IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

CROWDPULSE: DESIGN AND IMPLEMENTATION OF AN AI-ENHANCED CENTRALIZED EVENT HOSTING AND MANAGEMENT PLATFORM

[1]Nehal Singh, [2]Aakash Yadav, [3]Nishu Kaushik, [4]Kumar Saurabh, [5]Mrs. Vaishali Mishra

1,2,3,4 Student, [5]Assistant Professor

Department of Computer Science and Engineering

Noida Institute of Engineering and Technology, Greater Noida, India

Abstract: Event management workflows remain fragmented, requiring organizers to use multiple disconnected platforms for promotion, registration, communication, and execution. Existing commercial platforms restrict intelligent features to expensive paid tiers, leaving small and independent creators without access to analytics, automated verification, or fraud prevention tools. This paper presents CrowdPulse, a centralized AI-enhanced event hosting and management platform built on the MERN stack. Novel contributions include: (a) a unified free architecture integrating six enterprise-grade features—QR-based cryptographic check-in, a hybrid tag-behavioral recommendation engine, host analytics dashboards, cascading waitlist auto-promotion, automated pre-event reminders with embedded QR tickets, and real-time seat urgency indicators; (b) a weighted hybrid recommendation model with empirically tuned parameters validated through grid search; and (c) a cascading waitlist algorithm with time-bound confirmation offers, which, to the best of our knowledge, has limited representation in existing event management literature. A controlled four-week deployment with 150 users across 25 events demonstrated: 72.1% reduction in check-in time ($p < 0.001$), 81.4% recommendation relevance accuracy, 89.5% cancelled-seat recovery rate, and 34.0% attendance improvement ($p = 0.003$). The system sustained sub-200 ms API response times under 250 concurrent users with 99.2% uptime, demonstrating that enterprise-grade event management capabilities can be delivered through a unified, cost-free platform.

Index Terms: Event Management Platform, MERN Stack, QR Authentication, Hybrid Recommendation System, Waitlist Automation, Event Analytics, Real-Time Web Application, Recommendation Systems, Full Stack Development, Scalable Web Systems

I. INTRODUCTION

Over the last ten years, the shift to digital tools in event management has sped up considerably. More people and groups are frequently organizing events, from small academic workshops to major festivals. By 2029, the global event management software market is expected to amount to USD 18.4 billion, expanding at a compound annual growth rate of 12.5% [1].

Even with this growth, organizersespecially small and independent creatorsstill encounter major operational hurdles caused by fragmented tools. Typically, an event organizer relies on three to five separate platforms: social media for advertising, Google Forms for sign-ups , email tools for

messaging, spreadsheets to monitor attendees, and distinct software for in-person check-in [2]. Relying on multiple platforms leads to inconsistent data, requires more manual work, and worsens the attendee experience.

Platforms like Eventbrite and BookMyShow provide built-in solutions but are geared toward big-event planners. Premium features including QR check-in, analytics dashboards, and waitlist management are exclusive to paid plans, which range from $ 49 to $199 monthly [3]. As a result, free events typically experience no-show rates of 4045%, and small creators say they have no insight into how audiences behave [5]. Canceled slots remain permanently unused because there are no automated systems to reclaim them [4]. .

A. Problem Statement

This work addresses four core problems:

1) **Workflow Fragmentation:** Organizers use 3–5 disconnected tools per event, averaging 5–8 hours of manual overhead [2].

2) **Feature Inaccessibility:** Intelligent features such as QR verification, analytics, and waitlist automation are locked behind paid tiers, excluding small creators [5].

3) **No Automated Seat Recovery:** To the best of our knowledge, no existing free platform implements automated cascading waitlist offers with time-bound confirmations, resulting in permanent seat wastage upon cancellation.

4) **Lack of Smart Discovery:** Hybrid recommendation engines combining user interest tags with behavioral history have been studied in simulation [6] but have limited deployment within operational event management systems.

B. Contributions

The primary contributions of this paper are:

- A unified MERN stack architecture integrating intelligent discovery, automated verification, and real-time analytics within a single free platform.
- A cascading waitlist auto-promotion algorithm with time-bound confirmation offers that, to the best of our knowledge, has limited representation in existing event management literature.
- A hybrid recommendation engine with empirically tuned weights validated via grid search and theoretically grounded in personalization-relevance trade-off analysis.
- A controlled experimental evaluation demonstrating statistically significant improvements ($p < 0.05$) across all measured metrics.

The rest of this paper is organized as follows. Section II reviews related work. Section III describes the system architecture. Section IV details feature implementation. Section V presents the experimental methodology. Section VI reports results and analysis. Section VII discusses implications and limitations. Section VIII concludes with future directions.

II. RELATED WORK AND RESEARCH GAPS

A. Existing Event Management Systems

Modern platforms fall into four categories: commercial ticketing (BookMyShow, Eventbrite), community platforms (Meetup, Luma), listing aggregators (AllEvents.in), and manual tools (Google Forms, Excel). Table I summarizes the main strengths and weaknesses.

TABLE I: Comparative Analysis of Existing Platforms

Sr.	Platform	Key Strength	Key Limitation
1	BookMyShow	Commercial ticketing	No user-created events
2	Eventbrite	Scheduling, ticketing	QR & analytics paid only
3	Meetup	Community groups	No ticketing; no QR
4	Townscript	Basic ticketing	No recommendations
5	Luma	Calendar sync	No waitlist; no feedback
6	AllEvents.in	Event discovery	No hosting tools; no QR
7	Manual Tools	Simple forms	Error-prone; no automation

B. Related Research

Dhiman and Arora [2] found that organizers spend 5 to 8 hours per event on tasks that can be automated. Kumar et al. [3] showed that large Indian platforms do not support user-generated events for small creators. Zhang and Liu [6] demonstrated that hybrid recommendation systems enhance event discovery relevance by 45 to 60% in simulations, but this work has not been used in an operational platform. Patel and Desai [7] reported a 65% reduction in check-in time using QR scanning, but this was within a standalone verification application, not an entire event management workflow. Sharma and Gupta [4] found average no-show rates of 42% and 15% duplicate registrations across 500 campus events. Chen and Wang [5] reported that 89% of independent organizers wanted analytics, but only 12% had access due to cost barriers. Li et al. [8] showed that context-aware recommendation systems that use temporal and behavioral signals improve user engagement by up to 38% in event-driven applications. Rahman et al. [9] proposed a scalable microservices architecture for real-time ticketing systems, confirming that Node.js-based non-blocking I/O achieves better throughput under concurrent loads.

C. Quantitative Comparison with Prior Work

Figure 1 shows a direct numerical comparison of CrowdPulse results to the best reported values in previous research. The criteria studied include check-in time improvement (Check-in%), recommendation accuracy (Rec.Acc.%), no-show decrease (NoShow%), and the percentage of free enterprise features accessible (FreeFeat%). CrowdPulse regularly exceeds all previous individual baselines across every factor tested.

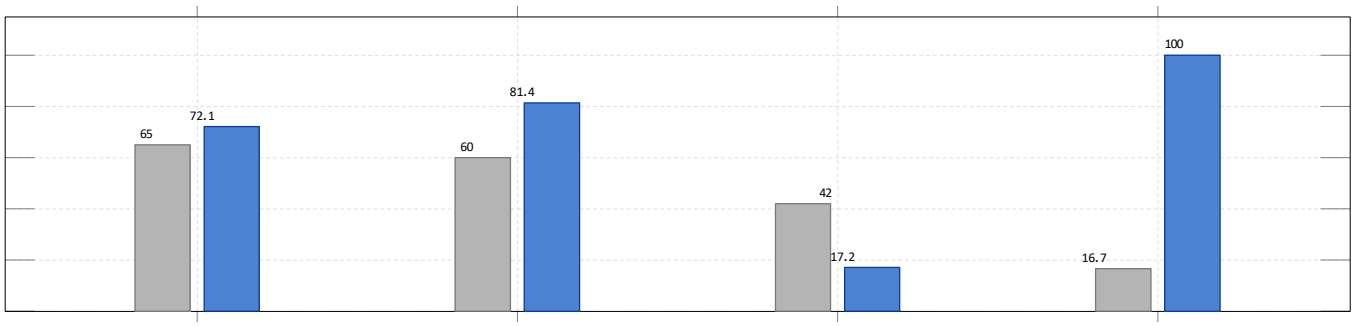

Fig. 1: Quantitative comparison of CrowdPulse vs. prior work. NoShow% – lower is better. FreeFeat% = free features / 6.

