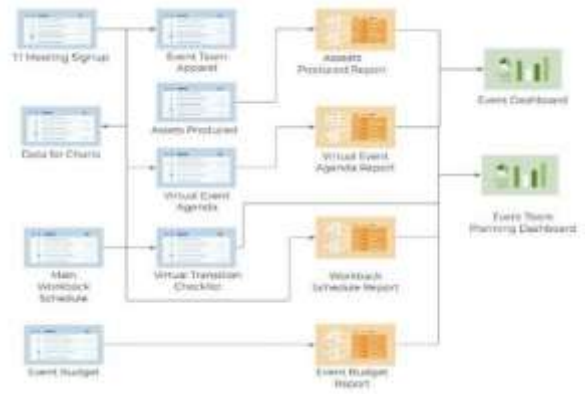

Fig. 2: System Flow Diagram of Event Management System

The system flow diagram in Fig. 2 illustrates the complete workflow from user authentication through event creation, participant registration, schedule management, attendance tracking, and notification delivery. The flow ensures that all event management activities are interconnected within a single platform.

3.3 UML Diagrams

3.3.1 Use Case Diagram

The Use Case Diagram in Fig. 3 shows two main actors: Event Organizer and Participant interacting with the system. Organizers manage events, schedules, attendance, and notifications after authentication, while participants register, browse events, enroll, and view their dashboards. The system also handles automated tasks like user validation, profile creation, and sending notifications.

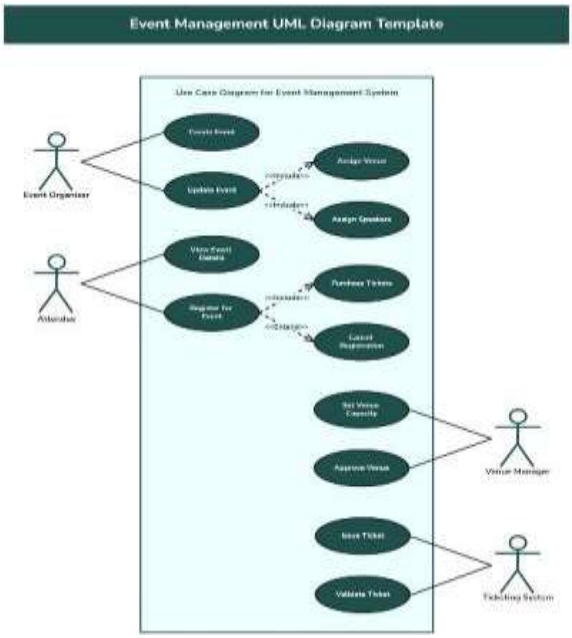

Fig. 3: Use Case Diagram of Event Management System

3.3.2 Class Diagram

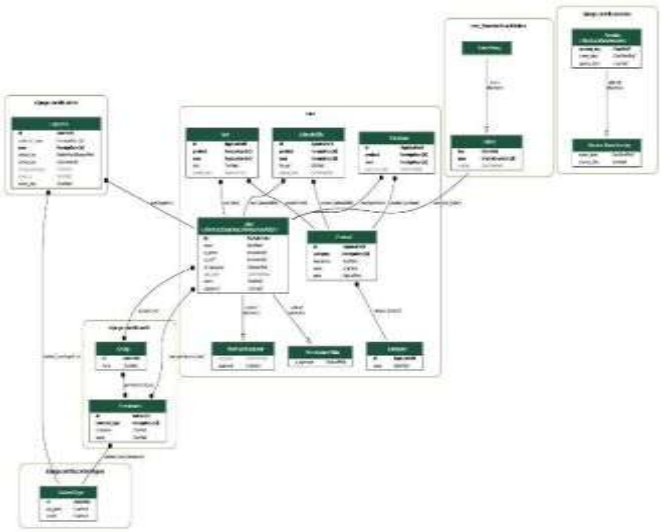

Fig. 4: Class Diagram of Event Management System

DOI: 10.30726/esij/v13.i2.2026.1320036

ISSN 2394-7187(Online)
ISSN 2394 - 7179 (Print)

The Class Diagram in Fig. 4 includes main entities such as Event, Participant, Schedule, and Attendance. The Event class stores event details and links with participants and schedules, while Participant manages registrations. Schedule defines session timings, and Attendance tracks participation. These classes are connected through one-to-many and many-to-many relationships for efficient data handling.

3.3.3 Sequence Diagram

The Sequence Diagram in Fig. 5 illustrates the interaction between system components over time during an event registration process, showing the flow between the web user, browser, and web server during authentication and account access.

