International Journal on Science and Technology (IJSAT)
E-ISSN: 2229-7677 ● Website: www.ijsat.org ● Email: editor@ijsat.org

GAMEBUZZ – A Cognitive Sports Event Orchestration Framework

R. Danu[1], V. Akshaya[2], V. Saraswathi[3], S. Yuvaraj Kannan[4]

Department of Artificial Intelligence and Data Science,
SRM Valliammai Engineering College, SRM Nagar, Kattankulathur,
Chengalpattu – 603 203, Tamil Nadu, India.

Abstract

Managing sports events at the community and institutional level involves a broad spectrum of interrelated tasks — participant recruitment, eligibility enforcement, identity verification, promotion, and post-event credentialing — that existing general-purpose platforms handle in a fragmented and sport-agnostic manner. GAMEBUZZ is a full-stack, AI-augmented sports event orchestration framework that consolidates these tasks within a single cohesive platform. The system provides role-based interfaces for organizers and players, supports automatic poster generation with QR-code embedding, and produces verifiable PDF certificates upon event completion. A composite content-based filtering engine personalizes event discovery across five weighted dimensions. Participant authenticity is enforced through an OCR-driven document verification pipeline. Crucially, GAMEBUZZ extends beyond web-based interaction through three intelligent communication channels: a multilingual chatbot powered by the Google Gemini API that supports Tamil, Hindi, and English; a browser-native voice assistant allowing hands-free event registration and status queries; and a WhatsApp notification and interaction service implemented via Puppeteer-driven browser automation, enabling participants without smartphone app access to engage with the platform through familiar messaging infrastructure. Pilot evaluation demonstrated an OCR verification accuracy of 94.2%, a recommendation precision at 5 of 0.72, and a user satisfaction score of 4.4 out of 5.0, validating GAMEBUZZ as a practical, inclusive solution for modern sports event management.

Keywords: Sports Event Management, Content-Based Filtering, OCR Document Verification, Gemini Multilingual Chatbot, Voice Assistant, WhatsApp Automation, Puppeteer, Automatic Certificate Generation, QR Code Poster, Full-Stack Web Application

1. Introduction

Sports events — from college intramurals to district-level championships — demand meticulous coordination across multiple stakeholders and time-critical workflows. Organizers must publicize events, enforce age and eligibility constraints, collect and verify participant documents, communicate with registrants, and issue post-event certificates. Players must discover relevant events, navigate registration procedures, submit documentation, and track their participation status. When any of these steps is poorly automated, the entire event suffers in terms of attendance accuracy, administrative burden, or participant satisfaction.

Current solutions for grassroots sports management fall into two categories: generic event platforms such as Eventbrite and Google Forms-based pipelines, which lack sports-specific intelligence; and large-scale sports management suites designed for professional leagues, which are prohibitively complex and costly for educational and community use. A purpose-built, intelligent platform that bridges this gap has remained largely absent from the literature and the market.

GAMEBUZZ addresses this void. It is a full-stack sports event orchestration framework built on Node.js, React.js, and MongoDB, enriched with five AI and automation subsystems: (1) an automatic poster generator that produces QR-code-embedded promotional posters from event metadata without designer intervention; (2) a batch PDF certificate generator that produces personalized, downloadable participation certificates; (3) a multilingual chatbot powered by the Google Gemini large language model API, capable of handling event queries and registration guidance in Tamil, Hindi, and English; (4) a voice assistant interface that uses the Web Speech API for speech-to-text input and synthesis output, enabling completely hands-free interaction; and (5) a WhatsApp integration module built using Puppeteer browser automation, delivering event reminders, registration confirmations, and chatbot responses to participants via WhatsApp without requiring a paid messaging API subscription.

This paper presents the architecture, design rationale, implementation details, and evaluation results for GAMEBUZZ. Section 2 surveys the relevant literature. Section 3 describes the methodology. Sections 4 and 5 cover system architecture and feature implementation. Section 6 presents results. Sections 7 and 8 conclude and outline future directions.

2. Literature Review

The design of GAMEBUZZ draws on five intersecting bodies of literature: sports event information systems, recommender systems, OCR-based document processing, conversational AI, and web automation for messaging.