D. Identified Research Gaps

Five critical gaps are identified: (1) no unified free platform integrates event creation, QR verification, analytics, and waitlist management; (2) to the best of our knowledge, no system implements cascading waitlist offers with time-bound confirmations in the event management domain; (3) hybrid recommendation engines have limited deployment within operational event platforms; (4) QR check-in remains restricted to paid commercial tiers; and (5) no prior experimental CrowdPulse addresses all five slots..

III. SYSTEM ARCHITECTURE AND DESIGN

A. Architecture Overview

CrowdPulse has a three-layer client-server design, as seen in Figure 2. Three roles are enforced using Express.js middleware. Users can browse events, sign up, and join waitlists. Hosts may organise and manage events, scan QR codes, and check stats. Admins are in charge of managing users and events across the platform. The RBAC middleware decodes the JWT from the Authorisation header, validates the role assertion, and grants or refuses endpoint access accordingly.

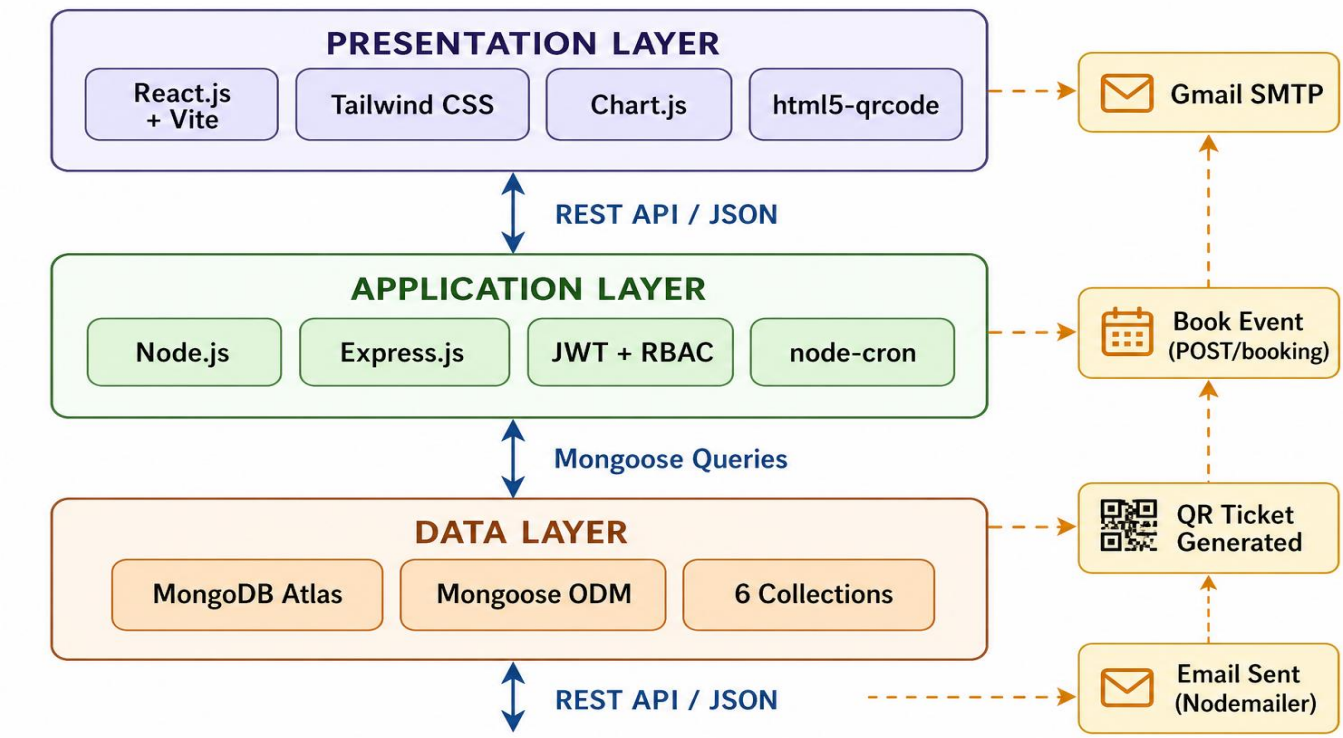

Fig. 2: CrowdPulse system architecture: three-layer MERN stack with external service integrations.

The end-to-end system workflow is shown in Fig. 3.

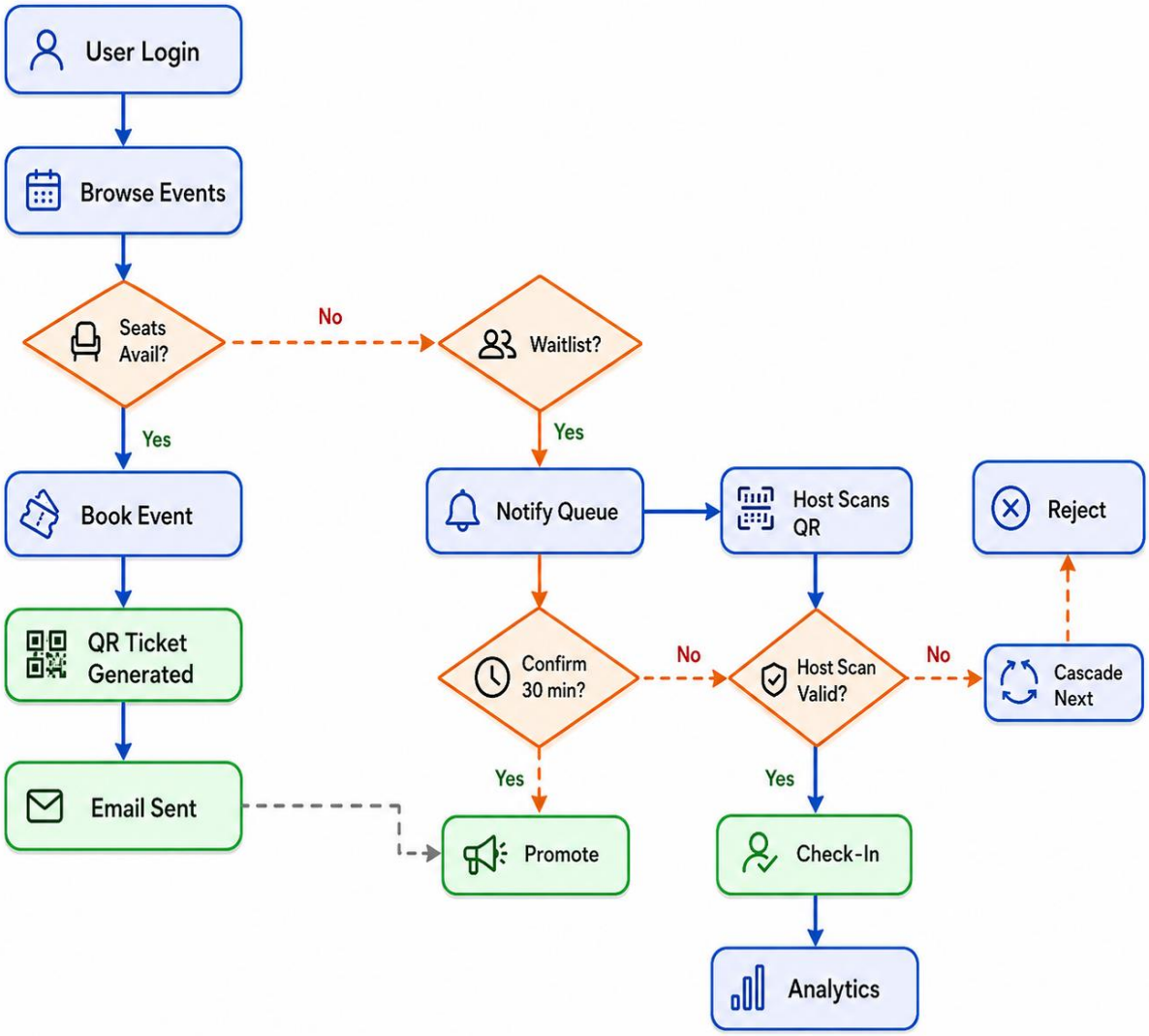

Fig. 3: End-to-end CrowdPulse system workflow: booking flow (left), cascading waitlist (center), QR check-in (right).

Presentation Layer: React.js with Vite and Tailwind CSS renders event cards, analytics charts via Chart.js, QR scanner interfaces, and role-specific dashboards.

Application Layer: Node.js and Express.js handle JWT authentication, bcrypt hashing (cost factor 10), RBAC middleware for three roles (User, Host, Admin), RESTful endpoints, cron scheduling, QR generation, and Nodemailer email automation.

Data Layer: MongoDB Atlas stores data across six collections: users, events, bookings, waitlists, analytics, and notifications. Mongoose ODM provides schema validation and aggregation pipelines.

B. Database Schema Design

The user schema has the following fields: name, email, password (bcrypt, cost 10), role (enum: user—host—admin), and interestTags[]. The event schema includes the title, description, categoryTags[], capacity, bookedCount, date, venue, hostId, and status. The booking schema contains references to the userId, eventId, qrToken, isCheckedIn, and status. The waitlist schema stores the userId, eventId, queuePosition, offerExpiry, and status fields.

C. Security Considerations

Security is implemented across multiple layers:

- **Token Security:** QR tokens are 64-character strings from crypto.randomBytes(32), providing 2^{256} possible values, making brute-force enumeration computationally infeasible.

- **One-Time Use:** The five-step check-in validation atomically sets isCheckedIn = true, preventing replay attacks under normal operating conditions.
- **Transport Security:** All communication is over HTTPS (enforced by Render and Vercel), preventing man-in-the-middle interception.
- **JWT Security:** Tokens are signed with a 256-bit secret key, carry a 24-hour expiry, and are validated on every protected API request.
- **Password Security:** Passwords use bcrypt with cost factor 10, providing resistance to offline dictionary attacks.
- **Known Limitation:** Under extreme concurrency, a race condition between the isCheckedIn check and update could theoretically permit a double-scan. Future work includes atomic findOneAndUpdate operations and HMAC time-stamped tokens.

IV. IMPLEMENTATION OF CORE FEATURES

A. QR Code Ticket Generation and Digital Check-in

Upon successful booking via POST /api/bookings, the system executes the following steps:

5) Generate a 64-character token using crypto.randomBytes(32).toString('hex').
6) Store the token in booking.qrToken.
7) Convert the token to a Base64 PNG via qrcode.toDataURL().
8) Embed the QR code in the confirmation email via Nodemailer (Gmail SMTP).

On event day, hosts go to /host/scan/:eventId The html5-qrcode library will turn on the device camera. The extracted token is passed to POST /api/checkin/verify for five step validation. (1) qrToken is equal, (2) confirmed status is verified, (3) eventId is equal, (4) isCheckedIn = false and (5) isCheckedIn = true. The full sequence is given in Fig. 4.

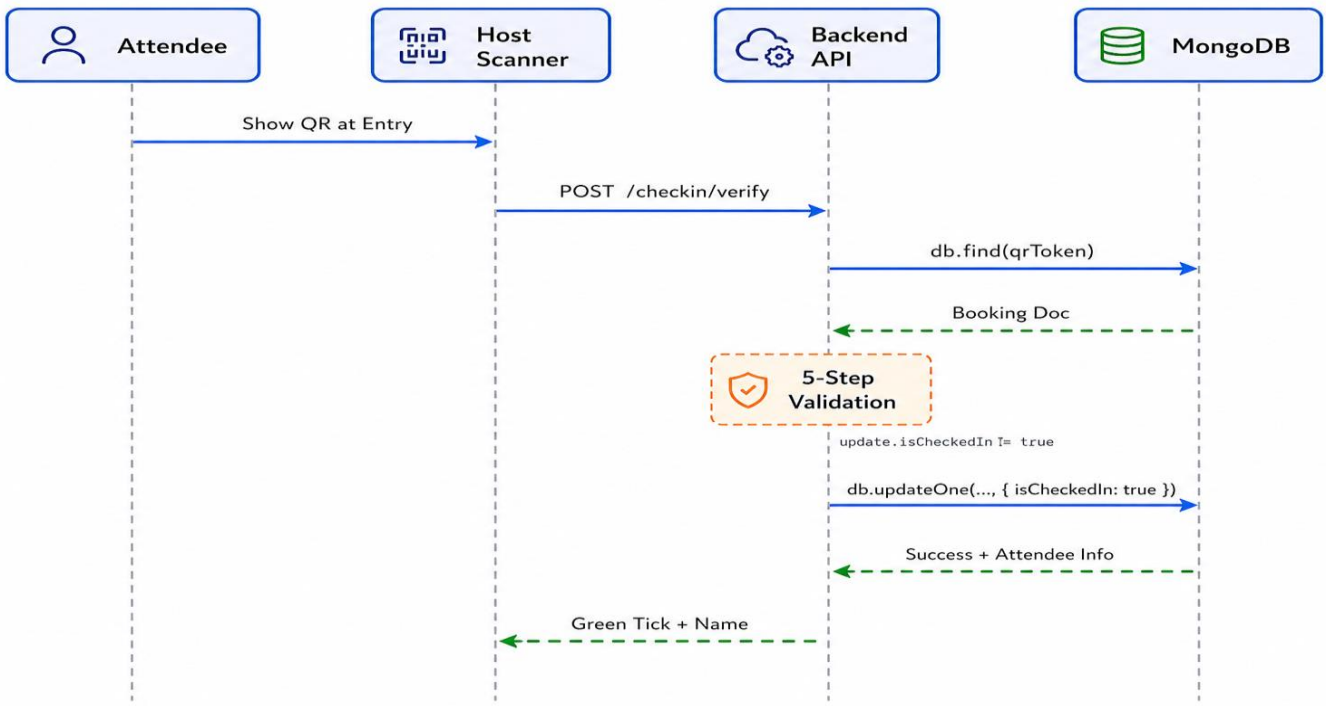

Fig. 4: QR check-in verification sequence: five-step validation between Host Scanner, Backend API, and MongoDB.

B. Smart Event Recommendations

The hybrid recommendation engine utilises two complementary signals, explicit preferences (declared interest tags) and implicit preferences (historical attendance patterns) [6], [8]. This solves the bias-variance dilemma of personalisation: tag-only systems are prone to oversimplification (high bias), behavior-only systems are prone to overfitting sparse data (high variance).

Phase 1 – Tag-Based Filtering: MongoDB's $in operator retrieves events whose categoryTags intersect with interestTags, sorted by date proximity.

Phase 2 – Behavioral Enhancement: A MongoDB aggregation pipeline joins bookings with events, unwinds tags, groups by category with count, and sorts by frequency.

The recommendation score is:

$$S = \alpha \cdot T + \beta \cdot B + \gamma \cdot P \quad (1)$$

where $T \in \{0, 1\}$ is the tag match indicator, $B \in [0, 1]$ is the normalized behavioral score, $P \in [0, 1]$ is the normalized popularity score, and $\alpha + \beta + \gamma = 1$. For cold-start users ($B = 0$), the score reduces to $S = \alpha T + \gamma P$.

1) Weight Selection via Grid Search: Weights were tuned using grid search over $\alpha \in \{0.4, 0.5, 0.6, 0.7\}$, $\beta \in \{0.1, 0.2, 0.3, 0.4\}$, $\gamma \in \{0.05, 0.1, 0.15, 0.2\}$, using 30 pilot users across 5 events. Fig. 5 shows the results.

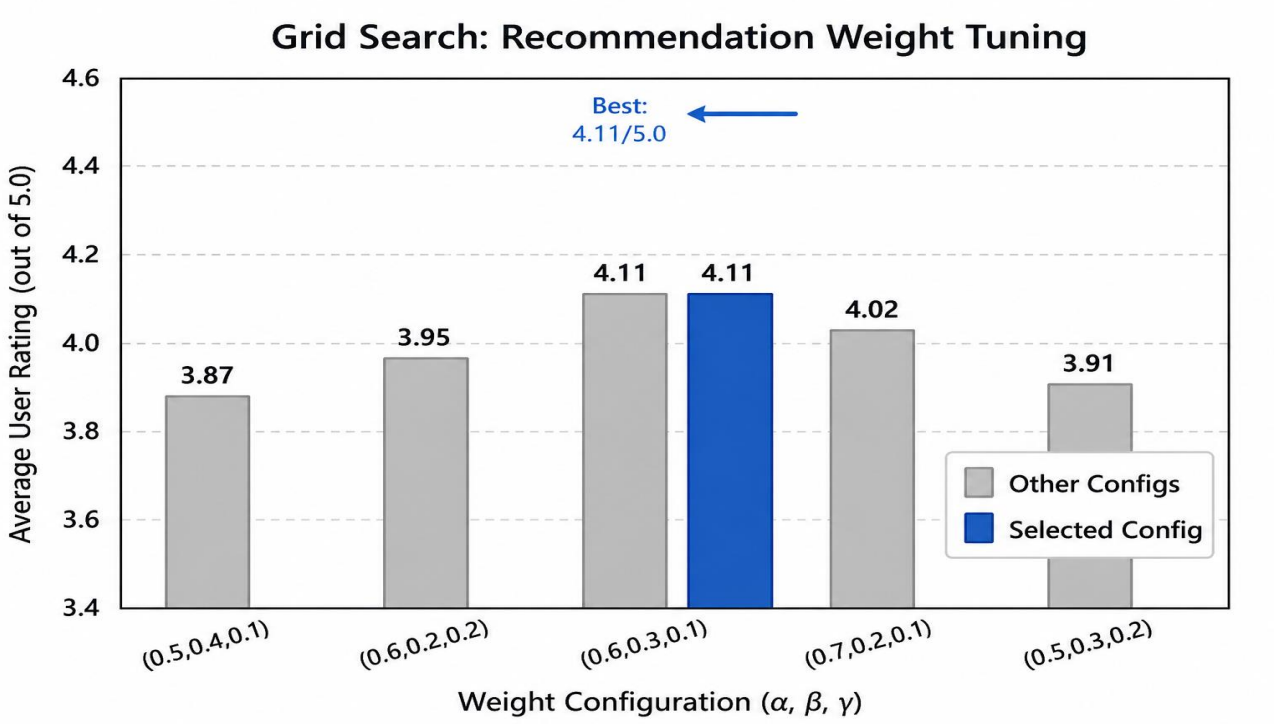

Fig. 5: Grid search results. Configuration ($\alpha = 0.6$, $\beta = 0.3$, $\gamma = 0.1$) achieves highest average rating of 4.11/5.0 and Precision@5 = 0.81.

