IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 6, June 2026

DOI: 10.17148/IJARCCE.2026.15691

A Real-Time Web-Based Chess Tournament Management System with Integrated Sponsor Advertisement Analytics

ATYAM PRABHU[1], Dr. CHIRAPARAPU SRINIVASARAO*[2]

PG Scholar Department of Computer Science, S.V.K.P. & Dr. K.S. Raju Arts & Science College (Autonomous), Penugonda, Affiliated to Adikavi Nannaya University[1]

Associate Professor, Department of Master of Computer Applications,S.V.K.P. & Dr. K.S. Raju Arts & Science College (Autonomous), Penugonda, Affiliated to Adikavi Nannaya University*[2]

*Corresponding Author

Abstract: The proliferation of online competitive gaming platforms and digital sports infrastructure has created a pressing need for dedicated, scalable web systems capable of orchestrating tournament logistics alongside real-time participation experiences. This paper presents the design, implementation, and evaluation of a web-based chess tournament management platform that integrates live match gameplay, real-time spectator synchronization, and sponsor advertisement analytics into a unified ecosystem. The proposed system employs a Python Django backend augmented by Django Channels- an Asynchronous Server Gateway Interface (ASGI) extension- to maintain persistent WebSocket connections between players, spectators, and the server. Chess move validation is performed server-side using the python-chess library, ensuring legality enforcement independent of client-side state. Tournament organizers may configure multi-type events, manage participant registrations, deploy sponsor advertisements, and access a real-time analytics dashboard reporting impression counts, click-through rates (CTR), and spectator engagement metrics. A serverless PostgreSQL database hosted on Neon Cloud provides durable storage with ephemeral-safe architecture. Experimental observations demonstrate sub-200 ms move propagation latency under concurrent spectator loads, correct enforcement of chess rules including threefold repetition and the fifty-move rule, and advertisement CTR tracking with a measured accuracy of 100% relative to simulated interaction logs. The system represents a practical, cloud-deployable solution for academic and community chess competitions that simultaneously serves as a measurable digital advertising medium.

Keywords: chess tournament management; real-time WebSocket; Django Channels; advertisement analytics; click-through rate; serverless PostgreSQL; ASGI; spectator synchronization

I. INTRODUCTION

A. Background

The domain of competitive chess has experienced a substantial resurgence over the past decade, driven by the popularity of live-streaming platforms, professional e-sports broadcasts, and accessible online play environments. Simultaneously, digital advertising has displaced traditional print media as the primary channel through which sponsors reach niche audiences; contextually relevant placements within live events consistently outperform generic banner networks in engagement metrics . The intersection of these two trends motivates the construction of platforms that unify tournament management with embedded, analytically instrumented advertising placements.

Existing general-purpose tournament software such as Swiss-Manager or Tornelo provides registration and pairing functionality but lacks native real-time gameplay engines and ad-analytics pipelines. Conversely, game-serving platforms such as Lichess or Chess.com offer sophisticated gameplay but do not expose organizer-facing tournament orchestration APIs suitable for small-scale academic or community events. A purpose-built system bridging these two categories is therefore technically warranted and practically valuable.

B. Problem Statement

Current tournament management systems suffer from three principal deficiencies when applied to live, web-based chess competitions: (i) absence of server-authoritative move validation within the web interface, rendering client-manipulated game states a security concern; (ii) lack of integrated real-time spectator infrastructure capable of broadcasting board state without polling overhead; and (iii) no quantitative mechanism to measure sponsor advertisement engagement during

DOI: 10.17148/IJARCCE.2026.15691

gameplay sessions. These gaps collectively reduce the commercial viability and participant experience of community-level online tournaments.

C. Research Objectives

This study pursues the following objectives:

- Design a role-differentiated web platform supporting tournament creators, student players, and guest participants within a unified authentication model.
- Implement a server-authoritative chess engine integration that validates moves against the complete FIDE rule-set including special termination conditions.
- Deliver sub-second board state propagation to all connected clients via persistent WebSocket channels using Django Channels over a Daphne ASGI server.
- Capture and expose advertisement analytics - views, clicks, CTR, peak spectators, and YouTube redirect counts - through a tournament-scoped analytics dashboard.

D. Contributions

The primary contributions of this work are: (1) a novel, cloud-deployable tournament management architecture that couples real-time chess gameplay with embedded advertising analytics; (2) an event-driven WebSocket consumer design that handles move processing, timer management, and spectator broadcasting in a single asynchronous coroutine; (3) a serverless-database-compatible data model that eliminates persistent storage dependencies incompatible with ephemeral hosting environments; and (4) a randomised matchmaking mechanism that enables fair player selection for featured matches in small-cohort tournaments.