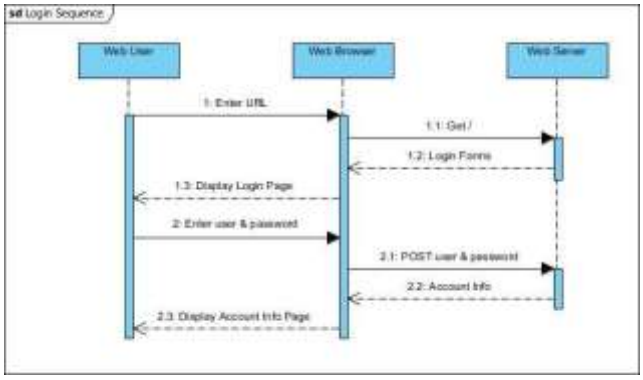

Fig. 5: Sequence Diagram of Event Management System

3.3.4 Activity Diagram

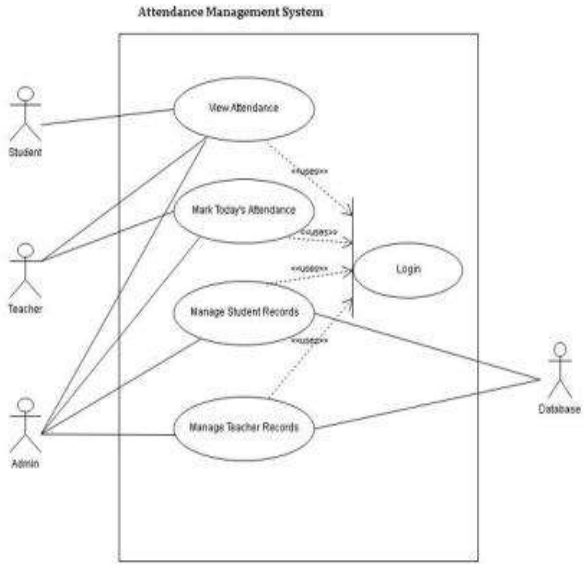

Fig. 6: Activity Diagram of Event Management System

The Activity Diagram in Fig. 6 represents the workflow of the system, showing sequential flow and decision-making processes. It covers schedule management, attendance tracking, notification delivery, and dashboard updates after event completion.

4. Implementation

The Event Management System is implemented as a full-stack web application using the Django framework for backend development and Bootstrap-based frontend technologies. The implementation follows the MVT architecture, ensuring modular design, maintainability, and scalability.

4.1 System Modules

The User Authentication Module manages user identity and access control for both participants and organizers. Participants register using Django's built-in user creation forms, while organizers register through a customized form with administrative privileges. Authentication is handled through secure session management with role-based access control. The Event Lifecycle Management Module enables organizers to create, manage, and publish events with details such as title, description, date, time, and location. Events are displayed on the homepage for participant access. The Participant Registration Module handles user enrolment into events through a many-to-many relationship, avoiding duplicate registrations and providing real-time feedback.

The Schedule Management Module manages session-level scheduling within events, with each event supporting multiple sessions linked through foreign key relationships. The Attendance Tracking Module records participant attendance through both self-reporting and organizer-controlled marking, with timestamp-based records preventing duplicate entries.

The Dashboard and Reporting Module provides personalized views for both participants (registered events, attendance status, event history) and organizers (events created, participant counts, attendance summaries). The Email Notification Module enables organizers to send event reminders with structured messages containing event details.

5. Results and Discussion

5.1 Functional Evaluation

The system successfully implements all core event management features including authentication, event creation, registration, scheduling, attendance tracking, and notifications. User authentication ensures secure login with role-based access. Event management allows organizers to create and display events dynamically. Registration prevents duplicates and maintains proper user-event mapping. Attendance tracking functions reliably through both self and organizer marking.

DOI: 10.30726/esij/v13.i2.2026.1320036

ISSN 2394-7187(Online)
ISSN 2394 - 7179 (Print)

5.2 System Performance Analysis

The system was evaluated for performance under moderate usage conditions, as shown in Fig. 7. The application handles multiple concurrent users with minimal delay, maintaining average response time below 500 milliseconds for standard operations. Database operations are optimized through Django ORM, and performance remains stable for small to medium-scale deployments.

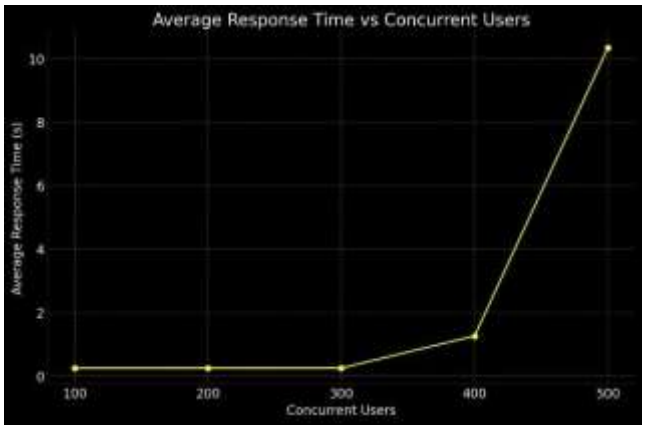

Fig. 7: Performance Analysis of Event Management System

5.3 User Experience Evaluation

The system interface developed using Bootstrap provides a clean and responsive design across different devices, as illustrated in Fig. 8. Users completed tasks such as registration, event creation, and attendance marking without difficulty, indicating good usability and accessibility.

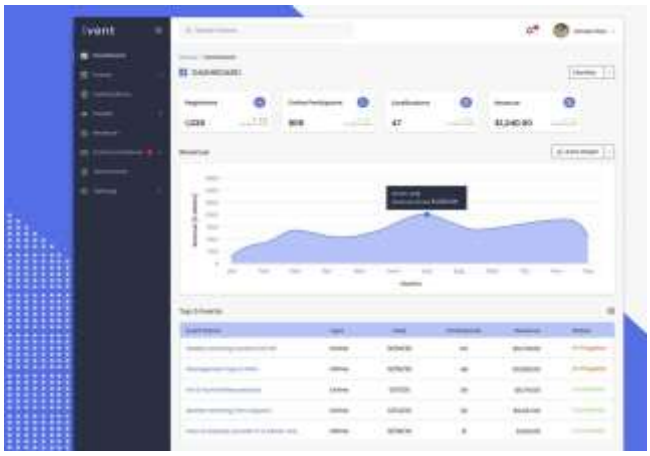

Fig. 8: User Interface and Dashboard Views

5.4 Discussion

The results show that the system improves efficiency by integrating all event management functions into a single platform with centralized data handling and automation. Unlike costly commercial tools, it is cost-effective and customizable using open-source technologies. However, limitations such as SQLite scalability for very large deployments and email notification configuration highlight areas for future improvement.

6. Conclusion

The Event Management System provides an efficient and centralized solution for managing events in academic and organizational settings by automating tasks such as event creation, registration, scheduling, attendance tracking, and notifications using the Django framework. It overcomes limitations of traditional manual methods by reducing errors, improving data consistency, and enhancing real-time accessibility. Built on the MVT architecture with responsive design, the system ensures modularity, scalability, and ease of use across devices.

7. Future Enhancements

Several enhancements can extend the system's capabilities. Migrating from SQLite to PostgreSQL or MySQL would improve scalability and performance for large-scale deployments. Adding payment gateway integration using services like Razorpay or Stripe would support paid event registration. Implementing advanced search and filtering by category, date, or location would improve event discovery. Introducing real-time notifications such as SMS and push alerts would enhance communication between organizers and participants.

References

[1] D. Merkel, "Django: A High-Level Python Web Framework," Linux Journal, vol. 2008, no. 168, pp. 1–10, 2008.
[2] Django Software Foundation, "Django Documentation," 2024.
[3] M. Otto and J. Thornton, "Bootstrap Framework Documentation," 2024.
[4] R. Elmasri and S. B. Navathe, Fundamentals of Database Systems, 7th ed. Pearson, 2016.
[5] SQLite Consortium, "SQLite Database Engine Documentation," 2024.
[6] I. Sommerville, Software Engineering, 10th ed. Pearson, 2015.
[7] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, Design Patterns. Addison-Wesley, 1994.
[8] W3C, "HTML5 Specification," 2024.
[9] Mozilla Developer Network, "JavaScript Documentation," 2024.
[10] S. Freeman and N. Pryce, Growing Object-Oriented Software. Addison-Wesley, 2009.
[11] P. Barry, Head First Python, 2nd ed. O'Reilly Media, 2016.
[12] T. Berners-Lee, "Information Management: A Proposal," CERN, 1989.
[13] A. Silberschatz, H. Korth, and S. Sudarshan, Database System Concepts, 6th ed. McGraw-Hill, 2011.
[14] G. Booch, J. Rumbaugh, and I. Jacobson, The Unified Modeling Language User Guide. Addison-Wesley, 2005.
[15] J. Nielsen, Usability Engineering Principles. Academic Press, 1993.

DOI: 10.30726/esij/v13.i2.2026.1320036