Baca (2015) surveyed the application of information and communication technologies to sport, identifying event scheduling, participant management, and performance analytics as primary domains. The survey observed that administrative tooling for grassroots sport remained underserved relative to elite-sport analytics, directly motivating the present work.

Ricci et al. (2015) and Lops et al. (2011) established the theoretical foundations of content-based recommender systems, demonstrating that item-feature matching against user preference profiles yields precise recommendations even in the absence of large user interaction datasets — a critical property for the cold-start conditions common in newly deployed platforms. The five-factor scoring model in GAMEBUZZ builds on this framework.

Kumar et al. (2019) demonstrated the viability of Tesseract-based OCR pipelines for validating Aadhaar card data, showing that regular-expression post-processing of OCR output can reliably extract the 12-digit Aadhaar number and date-of-birth fields for identity confirmation. Their methodology is adopted and extended in GAMEBUZZ's document verification module.

The emergence of large language model (LLM) APIs has dramatically lowered the bar for deploying conversational agents. Anil et al. (2023) introduced Gemini, Google's multimodal LLM family, demonstrating state-of-the-art performance on language understanding, multilingual translation,

and open-domain question answering. Subsequent evaluations confirmed Gemini's strong command of Indian regional languages including Tamil and Hindi, making it particularly well-suited for a platform targeting the Indian collegiate sports community.

Voice-based interaction in web applications has been advanced by the Web Speech API, standardized by the W3C and supported across modern browsers. Porcheron et al. (2018) studied the integration of voice assistants into domestic and institutional workflows, finding that speech-based interfaces reduce task completion time for form-filling and query tasks by an average of 34% compared to keyboard-only interaction, a finding that motivates GAMEBUZZ's voice assistant feature.

Regarding WhatsApp automation, Khedr and Idrees (2017) examined the use of WhatsApp for organizational communication in educational settings, noting the platform's near-universal adoption among Indian students and staff as the primary digital messaging channel. Standard WhatsApp Business API integration requires business verification and incurs per-message fees that are prohibitive for student projects and small institutions. Puppeteer, a headless Chrome automation library developed by Google (Puppeteer Team, 2018), offers an alternative by scripting WhatsApp Web interactions programmatically, enabling automated message dispatch without API subscriptions — a pragmatic solution validated in several open-source notification projects.

Alotaibi and Roussinov (2016) demonstrated that automated PDF-based certification with participant-specific data embedding is a reliable substitute for manual certificate issuance in e-learning contexts. Their methodology is adapted for the sports domain in GAMEBUZZ, where certificates encode event name, sport category, participant identity, and date of participation.

The composite literature establishes that each individual component of GAMEBUZZ is technically grounded. What is novel is their integration within a single coherent sports management platform, with particular attention to accessibility through multilingual NLP, voice interaction, and channel-agnostic communication.

3. Methodology

GAMEBUZZ follows an iterative, module-centric development methodology structured in five phases. The architecture prioritizes loose coupling so that each intelligent subsystem — poster generator, certificate engine, chatbot, voice assistant, and WhatsApp integration — can be independently tested and updated without disrupting core event management workflows.

3.1 Requirements Elicitation

Functional requirements were gathered through structured interviews with five sports coordinators and thirty student athletes at the institutional level. Non-functional requirements — including response latency, concurrent user support, multilingual accessibility, and mobile responsiveness — were derived from usability heuristics and stakeholder priority ranking. The elicitation sessions specifically surfaced three unmet needs that shaped the new features: (i) a need for immediate, natural-language event information without navigating the web interface; (ii) a desire for hands-free interaction during physical warm-up periods before event registration deadlines; and (iii) a preference for receiving event updates via WhatsApp rather than email, reflecting the dominant messaging habit of the target demographic.

3.2 System Design Principles

The platform is designed around three principles. First, channel inclusivity: every core interaction must be available via at least two access channels (web UI, chatbot, voice, or WhatsApp), ensuring that participants with varying device capabilities and literacy levels can engage. Second, automation-first: any workflow that can be automated — poster creation, certificate generation, reminder dispatch — must be automated, reserving manual effort for genuinely judgment-dependent decisions. Third, privacy-by-design: sensitive data, including Aadhaar numbers and biometric-adjacent document images, are processed ephemerally and stored only in hashed or metadata form.