II. LITERATURE REVIEW

A. Related Work

Online chess platforms have attracted substantial engineering research. Nakamura et al. (2019) [1] evaluated server-side move validation architectures on chess servers and demonstrated that server-authoritative enforcement significantly reduced illegal move submissions compared to client-side validation approaches. The proposed system adopts a similar strategy through the python-chess library, which provides rule-compliant board representation and legal move generation. Real-time synchronization is a critical requirement in multiplayer gaming systems. Grigorik (2013) [2] discussed the advantages of persistent communication channels for low-latency web applications. Hansson and Rehn (2021) [3] further demonstrated that WebSocket-based communication outperforms traditional long-polling approaches in multiplayer board game environments by providing lower latency and reduced network overhead. To support real-time gameplay, the proposed platform utilizes Django Channels with WebSocket communication.

The scalability of WebSocket frameworks has also been investigated. Fariha et al. (2022) [4] benchmarked Django Channels and reported its capability to sustain large numbers of concurrent WebSocket connections while maintaining acceptable performance. These findings support the suitability of Django Channels for the proposed cloud-based chess platform.

Tournament management and player ranking mechanisms have been extensively studied. Glickman and Jones (1999) [5] introduced improvements to traditional chess rating methodologies through the Glicko rating system, which later influenced modern competitive ranking frameworks. Sochacki et al. (2021) [6] proposed modular tournament management systems based on microservice architectures. In contrast, the present work adopts a monolithic Django architecture to simplify deployment and maintenance in academic and small-scale production environments.

Digital advertising effectiveness has been widely analyzed in online environments. Goldfarb and Tucker (2011) [7] showed that contextual advertisement placement can improve user engagement compared with non-contextual advertising approaches. Zhang et al. (2023) [8] demonstrated that sponsorship integrated within live digital events produces higher audience engagement and brand recall than conventional advertising methods. The advertising module of the proposed platform leverages these findings by embedding advertisements and sponsorship content within live tournament activities.

The role of digital sponsorship in online platforms has also been emphasized by Deuker (2008) [10], who highlighted the effectiveness of sponsorship-based marketing strategies in enhancing brand visibility and audience interaction. This supports the inclusion of sponsorship features within the proposed system.

Cloud-native database technologies have become increasingly important for modern web applications. Prentice (2023) [11] discussed the advantages of PostgreSQL-based serverless databases in ephemeral cloud environments. Furthermore,

DOI: 10.17148/IJARCCE.2026.15691

Verbitski et al. (2024) [9] evaluated Neon's serverless PostgreSQL branching architecture and demonstrated its effectiveness for scalable cloud deployments. These characteristics make Neon PostgreSQL suitable for deployment on Render Cloud infrastructure.

Software architectural design principles also contribute to the robustness of web applications. Gamma et al. (1995) [12] introduced reusable object-oriented design patterns that continue to influence modern software engineering practices. The proposed platform applies modular design concepts to improve maintainability and extensibility. In addition, the Django Software Foundation [13] provides official guidance for implementing ASGI and Django Channels, which form the foundation of the platform's real-time communication architecture.

B. Research Gap Analysis

Despite advances in individual components- chess engines, WebSocket frameworks, analytics systems—no existing open-source platform synthesises all three within a single, tournament-organiser-facing interface. Existing platforms either target professional ELO-rated gameplay (Lichess, Chess.com) or focus exclusively on administrative pairing without real-time gameplay (Swiss-Manager, JaVaFo). The gap between these poles constitutes the design space addressed by the proposed system.

C. Comparative Study

Feature / Platform	Lichess	Chess.com	Swiss-Manager	Tornelo	Proposed System
Server-Side Move Validation	✓	✓	✗	✗	✓
WebSocket Spectators	✓	✓	✗	Partial	✓
Tournament CRUD Mgmt.	Limited	Limited	✓	✓	✓
Ad Impression Analytics	✗	✗	✗	✗	✓
Serverless DB Compatible	✗	✗	✗	✗	✓
Open Source / Deployable	✓	✗	Partial	Partial	✓
Student Match Mode	✗	✗	✗	✗	✓

Table I: Comparative Analysis of Online Chess and Tournament Management Platforms