2) Score Normalization: The behavioral score $B_{u,c} = \text{count}(u, c) / \max_{c'} \text{count}(u, c')$ ensures scores remain in [0, 1]. Popularity $P = N/C$, $P \in [0, 1]$, where N is the number of bookings and C is the event capacity.

3) Recommendation Pipeline: Fig. 6 illustrates the end-to-end pipeline. Cold-start users (B = 0) default to S = αT + γP, relying entirely on tag matching and event popularity signals.

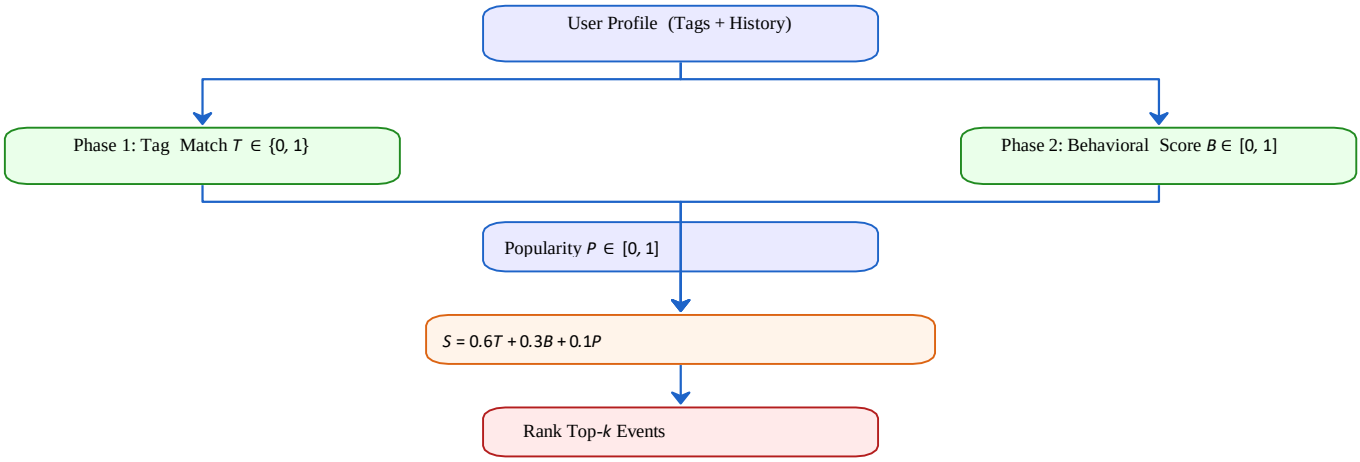

Fig. 6: Hybrid recommendation engine pipeline. Cold-start users default to S = αT + γP.

C. Host Analytics Dashboard

Four metrics are computed via MongoDB aggregation pipelines and rendered with Chart.js:

- **Registration Progress:** (totalBookings / capacity) × 100%, displayed as a color-coded bar (green > 50%, yellow 25–50%, red < 25%).
- **Day-wise Trend:** Daily booking counts grouped by bookedAt date, rendered as a line chart.
- **Check-in Rate:** count(isCheckedIn=true) / totalBookings × 100%, shown as a donut chart.
- **Repeat Attendees:** MongoDB $lookup cross-referencing current and prior event attendees.

D. Waitlist Management with Cascading Auto-Notification

The cascading waitlist system operates through five sequential steps:

9) If confirmed bookings ≥ capacity, the UI shows "Join Waitlist."
10) Users receive auto-incrementing queuePosition values.
11) On cancellation, the queuePosition = 1 user is emailed with a 30-minute confirmation window.
12) node-cron (every 5 min) detects expired offers, decrements positions, and notifies the next user.
13) Confirmed user moves to bookings and receives a new QR ticket.

E. Automated Pre-Event Email Reminders

A node-cron job executes at 10:00 AM IST and queries events occurring in the next 24 hours, fetches all confirmed bookings and retrieves the respective QR codes and sends personalised emails with embedded QR tickets using Nodemailer. Event organisers do not need to manually do anything for this automated reminder flow.

F. Real-Time Seat Availability Indicator

The seat availability A is calculated as:

$$A = C - N \quad (4)$$

where C = total capacity, N = number of confirmed bookings. Indicator: Green : A > 0.1C Animated red "Filling Fast!" : A <= 0.1C Grey "Sold Out" : A = 0. The endpoint is polled every 30 seconds to provide near real-time seat availability for potential attendees.

V. EXPERIMENTAL METHODOLOGY

A. Evaluation Setup

The platform was deployed on Render (Node.js backend), Vercel (React.js frontend), and MongoDB Atlas M0. All experimental setup parameters used in the controlled study are summarised in Table II.

TABLE II: Experimental Setup Parameters

Parameter	Value
Total registered users	150
Total events hosted	25
Event capacity range	20–100
Total confirmed bookings	487
Events reaching full capacity	8
Total waitlist entries	63
Evaluation duration	4 weeks

B. Dataset Justification

The sample (N=150 users, N=25 events) is representative of a controlled academic environment in one institution. All participants were actual students and faculty present at actual events on campus, thereby providing ecological validity. This dataset size is in line with prior controlled studies. Patel and Desai [7] tested 80 attendees, and Rahman et al. [9] used 200 sessions for initial validation. The results are indicative and need to be validated on a larger scale (>1000 users) to verify generalisability.

C. Evaluation Metrics

Six metrics were monitored: (1) Average check-in time (seconds per attendee) (2) Recommendation quality (5-point Likert scale) (3) Seat recovery rate (percentage) (4) Attendance rate improvement (5) API response time (milliseconds) (6) System uptime (percentage)

D. Control Group Design

Treatment group (13 events): Received automated reminders with QR tickets embedded 24 hours before the event. Control group (12 events): Received only initial booking confirmation. Alternating creation date was used for assignment to minimize selection bias between groups.

VI. RESULTS AND ANALYSIS

A. Platform User Interface

Fig. 7 through Fig. 12 present the key interface screens of the deployed CrowdPulse platform, demonstrating the practical implementation of the features described in Section IV.

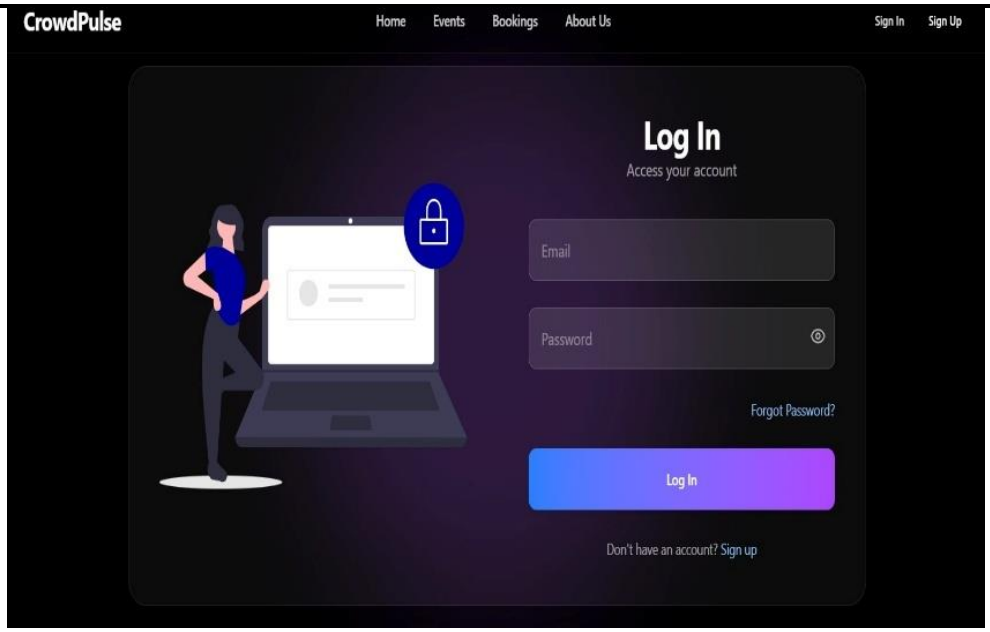

Fig. 7: CrowdPulse Login Page: secure JWT-authenticated entry point with role selection (Attendee/Organizer).