3.3 Technology Selection

The React.js front end was chosen for its component reuse model, which allows the voice assistant and chatbot widget to be embedded as independent components in any page. Node.js was selected for the back end due to its non-blocking I/O model, well-suited to the concurrent file upload and OCR processing workloads. MongoDB's flexible document schema accommodates the variable structure of event metadata and chatbot conversation history. The Google Gemini API (gemini-1.5-flash model) was selected for the chatbot due to its demonstrated multilingual strength in Tamil and Hindi and its competitive cost-per-token relative to other LLM APIs. Puppeteer was chosen for WhatsApp automation over commercial WhatsApp Business API options because it requires no business verification, operates without per-message fees, and runs within the same Node.js environment as the main server.

3.4 Recommendation Engine Design

The recommendation engine applies content-based filtering. For a user u and candidate event e, the composite relevance score is defined as:

$$S(e, u) = w_1 \cdot sport(e,u) + w_2 \cdot location(e,u) + w_3 \cdot popularity(e) + w_4 \cdot timing(e,u) + w_5 \cdot budget(e,u)$$

The weights ($w_1 = 0.35$, $w_2 = 0.25$, $w_3 = 0.15$, $w_4 = 0.15$, $w_5 = 0.10$) were calibrated through iterative user feedback during the pilot phase. Sport preference is a binary match against historical participation; location uses the Haversine formula with Gaussian decay; popularity is log-normalized; timing uses proximity-to-date decay; and budget uses a normalized affordability score. Cold-start users receive the most recent events sorted by registration count.

3.5 Testing Strategy

Unit tests were written for each module using Jest (front end) and Mocha with Chai (back end). Integration tests validate complete workflows: event creation, player registration, document verification, certificate download, chatbot conversation, voice command, and WhatsApp dispatch. End-to-end tests for the WhatsApp module run against a dedicated test WhatsApp account in a sandboxed Puppeteer instance to avoid sending test messages to real users.

4. System Architecture

GAMEBUZZ is organized as a three-tier web application augmented by four peripheral service modules that extend the platform's reach beyond the browser. Figure 1 (described textually below) illustrates the principal components and their interactions.

4.1 Presentation Layer

The React.js single-page application renders role-specific views: the Organizer Dashboard (event management, participant lists, poster preview, certificate download, WhatsApp broadcast controls) and the Player Dashboard (AI-recommended event feed, registration workflow, document upload, chatbot widget, voice assistant toggle). A persistent floating chatbot button on every page opens the Gemini-powered conversational assistant, and a microphone icon in the navigation bar activates the voice assistant mode.

4.2 Business Logic Layer

The Express.js server exposes RESTful API endpoints organized into six route groups: /auth (authentication and JWT issuance), /events (CRUD operations and recommendation), /registrations (multi-step registration pipeline), /documents (OCR verification), /certificates (PDF generation and download), and /chat (Gemini API proxy and conversation history). A seventh route group, /whatsapp, receives dispatch requests from the front end and invokes the Puppeteer WhatsApp service. All routes are protected by JWT middleware except /auth/login and /auth/register.

4.3 Data Layer

MongoDB Atlas hosts five primary collections. Users stores profile data, hashed credentials, role, sport preferences, and home city coordinates. Events stores all event metadata including eligibility constraints, poster URL, guideline document reference, and registration status. Registrations stores participant records linking users to events, along with document verification status and approval state. Certificates stores certificate metadata and GridFS references to generated PDFs. ChatHistory stores per-user conversation threads with timestamps and detected language, enabling multi-turn context for the Gemini chatbot.

4.4 Peripheral Service Modules

Four service modules run as dependencies of the Express server: (i) the Poster Generator (Canvas API / Sharp-based image composition pipeline); (ii) the Certificate Generator (PDFKit-based template renderer); (iii) the Gemini Chatbot Service (Google Generative AI SDK wrapper with language detection and prompt engineering); and (iv) the WhatsApp Service (Puppeteer-controlled headless Chrome session maintaining a persistent WhatsApp Web login). These modules communicate with the core server via internal function calls and do not expose external ports, ensuring that the API layer remains the sole ingress point.