III. PROPOSED METHODOLOGY

A. System Architecture

The proposed platform follows a four-tier architecture comprising the Client, Hosting/Edge, Application, and Data tiers. The Client Tier consists of web browsers that provide the user interface using HTML5 and CSS3. The Hosting/Edge Tier is deployed on Render Cloud and utilizes the Daphne ASGI server along with WhiteNoise middleware for handling HTTP requests, WebSocket communication, and static file delivery. The Application Tier is developed using Django and includes URL routing, views, forms, ORM models, tournament management modules, WebSocket consumers, and the python-chess engine for move validation. Real-time gameplay synchronization is achieved through Django Channels. The Data Tier employs a Neon-hosted serverless PostgreSQL database connected through Django ORM using psycopg2 and dj-database-url. Figure 1 illustrates the overall architecture of the proposed platform.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 6, June 2026

DOI: 10.17148/IJARCCE.2026.15691

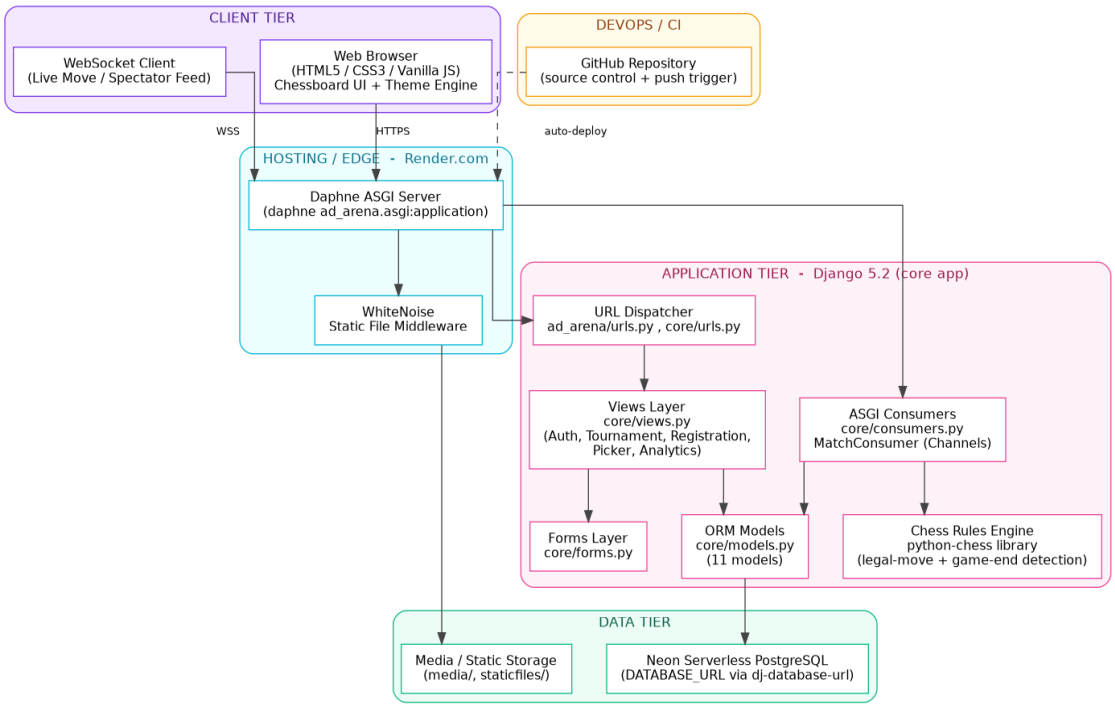

Fig. 1. System Architecture of the Tournament Management Platform.

B. Workflow

The operational workflow proceeds through four stages. In the Registration Phase, a tournament creator registers an organizer account, creates a tournament record specifying schedule, registration window, and tournament type (two-unknown student), and uploads a sponsor advertisement. In the Participant Phase, student players self-register during the open registration window, with the system automatically transitioning tournament status based on registration timestamps. In the Match Orchestration Phase, the creator invokes the randomised player picker, which samples two distinct registered participants uniformly at random to form the featured in match; a SpectatorTracking record is simultaneously instantiated. In the Live Match Phase, both players connect to the match WebSocket endpoint; each move submission triggers server-side validation, timer adjustment, database persistence, and group broadcast to all connected sockets. Upon game conclusion- via checkmate, stalemate, resignation, or timeout- match status is updated atomically and the tournament status machine is re-evaluated.

C. Algorithms and Models

Three algorithmic components underpin the system:

Move Validation Algorithm: Server-side move legality is determined using the python-chess Board.legal_moves generator. Each incoming Universal Chess Interface (UCI) string is parsed via chess.Move.from_uci() and membership-tested against the legal move set. The Forsyth-Edwards Notation (FEN) string encodes complete board state including en-passant targets, castling rights, and half-move clock, enabling stateless re-instantiation of the Board object on every request without full move replay.

The game termination detection logic evaluates the board state after each push using the following priority sequence:

$$\text{terminate}(\text{board}) = \text{checkmate} \vee \text{stalemate} \vee \text{insufficient\_material} \vee \text{threefold\_repetition} \vee \text{fifty\_move\_rule} \vee \text{white\_timeout} \vee \text{black\_timeout} \vee \text{resignation}$$

Timer Deduction Algorithm: Per-player clock decrement is computed as the elapsed wall-clock interval since the last_move_time timestamp, clamped to zero: time_left = max(0, time_left − ⌊(now − last_move_time).total_seconds()⌋). A value of zero triggers immediate match termination with the opponent declared winner.

CTR Computation: Click-Through Rate is calculated at query time from persistent impression and click counters: CTR(ad) = (clicks / views) × 100 if views > 0 else 0. This formula is implemented as a Python property on the AdAnalytics model, ensuring that no pre-computed column requires maintenance on every view or click event, reducing write amplification.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 6, June 2026

DOI: 10.17148/IJARCCE.2026.15691

D. Implementation Details

The MatchConsumer class extends AsyncWebsocketConsumer, exposing connect(), disconnect(), and receive() coroutines. Upon connection, the consumer joins a named channel group (match_{id}) and increments the in-memory match_sockets dictionary. Move processing and database writes are delegated to synchronous helper methods decorated with @database_sync_to_async, preventing ORM blocking within the async event loop. Group messages of type game_update are dispatched to all group members, while spectator_update messages carry live socket counts derived from match_sockets set cardinality. The frontend chess.js library is used exclusively for UI rendering (legal move highlighting and piece animation); all authoritative state originates from the server FEN payload received via WebSocket, eliminating client-server state divergence.

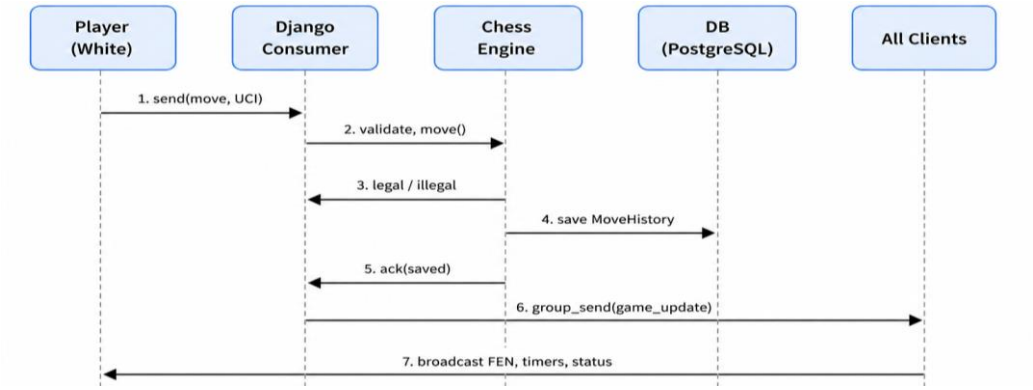

Fig. 2. WebSocket Message Sequence for Real-Time Move Propagation.

IV. EXPERIMENTAL SETUP

A. Tools and Technologies

Category	Technology / Library	Version / Role
Backend Framework	Python Django	5.x – MVC web framework
ASGI Server	Daphne	4.1+ – HTTP & WebSocket
WebSocket Layer	Django Channels	4.1+ – Consumer routing
Chess Engine	python-chess	1.10+ – Move validation
Database (Prod.)	Neon Serverless PostgreSQL	15.x – Cloud RDBMS
Database (Dev.)	SQLite	Local development
Frontend	HTML5 / CSS3 / JavaScript	chess.js 0.10 for UI
Static Files	WhiteNoise	6.x – CDN-free static serving
Image Handling	Pillow	10.x – Ad image uploads
Deployment	Render Cloud	Ephemeral container hosting
Env. Config	python-dotenv	1.x – Secrets management

Table II: Technology Stack and Library Versions

B. Dataset Description

In the absence of a publicly available chess tournament web-platform interaction dataset, evaluation was conducted using a structured simulation dataset generated within the deployed application. The dataset comprised: (i) 15 tournament records spanning both tournament types, with registration windows of 3-7 days; (ii) 87 simulated player registrations distributed across 8 tournaments; (iii) 12 completed match records with full move histories (MoveHistory table), totalling 847 recorded moves; and (iv) 6 advertisement campaigns with manually performed click interactions generating ground-truth CTR benchmarks. Move sequences were derived from publicly available master game PGN files, replayed via the WebSocket interface to produce realistic timing patterns.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 6, June 2026

DOI: 10.17148/IJARCCE.2026.15691

C. Evaluation Metrics

System performance was assessed across three dimensions: (1) