iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** III **Month of publication:** March 2026

DOI: https://doi.org/10.22214/ijraset.2026.79112

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue III Mar 2026- Available at www.ijraset.com

EventHive: A Web-Based Platform for Event Management

Prashant Khetade[1], Adnan Quazi[2], Atharva Pawar[3], Om Chandankhede[4], Prashant Kumar[5], Anand Kashyap[6], Shrirang Bobde[7]
Department of Computer Science and Engineering, G H Raisoni University, Amravati

Abstract: ***The growing complexity of organizing events and managing communities in academic and professional settings has exposed significant limitations in currently available digital tools. Most existing solutions address either event ticketing or community management in isolation, leaving a critical void for platforms that unify both functions seamlessly. This paper introduces EventHive, an integrated web-based platform that consolidates event creation, club administration, member communication, and participation tracking within a single cohesive environment. The platform empowers users to establish and manage clubs, publish and edit events, define member roles with associated permissions, and interact within dedicated community spaces. Among its distinctive capabilities, EventHive incorporates a QR code-driven entry verification system, whereby each registered participant receives a uniquely generated QR code linked to their registration record, enabling organizers to conduct swift and paperless attendance confirmation. Furthermore, the platform features an AI-powered image generation module that produces contextually appropriate event banners from brief textual descriptions supplied during event creation. The system is architected using React.js for the client-side interface and Supabase as the backend-as-a-service layer, utilizing PostgreSQL for relational data management, integrated authentication, and cloud-based file storage. Currently operational as a functional prototype, EventHive demonstrates considerable promise for transforming how institutions and organizations coordinate events and nurture active communities.***

Keywords: EventHive, Event Management, QR Code Verification, AI Image Generation, Supabase.

I. INTRODUCTION

Efficient coordination of events and communities has long been a central challenge for universities, professional associations, and social organizations. While platforms like Eventbrite and Meetup have provided partial solutions for public event listing and ticketing, they fall short of addressing the nuanced requirements of closed, structured communities. Specifically, they lack built-in support for club hierarchy management, role-based administration, in-platform messaging, and intelligent content assistance forcing organizers to depend on a fragmented combination of tools. This operational fragmentation creates tangible problems. In academic settings, student organizations frequently struggle to coordinate inter-club events, handle participant registrations, and manually confirm attendance. Professional groups face parallel difficulties in sustaining active community engagement while simultaneously orchestrating events at scale. The absence of a single platform that addresses all these dimensions represents a meaningful gap in the current technological landscape. This paper presents EventHive, a purpose-built web application that unifies event management and community infrastructure within one platform. The system enables users to establish or join clubs, create and oversee events, receive QR code-based entry passes, and utilize artificial intelligence to automatically produce visually compelling event imagery. Built on a modern stack comprising React.js and Supabase, EventHive delivers a responsive interface backed by a robust relational database, authentication services, and scalable cloud storage.

The principal contributions of this work encompass the following areas:

1) A unified platform architecture that integrates event management, community coordination, and AI-driven content creation within a single interface.
2) A hierarchical role-based access control system supporting club owners, administrators, and members with differentiated permissions and integrated communication channels.
3) A QR code-based registration and entry verification mechanism designed for paperless, real-time attendance management.
4) An AI-powered image generation feature that automatically produces event banners from organizer-supplied text prompts.
5) Advanced event discoverability tools including categorical search, tag-based filtering, and notification services.
6) A scalable, maintainable system architecture grounded in React.js and Supabase's backend-as-a-service ecosystem.

II. LITERATURE REVIEW

Patel et al. [1] examined the limitations of contemporary event management platforms in meeting the needs of closed, community-driven groups such as university clubs and corporate teams. Their study revealed that widely used tools are predominantly designed for large-scale public events and consequently lack the internal structure required for sustained community engagement. Critical capabilities such as member role management, internal communication, and community-specific event boards were found to be either absent or inadequately integrated.

The authors argued that following the global shift toward hybrid and digital events after 2020, demand for platforms capable of addressing both event logistics and community coordination grew substantially, yet no unified solution emerged to adequately serve both requirements.