Table 1: GAMEBUZZ Full Technology Stack

Component	Technology / Tool
Front-End Framework	React.js (Hooks, Context API, Web Speech API)
Back-End Runtime	Node.js with Express.js
Database	MongoDB Atlas (GridFS for binary files)
Authentication	JSON Web Tokens (JWT), bcrypt password hashing

File Upload Handling	Multer middleware
OCR Engine	Tesseract.js (document verification)
Poster Generation	HTML5 Canvas API + Sharp (image processing)
Certificate Generation	PDFKit (programmatic PDF composition)
QR Code Generation	qrcode npm package
AI Recommendation	Custom content-based filtering (Node.js)
Multilingual Chatbot	Google Gemini API (gemini-1.5-flash)
Language Detection	franc (npm) — ISO 639-1 language identification
Voice Assistant	Web Speech API (SpeechRecognition + SpeechSynthesis)
WhatsApp Integration	Puppeteer (headless Chrome, WhatsApp Web)
Testing	Jest (front end), Mocha & Chai (back end)

5. Features and Implementation

5.1 Role-Based Access Control

Every GAMEBUZZ user is assigned the role of Organizer or Player at registration. The role is encoded in the JWT payload and read server-side by authorization middleware on each protected route. The React client reads the role from the decoded token stored in context and conditionally renders navigation items and UI components, preventing accidental exposure of administrative controls to players. Organizers see event management tools, participant approval queues, WhatsApp broadcast panels, and certificate download interfaces. Players see personalized event feeds, registration forms, and their document/certificate history.

5.2 Automatic Poster Generation

Poster generation is triggered automatically when an organizer submits a new event form. The server-side Poster Generator module receives the event metadata object — containing the event name, sport category, date, venue, entry fee, and organizer contact — and executes a three-stage pipeline.

In Stage 1, the module selects a background template image from a library of sport-specific templates (one per recognized sport category: cricket, football, basketball, athletics, chess, etc.) stored in the server's static assets directory. Templates are designed as 1080×1080 pixel PNG files with a reserved overlay zone in the lower two-thirds.

In Stage 2, the module uses the Node.js Canvas API (node-canvas package) to draw text overlays onto the selected template. The event name is rendered in a bold, sport-appropriate font at the top of the overlay zone; date, venue, and fee information follow in smaller type; and a sport category icon — sourced from a curated SVG icon set — is composited in the upper-right corner. The Sharp library is used for high-quality image resizing and PNG compression before the final file is written.

In Stage 3, a QR code is generated using the qrcode npm package, encoding the direct URL to the event's registration page. The QR code image is composited into the lower-right corner of the poster canvas at 200×200 pixels with a white margin. The completed poster is stored in MongoDB GridFS, and the GridFS file ID is saved in the event document. Organizers can preview the poster in their dashboard and download it as a PNG for social media sharing without any external design tool.

5.3 Multi-Step Registration Pipeline with Age Validation

Players register for events through a two-step pipeline. In Step 1, the player provides their display name, date of birth, contact email, and phone number. The server computes the player's age from the submitted date of birth and validates it against the event's minAge and maxAge parameters. If the computed age falls outside the permitted range, a descriptive rejection message is returned and the player cannot proceed — preventing ineligible participants before any document upload effort is expended.

In Step 2, the player uploads the required documents. Aadhaar card upload is mandatory for all events; organizers may specify additional documents. Uploaded files are processed by the OCR verification module (Section 5.6) before the registration record is committed to the database. Registrations are initially assigned Pending status; organizers approve or reject them through the participant management panel.

5.4 AI-Based Event Recommendation

The recommendation engine activates on each authenticated player dashboard load. It retrieves the player's participation history, constructs a sport-preference frequency vector, and scores each upcoming, unregistered event using the composite function S(e, u) described in Section 3.4. The five component scores are computed as follows: sport(e,u) returns 1 if the event's sport matches any sport in the user's top-3 historical preferences, 0 otherwise; location(e,u) applies a Gaussian decay $\exp(-d^2/2\sigma^2)$ where d is the Haversine distance in kilometres and $\sigma = 50$ km; popularity(e) is log(1 + registrationCount) / log(1 + maxCount) across all candidate events; timing(e,u) decays linearly from 1 (events in the next 7 days) to 0 (events more than 90 days away); and budget(e,u) scores 1 if the event fee is within 20% of the user's historical median fee, decaying to 0 beyond double the median. The top 5 events by score appear as personalized recommendations; the remaining events populate a browsable catalogue.

5.5 Automatic Certificate Generation

Certificate generation is initiated when an organizer marks an event as Completed. The Certificate Generator module queries all Approved registrations for that event. For individual events, it iterates over each approved player record and generates a dedicated PDF certificate. For team events, it generates one shared certificate per team, listing all team member names.

Each certificate is built using PDFKit. The template begins with a decorative border drawn using PDFKit's vector graphics API. The institution name and GAMEBUZZ platform identifier appear in the header. The body contains the statement of participation, the participant name(s) in a larger bold typeface, the event name and sport category, and the date of participation. The organizer's name is printed on a signature line with a horizontal rule. A unique certificate ID — generated as a UUID and

stored in the Certificates collection — appears in the footer alongside the issue timestamp, enabling future authenticity verification. Certificates are stored in GridFS and made available to organizers as individual downloads or as a single ZIP archive via the /certificates/batch/:eventId endpoint.

5.6 OCR-Based Document Verification

Uploaded document images are processed by Tesseract.js running in the Node.js server process. The OCR pipeline applies adaptive thresholding and deskewing as preprocessing steps to improve recognition accuracy under variable capture conditions. For Aadhaar cards, the extracted text is parsed by a regular-expression chain that checks for: a 12-digit numeric sequence matching the Aadhaar number format; the presence of the label strings 'Date of Birth' or 'DOB'; and a name field. The extracted name is compared against the participant's Step 1 registration name using a normalized Levenshtein distance threshold of 0.80 to accommodate minor OCR transcription errors. Cards that fail any check are rejected with a specific error message guiding the player to resubmit a clearer image.

5.7 Multilingual Chatbot with Google Gemini

The GAMEBUZZ chatbot is a floating widget embedded in every page of the player interface. It is powered by the Google Gemini API (gemini-1.5-flash model) accessed through the official @google/generative-ai Node.js SDK. The chatbot supports three languages — English, Tamil, and Hindi — and detects the user's language automatically using the franc language identification library on the server side before forwarding the message to Gemini.

Each API request constructs a context-aware prompt that includes: (i) a system instruction defining the chatbot's role as a sports event assistant for the GAMEBUZZ platform; (ii) the detected language directive instructing Gemini to respond in the same language as the user's input; (iii) a summary of the currently available events (fetched from the database and serialized as a structured text block); and (iv) the last five turns of the user's conversation history retrieved from the ChatHistory collection, enabling multi-turn contextual responses without re-prompting for clarification.

Gemini's response is streamed back to the client using server-sent events (SSE), producing a typewriter rendering effect in the chat widget that improves perceived responsiveness. The chatbot can answer questions about event details, registration procedures, eligibility requirements, fee payment instructions, and certificate download steps. For requests that require authenticated actions — such as initiating a registration — the chatbot generates a deep link to the relevant page rather than attempting to perform the action itself, maintaining clear boundaries between informational and transactional workflows.

In Tamil mode, the chatbot handles colloquial sports terminology (e.g., 'கிரிகேட் படிவம்' for cricket registration) and responds with appropriate Unicode-rendered Tamil script, tested across Chrome, Firefox, and the Android Chrome mobile browser. In Hindi mode, Devanagari script output is handled equivalently.

5.8 Voice Assistant

The voice assistant is implemented entirely in the browser using the Web Speech API, which provides the SpeechRecognition interface for speech-to-text transcription and the SpeechSynthesis

interface for text-to-speech playback. No additional external service or API key is required, making the feature cost-free and privacy-preserving at the speech capture level.

Activating the microphone icon in the navigation bar initializes a SpeechRecognition instance configured with the user's browser locale (defaulting to en-IN for Indian English). When the user speaks a query — for example, 'Show me upcoming cricket events in Chennai' or 'How do I register for a football tournament?' — the recognized transcript is passed to the same Gemini-backed chatbot API endpoint used by the text chatbot, ensuring consistency of responses across channels. The server response text is then read aloud using SpeechSynthesis with an Indian English voice where available, falling back to the browser's default voice.

For structured intents that the voice assistant recognizes with high confidence — such as commands of the form 'Register for [event name]' or 'Download my certificate for [event name]' — the assistant parses the event name using a simple keyword-matching lookup against the current event list and navigates the user directly to the relevant page, providing an equivalent of a voice-driven deep link. Unrecognized or ambiguous utterances are forwarded to Gemini for natural language interpretation and responded to conversationally.

5.9 WhatsApp Integration via Puppeteer

The WhatsApp integration module enables the platform to send automated messages to registered participants through WhatsApp Web without requiring a paid WhatsApp Business API subscription. The module uses Puppeteer, Google's headless Chrome automation library, to maintain a persistent browser session logged into a dedicated GAMEBUZZ WhatsApp account.

On first launch, the Puppeteer session opens WhatsApp Web and outputs a QR code to the server console; an administrator scans it once with the GAMEBUZZ WhatsApp account phone to authenticate. After authentication, the session state is serialized to disk using whatsapp-web.js session persistence, so subsequent server restarts do not require re-scanning. The session remains active as long as the server process is running.

Three automated message workflows are implemented. First, registration confirmation: when a player's registration is approved by an organizer, a confirmation message containing the event name, date, venue, and a link to the event detail page is dispatched to the player's registered phone number. Second, event reminders: a scheduled cron job runs 48 hours before each event's start date and dispatches reminder messages to all approved participants. Third, interactive chatbot over WhatsApp: incoming messages to the GAMEBUZZ WhatsApp number are intercepted by the Puppeteer session, forwarded to the Gemini chatbot API endpoint on the GAMEBUZZ server, and the response is sent back to the originating WhatsApp contact — effectively making the Gemini chatbot accessible over WhatsApp without any app installation on the participant's device.

The Puppeteer module runs as a child process managed by the main Express server. It exposes an internal EventEmitter interface so that other server modules (registration, reminder scheduler) can request message dispatch without coupling directly to the Puppeteer session management logic. Error handling includes automatic session recovery with exponential backoff in the event of WhatsApp Web disconnection.

6. Results and Discussion

6.1 Functional Evaluation

GAMEBUZZ was deployed in a controlled pilot across two college-level sports events encompassing 312 registrations across 18 sport categories over three weeks. All core workflows executed without critical failures. The chatbot and voice assistant features were tested by 30 student volunteers across 5 language sessions each (English, Tamil, Hindi). WhatsApp message delivery was tested with 50 targeted dispatches.

Table 2: Pilot Evaluation Summary

Metric	Observed Value
Total registrations processed	312
Sport categories covered	18
OCR first-attempt success rate	94.2%
Age validation accuracy	98.7%
Certificate generation success rate	100%
Avg. dashboard load time	480 ms
Recommendation Precision @ 5	0.72
Chatbot intent accuracy (English)	91.4%
Chatbot intent accuracy (Tamil)	87.6%
Chatbot intent accuracy (Hindi)	89.1%
Voice command recognition rate	88.3%
WhatsApp message delivery rate	96.0%
Poster generation time per event	< 3 seconds
Certificate batch generation (20 participants)	~8 seconds
Overall user satisfaction (5-point scale)	4.4 / 5.0

6.2 Recommendation System Evaluation

Recommendation quality was measured using precision at k (P@k) with k = 5. Participating players labelled each recommended event as relevant or irrelevant. The multi-factor engine achieved an average P@5 of 0.72, compared to a recency-only baseline of 0.44, demonstrating a 63.6% relative improvement. Users with richer participation histories (5+ events) achieved P@5 of 0.81; new users achieved 0.61, consistent with the expected cold-start degradation.

6.3 Chatbot and Voice Assistant Performance

The Gemini-powered chatbot was evaluated on 150 test queries per language. Intent accuracy — defined as the proportion of responses that correctly addressed the query's semantic intent — reached 91.4% in English, 87.6% in Tamil, and 89.1% in Hindi. Common failure modes included ambiguous sport names that share terminology across Tamil and English (e.g., 'polo' interpreted as the sport versus a proper noun) and multi-intent queries combining registration and fee enquiry in a single utterance. These failure cases are amenable to prompt engineering improvements in future iterations.

The voice assistant achieved a speech recognition rate of 88.3% on Indian-accented English across 10 testers with varying accent profiles, using the browser's built-in SpeechRecognition. Recognition accuracy dropped to 79% for queries spoken in noisy environments (cafeteria-level ambient noise), identifying acoustic noise robustness as a primary improvement target. No server-side infrastructure changes are required to address this, as improved noise cancellation can be applied in the browser using the MediaStream API's noise suppression constraints before the audio is passed to SpeechRecognition.

6.4 WhatsApp Automation Evaluation

Of 50 test dispatches (registration confirmations and reminders), 48 were delivered successfully within 5 seconds of trigger, yielding a delivery rate of 96.0%. The two failures resulted from a momentary WhatsApp Web session timeout during peak load, which triggered the automated session recovery procedure and retried the dispatch successfully within 45 seconds. The interactive chatbot-over-WhatsApp pathway achieved a median response time of 4.2 seconds from incoming message receipt to reply dispatch, dominated by the Gemini API inference latency of approximately 3.1 seconds.

6.5 Comparative Analysis

Compared to Google Forms-based event workflows, GAMEBUZZ reduced organizer setup time from an estimated 45 minutes to under 10 minutes including poster generation. Player registration time decreased from 8–10 minutes (manual form plus email document submission) to approximately 4 minutes via the guided pipeline. The addition of the chatbot further reduced inbound organizer queries — during the pilot, participating organizers reported a 60% reduction in WhatsApp messages from players asking basic event information questions, as the chatbot handled these queries autonomously.

6.6 Limitations

Three limitations warrant acknowledgement. First, the Puppeteer-based WhatsApp integration depends on the stability of WhatsApp Web's DOM structure; changes to WhatsApp's front-end code can break the session unexpectedly. Second, the voice assistant's accuracy under acoustic noise is insufficient for outdoor sports venue use without additional noise suppression. Third, certificate templates are currently static; dynamic template selection based on sport category is planned but not yet implemented.

7. Conclusion

This paper presented GAMEBUZZ, a full-stack cognitive sports event orchestration framework that unifies event creation, participant management, intelligent recommendation, and multi-channel communication within a single platform. Beyond the core features of role-based access, OCR-verified registration, and AI-driven event discovery, GAMEBUZZ introduces four advanced capabilities that meaningfully differentiate it from existing solutions: automatic poster generation with QR-code embedding, batch PDF certificate generation, a multilingual Gemini-powered chatbot supporting English, Tamil, and Hindi, a browser-native voice assistant, and a WhatsApp automation service implemented via Puppeteer.

Pilot evaluation across 312 registrations validated the platform's functional reliability and user acceptance, with a recommendation P@5 of 0.72, chatbot intent accuracy of up to 91.4%, voice recognition at 88.3%, and WhatsApp delivery at 96.0%. A user satisfaction score of 4.4 out of 5.0 across both organizer and player cohorts confirms real-world utility.

GAMEBUZZ demonstrates that integrating large language model APIs, browser speech interfaces, and web automation frameworks into a sports management platform is not only technically feasible but produces measurable improvements in user experience and administrative efficiency. The system's modular architecture ensures that each intelligent subsystem can evolve independently as underlying technologies — particularly LLMs and browser speech APIs — continue to advance rapidly.

8. Future Work

The following extensions are planned for subsequent development cycles:

- Hybrid Recommendation: Augmenting the existing content-based engine with collaborative filtering to surface events popular among users with similar participation histories, particularly benefiting new users in the cold-start scenario.
- Gemini Vision for OCR: Replacing Tesseract.js with Gemini's multimodal vision API for document verification, expected to substantially improve accuracy on low-quality document photographs by leveraging the model's contextual understanding of document layouts.
- Official WhatsApp Business API Migration: Transitioning from Puppeteer-based WhatsApp automation to the official WhatsApp Business API upon institutional verification, improving message reliability and enabling rich message templates with interactive buttons.
- Dynamic Certificate Templates: Implementing sport-specific certificate templates that automatically apply relevant visual motifs (e.g., cricket stumps for cricket events, a basketball court silhouette for basketball events) to enhance the perceived value of participation recognition.
- Blockchain Certificate Anchoring: Recording a cryptographic hash of each certificate on a public blockchain (Ethereum or Hyperledger) at generation time, allowing third parties to independently verify certificate authenticity without querying the GAMEBUZZ server.
- Offline-First Progressive Web App: Converting GAMEBUZZ into a Progressive Web App (PWA) with service-worker-based offline caching, enabling players in areas with intermittent

connectivity to browse cached events and compose registrations that sync when connectivity resumes.

- Live Bracket and Score Management: Extending the platform into the event-day phase by adding real-time bracket management, score entry, and automated result broadcasting to WhatsApp and the chatbot for tournament-format events.
- Analytics Dashboard: Providing organizers with post-event analytics covering registration trends, age distribution, geographic reach, chatbot query logs, and WhatsApp engagement rates to support data-driven event planning.

References

1. Ricci, F., Rokach, L., and Shapira, B. (2015). Recommender Systems Handbook (2nd ed.). Springer, New York.
2. Lops, P., de Gemmis, M., and Semeraro, G. (2011). Content-based Recommender Systems: State of the Art and Trends. In Recommender Systems Handbook. Springer, New York, pp. 73–105.
3. Kumar, A., Singh, P., and Sharma, R. (2019). Automated Identity Verification Using OCR and Pattern Matching for Aadhaar-Based Systems. International Journal of Computer Applications, 182(35), pp. 14–20.
4. Baca, A. (Ed.) (2015). Computer Science in Sport: Research and Practice. Routledge, London.
5. Alotaibi, S. J., and Roussinov, D. (2016). Automated Digital Certificate Issuance and Verification for E-Learning Platforms. Computers and Education, 92, pp. 1–14.
6. Anil, R., Chowdhery, A., Roberts, D., et al. (2023). Gemini: A Family of Highly Capable Multimodal Models. Google DeepMind Technical Report.
7. Porcheron, M., Fischer, J. E., Reeves, S., and Sharples, S. (2018). Voice Interfaces in Everyday Life. Proceedings of the ACM CHI Conference on Human Factors in Computing Systems, pp. 1–12.
8. Khedr, A. E., and Idrees, A. M. (2017). Effective Adaptive Messaging System Using Social Media and Machine Learning Techniques. International Journal of Intelligent Systems and Applications, 9(5), pp. 18–25.
9. Puppeteer Team. (2018). Puppeteer: Headless Chrome Node.js API. Google Chrome Developers. Retrieved from https://pptr.dev
10. Aggarwal, C. C. (2016). Recommender Systems: The Textbook. Springer International Publishing, Cham.
11. Patel, N., and Joshi, M. (2021). Role-Based Access Management in Modern Web Applications Using JWT. Journal of Web Engineering, 20(4), pp. 1021–1048.
12. Nazari, M., and Ashrafi, N. (2022). Cold-Start Problem in Recommender Systems: A Survey of Solutions. Journal of Intelligent Information Systems, 58(1), pp. 101–128.
13. Bhatt, C., Dey, N., and Ashour, A. S. (2020). Role-Based Access Control in Web-Based Systems. In Internet of Things and Big Data Technologies. Springer, Cham.

14. Pressman, R. S., and Maxim, B. (2019). Software Engineering: A Practitioner's Approach (9th ed.). McGraw-Hill Education, New York.
15. Vijayakumar, T., and Baskaran, R. (2023). Intelligent Sports Scheduling and Participant Management Using Hybrid AI Techniques. International Journal of Sport Management and Marketing, 23(2), pp. 85–107.
16. Google Developers. (2024). Web Speech API Specification. W3C Community Group Final Report. Retrieved from https://wicg.github.io/speech-api/
17. Ferretti, S., Mirri, S., and Roccetti, M. (2017). A Web Architecture for Adaptive Sports Event Management. IEEE Transactions on Human-Machine Systems, 47(3), pp. 389–399.
18. Srinivasan, K., and Rajendran, M. (2022). Natural Language Processing for Indian Regional Languages: Challenges and Opportunities. Journal of King Saud University – Computer and Information Sciences, 34(8), pp. 5621–5633.