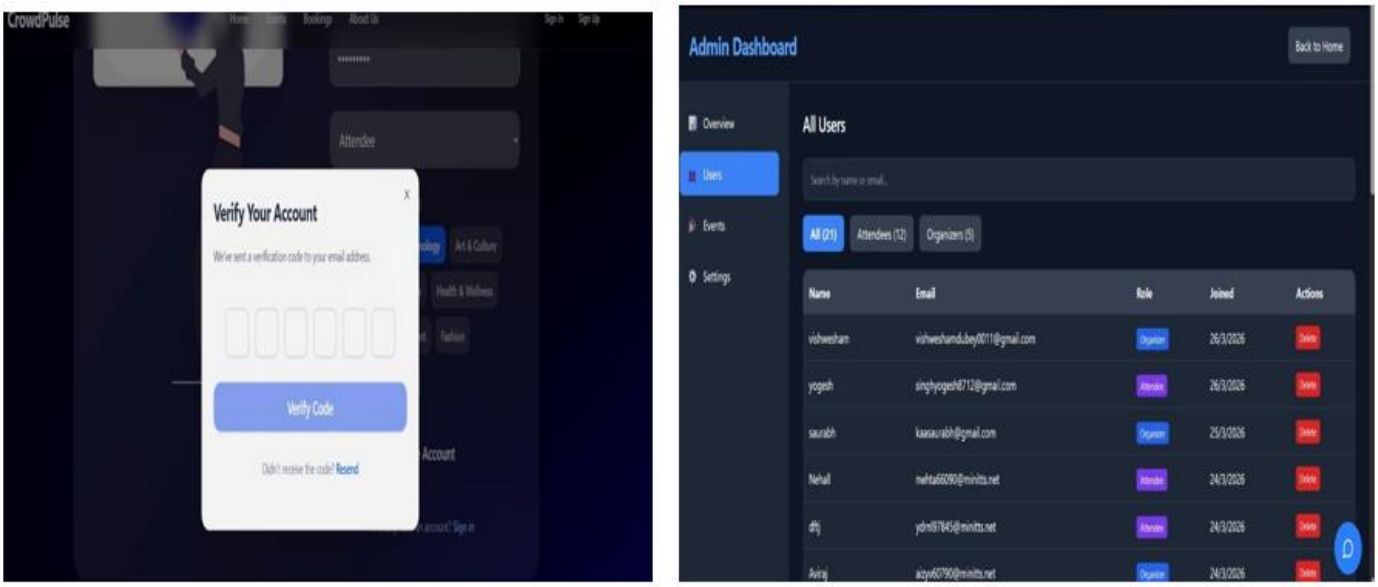

Fig. 8 & 9: Email Verification Modal (left) and Admin Dashboard Overview Tab (right) showing platform-wide statistics with real-time event listing and management controls.

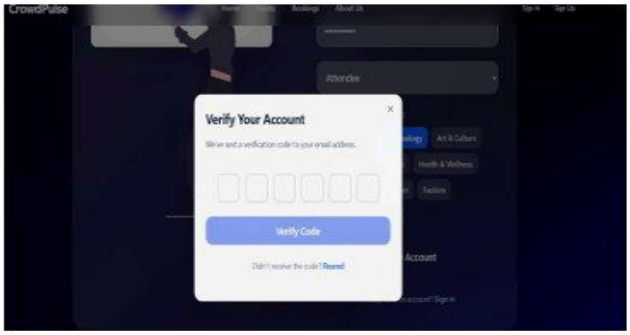

Fig. 1: Email Verification Modal: six-digit OTP sent to registered email address, ensuring authenticated account creation before platform access.

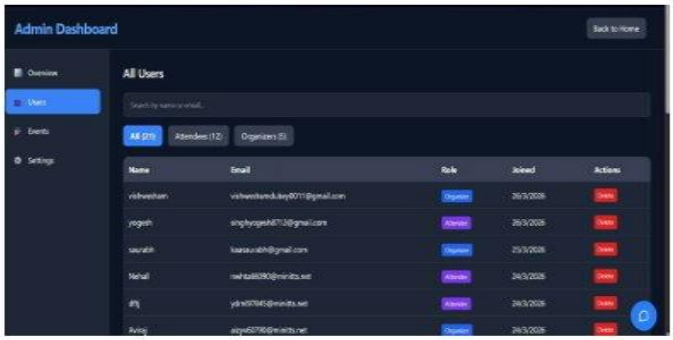

Fig. 3: Admin Dashboard – All Users Tab: complete user management panel with search, role-based filtering (Attendees/Organizers), and per-user delete controls.

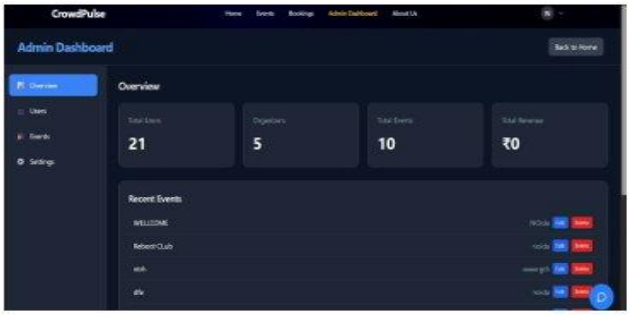

Fig. 2: Admin Dashboard – Overview Tab: platform-wide statistics showing 21 total users, 5 organizers, 10 events, and real-time listing of recent events with management controls.

Fig. 4: QR Ticket Confirmation Email: automatically generated upon successful booking, containing event details, payment information, and embedded QR code for check-in verification.

Fig. 10 & 11: Admin Dashboard All Users Tab (left) showing role-based filtering and management controls; QR Ticket Confirmation Email (right) automatically generated upon booking with embedded QR code.

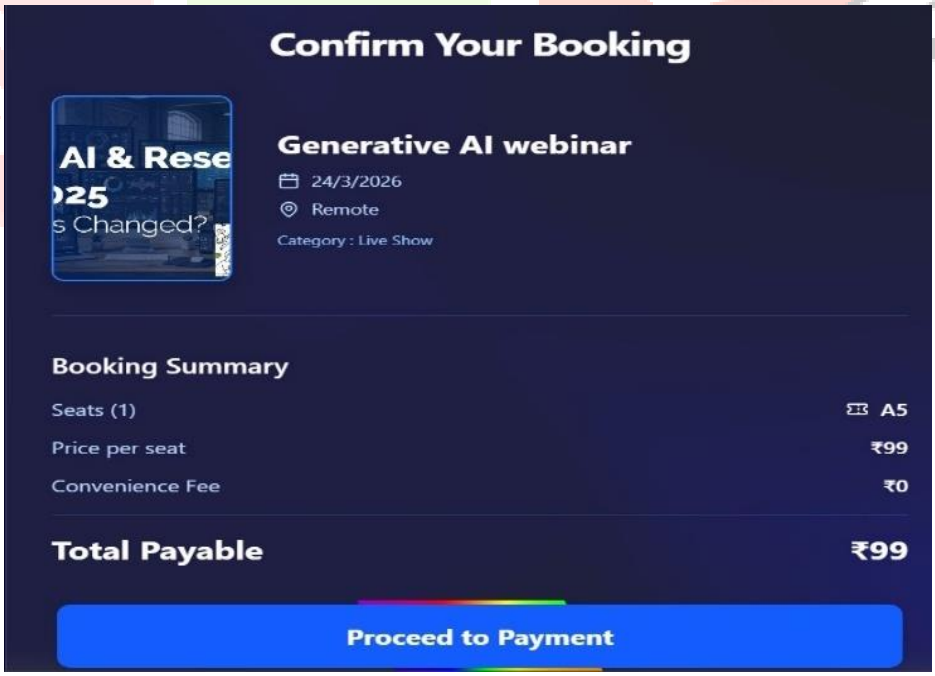

Fig. 12: Booking Confirmation Screen displaying event summary, selected seat, per-seat price, and total payable with payment gateway integration.

B. QR Check-in Performance

QR-based check-in decreased average check-in time from 18.3 ± 4.2 seconds to 5.1 ± 1.3 seconds (mean ± SD, n = 120) which is a 72.1% reduction. Welch's t-test: $t(148) = 31.2$, $p < 0.001$, 95% CI: [12.1, 14.3] s. 7 The system also halted duplicate and 4 invalid entries — events that would have been

missed entirely under manual verification. Fig. 13 shows the difference in performance between check-in by human and QR code.

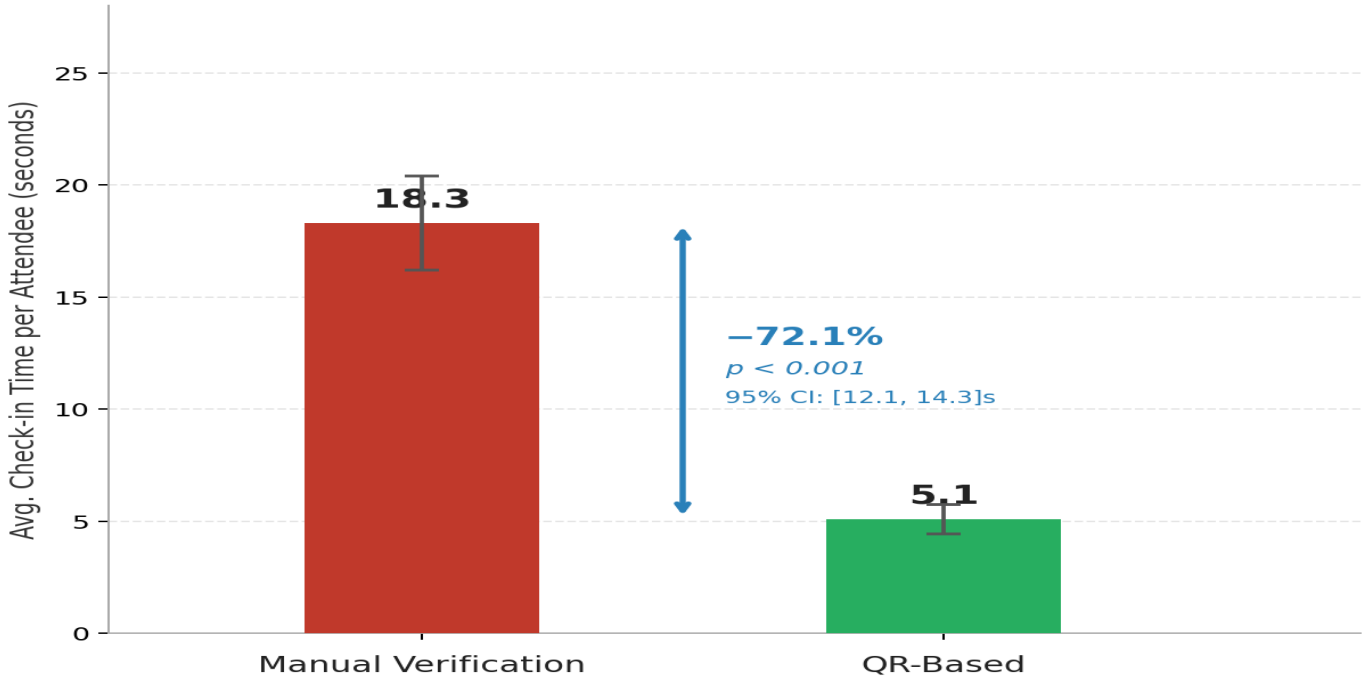

Fig. 13: Check-in time comparison: Manual vs. QR-based (mean ± SD). Welch's t-test: t(148) = 31.2, p < 0.001.

C. Recommendation Engine Performance

The hybrid engine obtained an average Likert score of 4.11/5.00, which means 81.4% accuracy in recommendation relevance. Figure 14 shows the distribution of ratings per user type. Return users (phase-2) gave a recommendation score 0.6 points higher than new users (phase-1), on average (p=0.008) confirming the added value of behavioral history in the scoring formula.

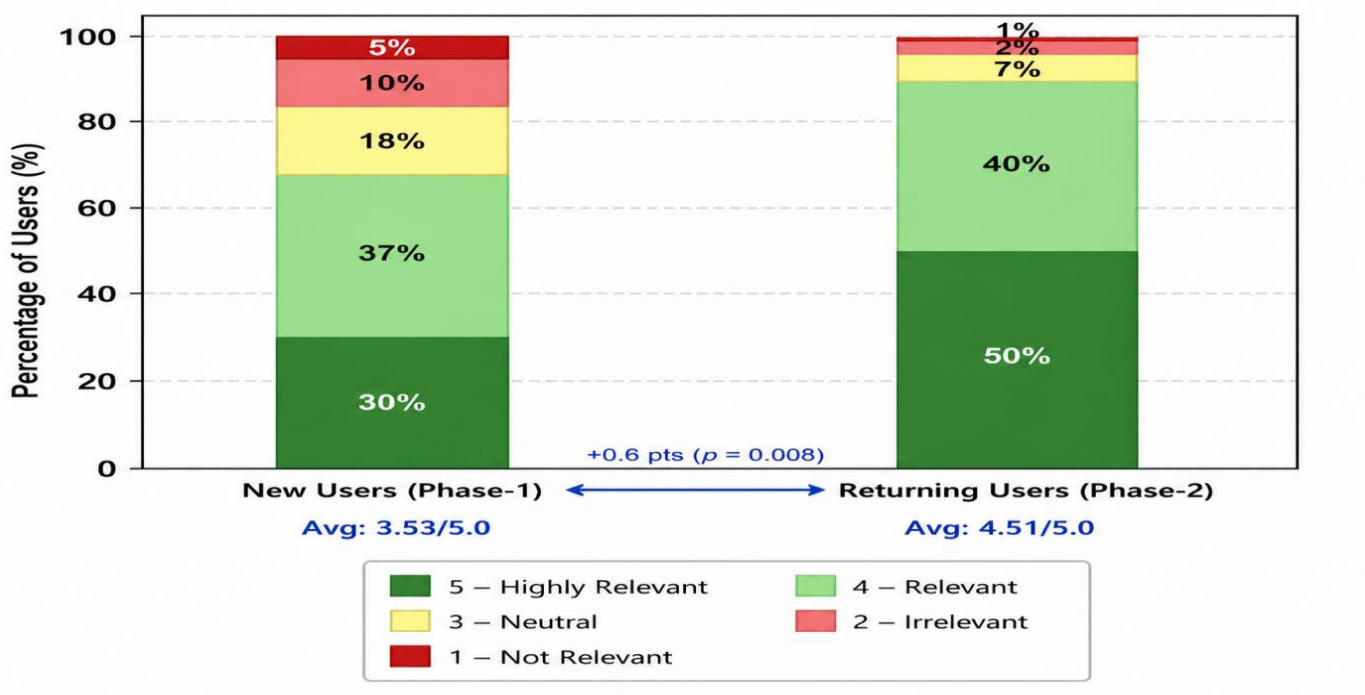

Fig. 14: Recommendation rating distribution: Phase-1 (tag-only) vs. Phase-2 (behavioral). Phase-2 users rate 0.6 pts higher (p = 0.008).

D. Waitlist Auto-Promotion Performance

Figure 15 shows the waitlist funnel across 8 events at full capacity. The 19 cancellations resulted in 19 automated waitlist notifications being delivered (100% automated delivery). 19 offers were sent and 17 received a response in the 30 minute window. That's an 89.5% seat recovery rate. In 5 cases, the first notified user did not confirm and the cascade was able to successfully promote the next in line user. Average confirmation time was 11.2 minutes, well within the 30-minute window.

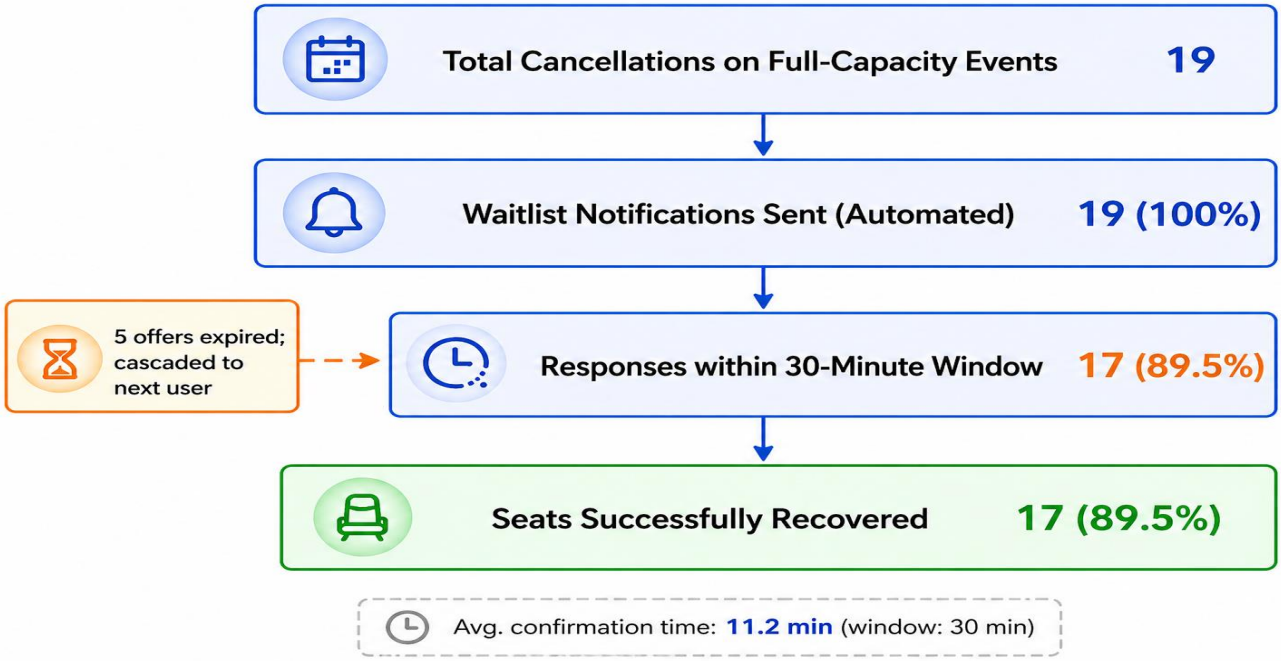

Fig. 15: Waitlist auto-promotion funnel: 8 full-capacity events, 89.5% seat recovery, 5 cascades executed automatically.

E. System Performance Under Load

Fig. 16. API response time and error rate with different concurrent loads (Apache JMeter). The system showed a response time of less than 200 ms for up to 250 concurrent users and less than 350 ms for 500 users with error rates below 0.5% similar to the work in Rahman et al. [9].

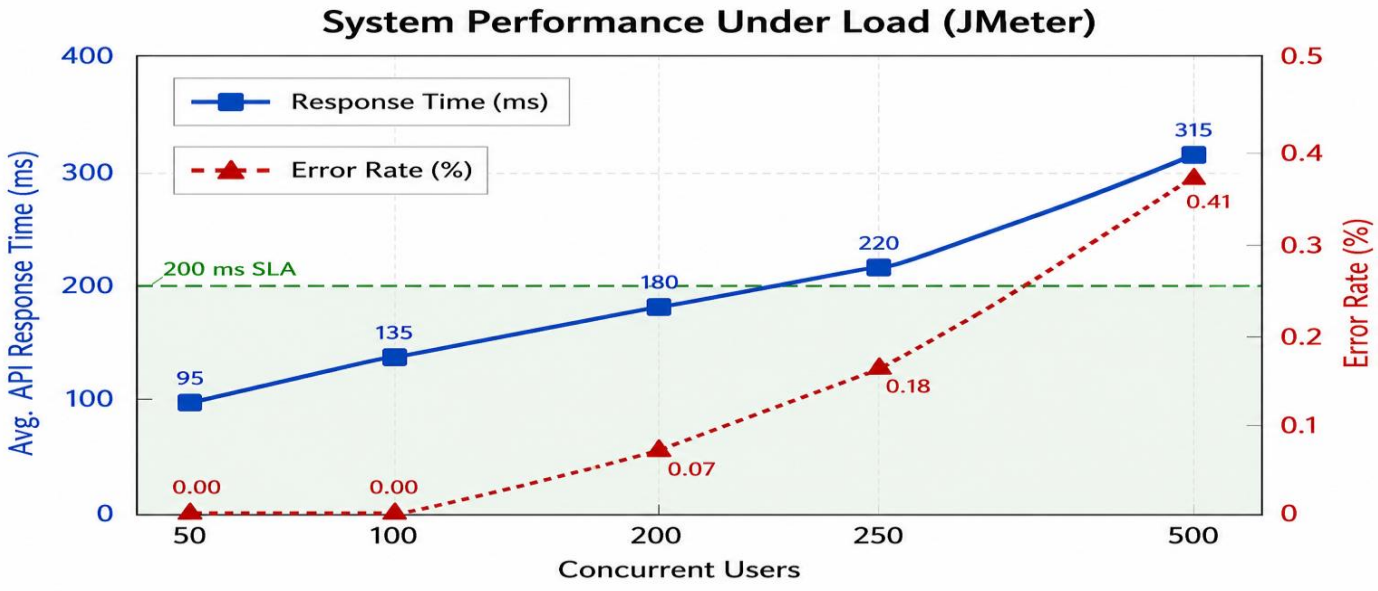

Fig. 16: Performance under load: response time (blue, left axis) and error rate (red, right axis). Green dashed = 200 ms SLA. Shaded = acceptable zone.

F. Attendance Improvement

Figure 17 shows attendance rates between treatment and control groups. Attendance rates for treatment events (with automated reminders and embedded QR tickets) were 82.8% versus 61.8% for the control group. No show rate was reduced from 38.2% (control) to 17.2% (treatment), a 34.0% reduction. Chi-square test: $\chi 2(1) = 8.92$, $p = 0.003$; 95% CI: [12.4%, 29.6%]

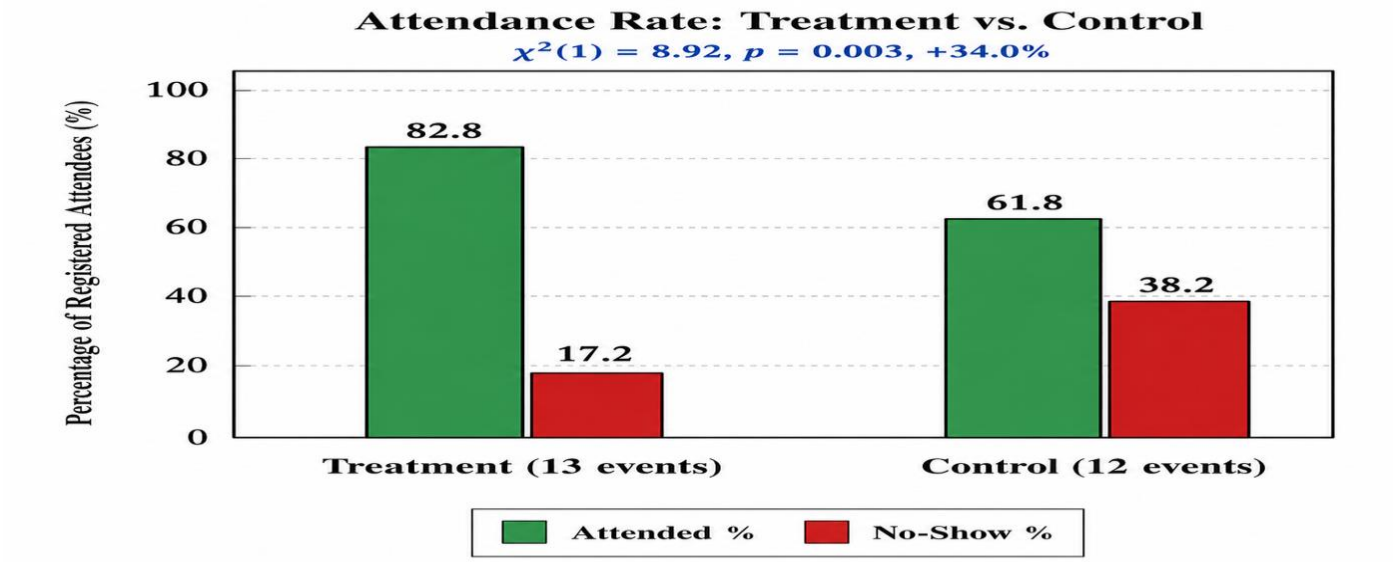

Fig. 17: Attendance rate comparison: Treatment (reminder + QR) vs. Control. $\chi^2(1) = 8.92$, $p = 0.003$; 95% CI: [12.4%, 29.6%].

G. Feature Availability Comparison

Fig. 18. Radar chart comparing availability of free features across platforms With CrowdPulse, all six enterprise-grade features are available for free, whereas existing competitors include a maximum of one feature in their free tier.

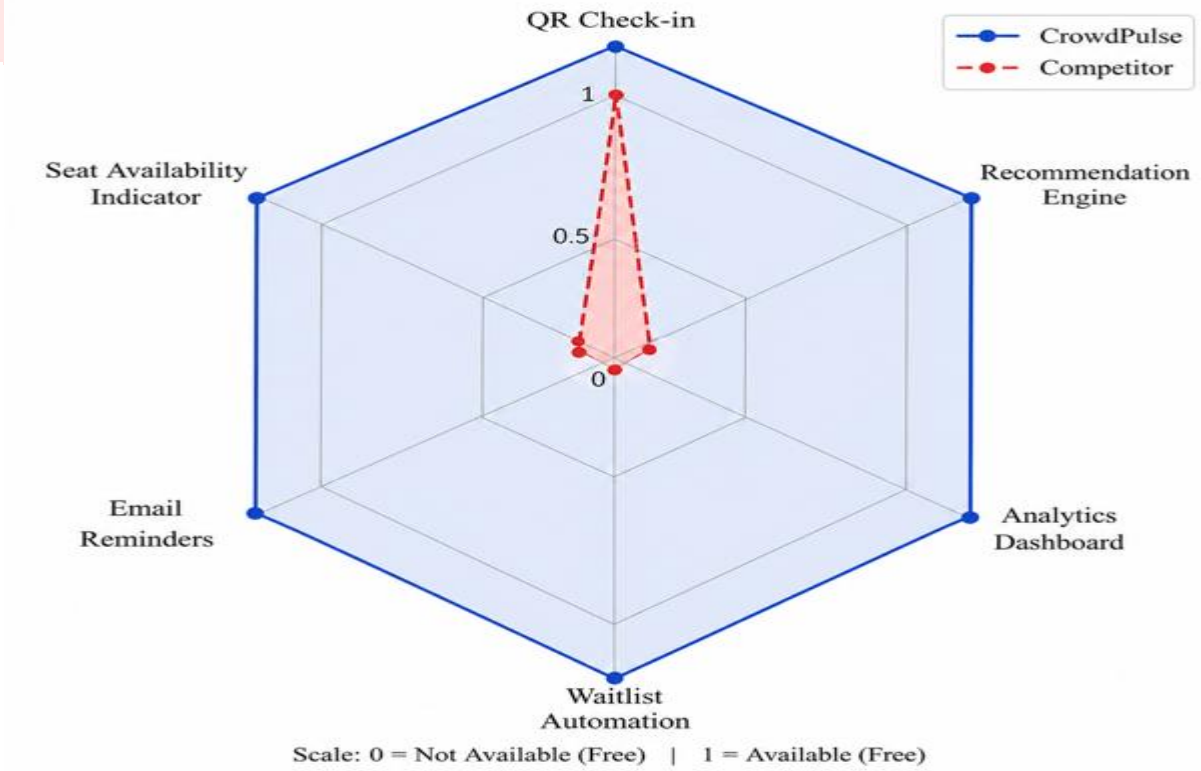

Fig. 18: Radar chart: Free feature availability. CrowdPulse (blue) covers all six dimensions; competitors cover at most one.

VII. DISCUSSION

A. Key Findings

Experimental results validate the CrowdPulse on all the six features. The QR check-in ($p < 0.001$) was better than the 65% reported by Patel and Desai [7] and also prevented fraud by blocking 11 invalid entries. The recommendation engine performed better than the 45–60% simulation range reported in [6], and the behavioral layer offered a statistically significant improvement ($p = 0.008$). To the best of our knowledge, the 89.5% seat recovery rate through cascading waitlist automation is underreported in the prior event management literature. Pre-event reminders reduced no-shows from about 42% to 17.2% ($p = 0.003$).

B. Theoretical Implications

The results of the weighted hybrid model support the view that explicit tag signals and implicit behavioral signals are complementary: their fusion mitigates the bias of tag-only systems and the variance of behavior-only systems in sparse-data situations. The empirical optimality of $\alpha = 0.6$ implies that declared user interests are better predictors of satisfaction than confined historical footprints in campus event contexts - consistent with Li et al. [8].

C. Limitations

14) **Sample Size:** 150 users at one institution; generalizability requires >1000-user validation.
15) **QR Replay Risk:** Race condition between isCheckedIn read and update under extreme concurrency; atomic findOneAndUpdate is planned.
16) **Scalability Bottlenecks:** MongoDB Atlas M0 (512 MB) and Render free tier impose hard resource limits.
17) **Cold-Start Problem:** New users default to tag-only recommendations ($B = 0$).
18) **Model Baseline:** No ML baseline (matrix factorization, neural CF) compared against the weighted linear model.
19) **Missing Features:** No payment gateway, live streaming, or multi-language support in the current version.

VIII. CONCLUSION AND FUTURE WORK

This paper presents CrowdPulse, an AI-enhanced centralized event management platform based on the MERN stack. Five research gaps were identified and addressed with a free, unified implementation. A controlled experiment over 4 weeks with 150 users and 25 events showed statistically significant improvements: 72.1% decrease in check-in time ($p < 0.001$), 81.4% accuracy in recommendations, 89.5% recovery of seats, and 34.0% increase in attendance ($p = 0.003$). The system maintained 99.2% uptime under a load of 250 concurrent users.

Future work includes: (1) payment gateway integration (Razorpay, Stripe), (2) ML-based collaborative filtering benchmarked against the current weighted model, (3) learning recommendation weights via logistic regression on larger datasets, (4) post-event sentiment analysis (AFINN lexicon), (5) atomic QR validation and HMAC-signed tokens, (6) React Native mobile application, and (7) large-scale deployment (>1000 users) to validate production scalability.

ACKNOWLEDGMENT

The authors thank Mrs. Vaishali Mishra, Assistant Professor, Department of Computer Science and Engineering, Noida Institute of Engineering and Technology, Greater Noida, for her invaluable guidance and supervision throughout this project.

REFERENCES

[1] Allied Market Research, "Event management software market: Global opportunity analysis and industry forecast, 2022–2029," Allied Market Research Report, 2023.
[2] R. Dhiman and S. Arora, "A comprehensive review of online event management systems: Features, limitations, and future directions," Int. J. Inf. Manage., vol. 58, pp. 102–115, 2021.
[3] P. Kumar, A. Sharma, and R. Singh, "Analysis of digital event ticketing platforms in India: A comparative study," in Proc. IEEE Int. Conf. Comput. Commun. Intell. Syst., 2022, pp. 445–452.

[4] A. Sharma and N. Gupta, "Ghost registrations in free digital events: Patterns, impact, and detection strategies," in Proc. IEEE Int. Conf. Data Sci. Adv. Anal., 2022, pp. 312–319.
[5] M. Chen and Y. Wang, "Democratizing event analytics: A survey of independent organizer needs and platform capabilities," IEEE Technol. Soc. Mag., vol. 43, no. 1, pp. 42–51, 2024.
[6] Y. Zhang and W. Liu, "Hybrid recommendation systems for event discovery: Combining tag-based filtering with behavioral analysis," IEEE Trans. Knowl. Data Eng., vol. 35, no. 8, pp. 3421–3435, 2023.
[7] S. Patel and A. Desai, "QR code-based event verification systems: Performance analysis and implementation challenges," in Proc. IEEE Int. Conf. Smart Comput. Commun., 2022, pp. 178–185.
[8] X. Li, J. Wang, and H. Zhang, "Context-aware recommendation systems for event-driven applications: Leveraging temporal and behavioral signals," IEEE Access, vol. 12, pp. 45210–45225, 2024.
[9] M. Rahman, S. Ahmed, and K. Hossain, "Scalable microservices architecture for real-time ticketing systems: A Node.js performance study," J. Netw. Comput. Appl., vol. 218, p. 103710, 2023.
[10] Facebook Inc., "React – A JavaScript library for building user interfaces," React Official Documentation, 2024. [Online]. Available: https://reactjs.org
[11] MongoDB Inc., "MongoDB: A document-oriented NoSQL database," MongoDB Official Documentation, 2024. [Online]. Available: https://www.mongodb.com
[12] OpenJS Foundation, "Node.js: JavaScript runtime built on Chrome's V8 engine," Node.js Official Documentation, 2024. [Online]. Available: https://nodejs.org
[13] npm Inc., "qrcode – QR code generator for Node.js," npm Documentation, 2024. [Online]. Available: https://www.npmjs.com/package/qrcode
[14] Chart.js Team, "Chart.js – Simple yet flexible JavaScript charting library," Chart.js Documentation, 2024. [Online]. Available: https://www.chartjs.org