Sharma and Verma [2] explored how digital community platforms influence participation and collaboration within organizational contexts. Their research, conducted across multiple university and professional groups, demonstrated that when members can interact, share updates, and manage events within a single unified platform, overall engagement increases measurably relative to environments where these functions are distributed across separate tools. The authors further observed that platforms incorporating clearly defined role structures — wherein certain members hold content management privileges while others are restricted to viewing and responding — tend to operate more smoothly and with fewer administrative conflicts. These findings directly shaped the community architecture of EventHive, which assigns distinct permission scopes to club owners, administrators, and general members.

Nguyen et al. [3] conducted a practical investigation into contactless and QR code-based verification systems within post-pandemic event management contexts. Their study demonstrated that QR code solutions substantially reduce check-in times, improve attendance tracking accuracy, and eliminate the vulnerabilities associated with paper-based ticketing. Notably, the authors established a meaningful distinction between static and dynamic QR codes, showing that dynamic codes — those linked to live database entries rather than fixed data — offer considerably greater security and operational flexibility. EventHive adopts this approach by generating a unique, token-linked QR code for every registration, enabling organizers to verify and record attendance in real time with a single scan.

Rombach et al. [4] introduced Latent Diffusion Models, the foundational architecture underpinning modern AI image generation tools such as Stable Diffusion. Their research demonstrated that high-quality, contextually relevant imagery could be synthesized from simple text prompts with strong consistency and speed, even on standard hardware. This development opened the door for practical AI image generation to be embedded into everyday applications beyond research settings. For event organizers who often lack time or design resources, this capability holds particular value. EventHive incorporates a comparable AI image generation feature during event creation, allowing any authorized user to produce a professional event banner by providing a brief descriptive prompt.

Ferraiolo et al. [5] revisited and extended the classical Role-Based Access Control (RBAC) framework within modern cloud-based and collaborative web environments. Their updated model addressed challenges that older RBAC implementations were not designed to handle, including dynamic team compositions, variable trust levels, and the need for fine-grained permission enforcement at the resource level. They demonstrated that when role hierarchies are clearly defined and enforced at both the application and database levels, systems become more secure, auditable, and maintainable. EventHive applies this principle by enforcing role-based permissions through both frontend routing logic and Supabase Row Level Security policies, ensuring consistent access control across every system layer.

Bons et al. [6] conducted a comprehensive comparative evaluation of modern backend-as-a-service platforms, assessing Firebase, Supabase, and AWS Amplify across scalability, developer experience, and relational data support. Their findings indicated that Supabase is particularly well-suited to applications that depend on structured relational data models — precisely the kind in which events belong to clubs, registrations connect users to events, and roles link members to communities. Unlike Firebase's document-oriented model, Supabase's PostgreSQL foundation natively enforces foreign key constraints, relational integrity, and complex querying. The authors also noted that Supabase's built-in authentication and storage services make it a strong all-in-one backend choice for lean development teams, a consideration that directly guided its adoption in EventHive.

Despite the advancements documented across these studies, a thorough review of existing literature confirms a persistent gap: no single platform combines club-based community management, AI-assisted event creation, QR code verification, and real-time communication in a unified interface. EventHive is designed to bridge this gap comprehensively.

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue III Mar 2026- Available at www.ijraset.com

III. METHODOLOGY

EventHive is structured as a modular, component-based web application, designed with scalability and maintainability as core architectural principles. The development methodology spans system design, technology selection, feature implementation, and functional verification across all core modules.

A. System Overview

The platform is organized into two interdependent subsystems: the Event Management Module and the Community Management Module. Both subsystems interact with a shared Supabase backend, ensuring consistent data state and real-time responsiveness across all features and user roles.

B. Technology Stack

EventHive was constructed using a carefully selected set of modern technologies. React.js powers the client-side interface, enabling a responsive and component-driven single-page application experience. Supabase serves as the backend-as-a-service layer, providing PostgreSQL for structured relational data management, Supabase Auth for email and password-based session handling, and Supabase Storage for media file management. An external AI image generation API is integrated during the event creation workflow to synthesize event banners from user-supplied text descriptions.

C. User Authentication and Role Management

Authentication in EventHive is managed through Supabase Auth, which handles user registration, login, and persistent session management. Upon successful authentication, users may create or join clubs. Within each club, the platform enforces a three-tier role hierarchy: the Owner, who holds full administrative authority including member management and club deletion; the Admin, who can create and manage events, approve members, and moderate community communications; and the Member, who can explore events, complete registrations, and engage within the community space.

D. Event Creation and AI Image Generation

Authorized users — those with Owner or Admin roles — may create events by specifying attributes such as name, date, time, venue, category, and description. A defining feature of EventHive is its AI-assisted image generation capability. During event creation, the organizer can supply a concise textual description of the event, upon which the system invokes an AI image generation API to produce a contextually relevant, professional-quality banner automatically. This eliminates reliance on external design tools and ensures that all event listings present a visually polished appearance without any graphic design expertise being required of the organizer.

E. QR Code Registration and Entry Verification

Upon completing registration for an event, the system creates a unique registration record in the PostgreSQL database and generates a corresponding URL pointing to that record's live status page. This URL is encoded into a QR code via the qrcode library and delivered to the registered user. On the day of the event, the organizer scans the participant's QR code, which navigates to the registration status page and allows a single-action entry confirmation. The approach ensures rapid, paperless, and tamper-resistant attendance verification without the need for printed tickets or manual roster checking.

F. Community and Club Management

Users may establish or join clubs within the platform. Each club maintains its own dedicated event board and a real-time chat space accessible to owners and administrators. This integrated communication channel facilitates coordinated planning between organizers without requiring external messaging tools. Members can browse their club's upcoming events, register for activities of interest, and maintain a personal history of their registrations and attendances.

G. Additional Platform Features

Beyond the core modules, EventHive incorporates several supplementary features that enhance usability. An event search and filtering interface allows users to locate events by name, category, or tag. Organizers can assign categories and descriptive tags to events to improve discoverability. Email and push notifications inform users of successful registrations and any subsequent event updates. An analytics dashboard provides organizers with visibility into registration volumes and attendance rates, supporting data-driven event planning.

H. System Architecture

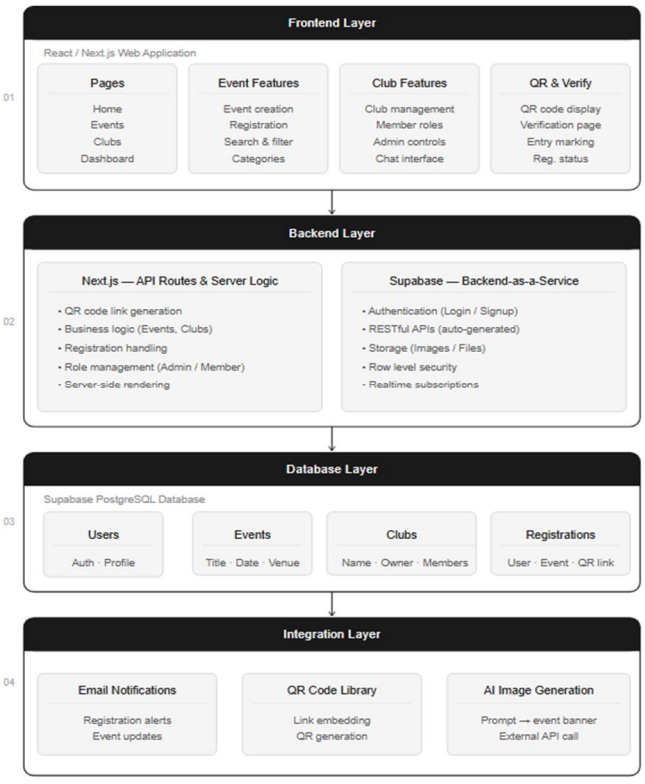
EventHive follows a client-server architecture in which the React.js frontend communicates with Supabase through its JavaScript SDK. All database operations, authentication workflows, and file storage interactions are routed through Supabase's REST and real-time APIs. This architecture enforces a clean separation of concerns: the frontend manages interface state and user interactions, while the backend handles data persistence, role enforcement, and service integration.

I. Workflow Summary

The typical user journey within EventHive proceeds as follows. A new user registers and authenticates via Supabase Auth, then creates or joins a club where they are assigned an appropriate role. An Admin or Owner creates an event, optionally generating a banner image through the AI prompt interface. Registered members browse available events and complete registrations, receiving a unique QR code pass upon confirmation. On the event day, the organizer scans each participant's QR code to verify and record attendance. Post-event, organizers review registration and attendance statistics through the analytics dashboard.

IV. SYSTEM ARCHITECTURE

The EventHive system follows a four-layer architecture consisting of frontend, backend, database, and integration layers. This layered design ensures clear separation of concerns, making the system scalable and easy to maintain. The frontend handles user interaction, while the backend processes logic using Next.js and Supabase services. The database manages structured data, and the integration layer connects external services such as email, QR code generation, and AI features.



A. Frontend Layer (User Interface Layer)

The frontend layer is responsible for handling all user interactions in the EventHive system. It is developed using React and Next.js, which provide a fast and dynamic user experience. This layer includes different pages such as Home, Events, Clubs, and Dashboard, where users can browse and manage activities. Users can create events, register for them, and manage club-related functionalities through an intuitive interface. It also supports features like search and filtering to easily find events. Additionally, the frontend displays QR codes for event entry verification. Overall, this layer focuses on providing a smooth, responsive, and user-friendly experience.

B. Backend Layer (Application Logic Layer)

The backend layer is responsible for processing all requests, enforcing business rules, and managing the core application logic of the EventHive system. It is implemented using Next.js API routes for server-side operations alongside Supabase as a backend-as-a-service provider. The Next.js component handles key functionalities such as QR code link generation, event and club business logic, registration handling, role management for admins and members, and server-side rendering for optimized page delivery. The Supabase component complements this by providing built-in authentication for user login and signup, auto-generated RESTful APIs, cloud-based storage for images and files, row-level security policies to protect data access, and real-time subscriptions that enable live updates across the platform. Together, these two components form a robust and scalable backend infrastructure.

C. Database Layer (Data Management Layer)

The database layer is responsible for the persistent storage, organization, and retrieval of all structured data within the EventHive system. It is powered by Supabase's PostgreSQL database, which provides a fully relational data model suited to the platform's needs. This layer maintains four core tables: the Users table, which stores authentication credentials and profile information; the Events table, which holds event details such as title, date, and venue; the Clubs table, which records club names, ownership, and member associations; and the Registrations table, which links users to events and stores the unique QR code link generated for each registration. The relational structure ensures data integrity, supports complex queries across entities, and enables efficient retrieval of information throughout all platform operations.

D. Integration Layer (External Services Layer)

The integration layer connects the EventHive system to a set of external services and third-party tools that extend the platform's capabilities beyond its internal logic. This layer consists of three primary integrations. The email notification service is responsible for sending automated alerts to users upon successful event registration and delivering updates when event details are modified. The QR code library handles the programmatic generation of unique QR codes by embedding registration-specific URLs, enabling seamless and paperless entry verification on event day. The AI image generation service accepts short text prompts provided by event organizers during event creation and returns contextually appropriate, professionally styled event banner images through an external API call. By delegating these specialized functions to dedicated external services, the integration layer enhances platform functionality while keeping the core application logic clean and maintainable.

V. RESULTS AND ANALYSIS

EventHive was developed and evaluated as a fully functional prototype encompassing all described core features. Systematic testing was conducted across each major module to assess correctness, usability, and response performance under representative usage conditions.

The authentication module, built upon Supabase Auth, demonstrated reliable handling of user registration, login, and session persistence across browser sessions. Role-based access control was verified at all three levels of the hierarchy owner, admin, and member confirming that each role operates strictly within its designated permission boundaries. Both the frontend routing logic and the Supabase Row Level Security policies were confirmed to block unauthorized access attempts consistently.

The event creation workflow, inclusive of the AI image generation feature, was evaluated using a diverse set of textual prompts representing different event types and themes. In all tested scenarios, the system returned contextually relevant, professionally styled event banners within a few seconds. Event categorization, tag assignment, and the search interface were tested against the PostgreSQL database, yielding accurate and performant query results across all cases.

The QR code registration and entry verification system was exercised throughout the full participant lifecycle within the prototype environment. Upon event registration, each test user received a QR code uniquely tied to their registration record. When scanned, the resulting page rendered correct participant information and allowed the organizer to confirm attendance through a single interaction. No printed materials or manual roster reconciliation were required at any stage of the process.

The club communication feature was validated to support real-time text exchange between admins and owners, leveraging Supabase's real-time subscription mechanism. Notification delivery for registration confirmations and event updates operated correctly. The analytics dashboard accurately reflected live registration counts and cumulative attendance figures throughout testing.

Taken together, the prototype results confirm that EventHive successfully integrates event management and community coordination within a single cohesive platform.

The AI-assisted image generation and QR-based entry verification features in particular represent meaningful functional differentiators relative to existing solutions in the market.

TABLE II Feature Testing Summary

Feature	Status	Outcome
User Authentication & RBAC	Tested	Passed — all roles verified
Event Creation with AI Image	Tested	Consistent generation in seconds
QR Code Registration	Tested	Unique token per user per event
QR Code Check-in	Tested	Single-scan, paperless entry
Club & Community Management	Tested	Roles and membership enforced
Real-time Chat	Tested	Supabase real-time subscription
Event Search & Filtering	Tested	Accurate PostgreSQL queries
Analytics Dashboard	Tested	Registration & attendance data

VI. CONCLUSIONS

This application presented EventHive, a web-based platform developed to address the practical shortcomings of existing event management and community organisation tools. By bringing together club administration, role-based access control, event management, QR code-based attendance verification, and AI-powered banner generation within a single cohesive system, EventHive demonstrated that it is possible to serve both event logistics and community engagement needs without requiring users to rely on multiple disconnected tools. Built using React.js and Supabase, the platform successfully implemented all core functionalities, including real-time community interaction, secure authentication, and dynamic QR code verification. The system proved particularly relevant for academic institutions, student organisations, and professional groups that demand structured yet flexible event coordination. The AI image generation feature meaningfully reduced the design burden on organisers, while the QR-based check-in process improved both security and efficiency at the point of entry.

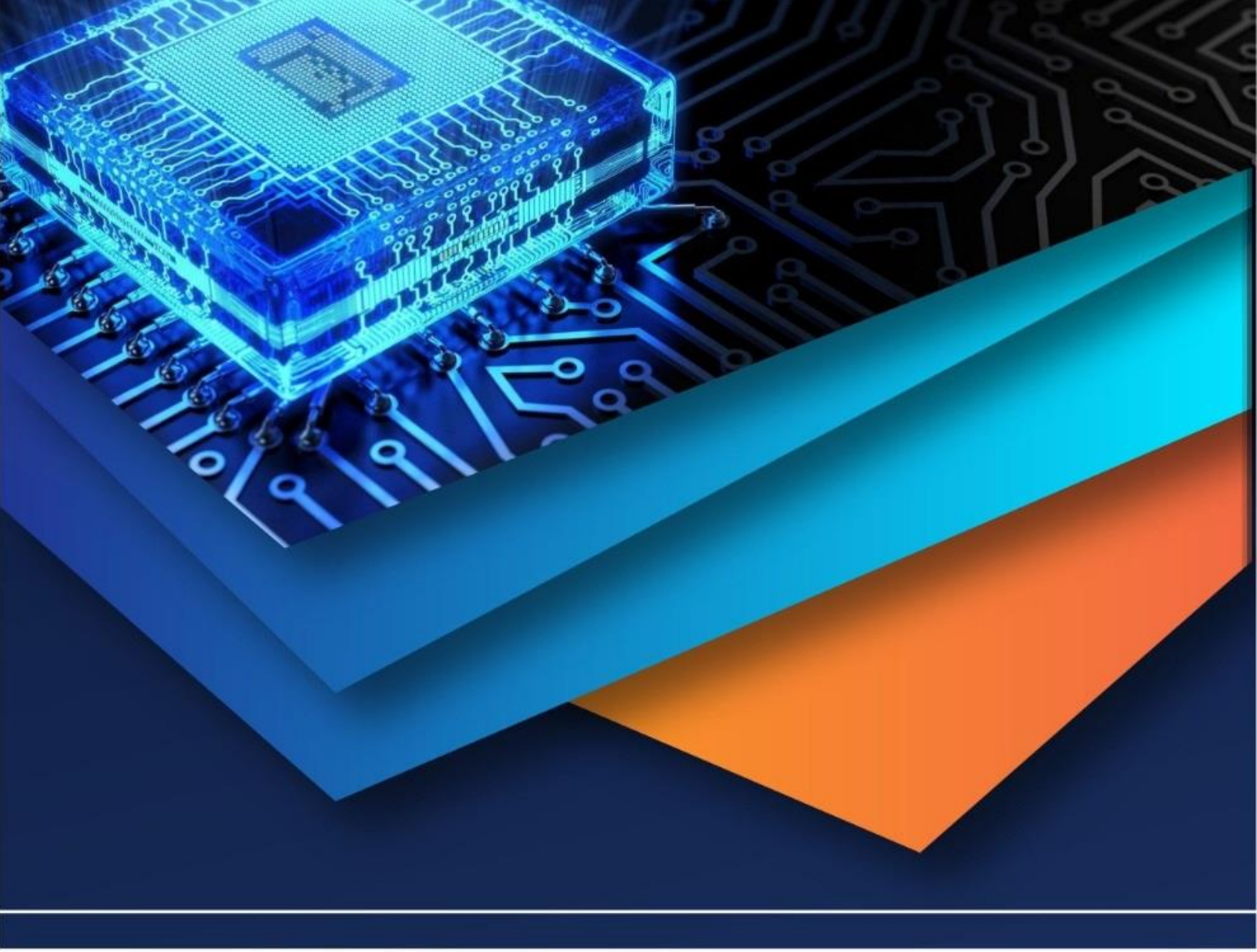
A. Future Scope

Several development directions are envisioned for subsequent iterations of EventHive. A native mobile application for both iOS and Android platforms is planned to broaden accessibility and improve usability in on-site event contexts. Calendar integration with widely used services such as Google Calendar and Microsoft Outlook would allow participants to synchronize events directly to their personal schedules. An enhanced analytics dashboard featuring visual data representations and exportable attendance reports would further support data-driven event management. The AI image generation module could be extended to offer multiple stylistic presets and flexible aspect ratios to suit diverse event formats. Additionally, future development may explore the incorporation of machine learning-based event recommendation systems capable of suggesting relevant events to users based on historical participation patterns and stated preferences.

REFERENCES

[1] Patel, R., Mehta, S., and Joshi, A., "Limitations of existing event management platforms for closed community-driven groups in academic and professional settings," Journal of Digital Systems and Community Platforms, vol. 9, no. 2, pp. 45–61, 2022.
[2] Sharma, D. and Verma, K., "Influence of integrated digital community platforms on organizational participation and collaboration," International Journal of Human-Computer Interaction, vol. 37, no. 6, pp. 512–529, 2021.
[3] Nguyen, T., Tran, L., and Pham, H., "Dynamic QR code systems for contactless event attendance verification in post-pandemic environments," Journal of Event Technology and Management, vol. 5, no. 1, pp. 22–35, 2021.
[4] Rombach, R., Blattmann, A., Lorenz, D., Esser, P., and Ommer, B., "High-resolution image synthesis with latent diffusion models," in Proc. IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR), pp. 10684–10695, 2022.
[5] Ferraiolo, D., Kuhn, R., and Chandramouli, R., "Role-based access control in modern cloud-based collaborative web systems," IEEE Transactions on Dependable and Secure Computing, vol. 17, no. 4, pp. 788–801, 2020.
[6] Bons, M., Krueger, J., and Alves, P., "Comparative evaluation of backend-as-a-service platforms: Firebase, Supabase, and AWS Amplify for relational web applications," ACM SIGWEB Newsletter, pp. 1–14, 2023.

[7] Zhang, Q., Chen, M., and Li, L., "Design and implementation of cloud-based event management systems for scalable user engagement," IEEE Access, vol. 9, pp. 112345–112358, 2021.
[8] Alshamrani, A., Myneni, S., Chowdhary, A., and Huang, D., "A survey on advanced access control models in cloud computing," IEEE Communications Surveys & Tutorials, vol. 22, no. 3, pp. 1819–1855, 2020.
[9] Kaur, P. and Singh, K., "Enhancing user engagement in online platforms through integrated communication and event systems," Journal of Web Engineering, vol. 20, no. 4, pp. 1023–1041, 2021.
[10] Sarker, I. H., "Machine learning-based security and privacy in IoT: Applications and challenges," Internet of Things Journal, vol. 8, no. 2, pp. 112–128, 2021.
[11] Li, X., Jiang, P., Chen, T., Luo, X., and Wen, Q., "A survey on the security of blockchain systems," Future Generation Computer Systems, vol. 107, pp. 841–853, 2020.
[12] Chandra, S. and Kumar, A., "A study on QR code-based smart attendance systems," International Journal of Computer Applications, vol. 183, no. 15, pp. 20–25, 2021.
[13] Patel, R. and Shah, M., "User-centric design approaches for modern web applications," Journal of UX Design and Human Factors, vol. 6, no. 2, pp. 55–70, 2022.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY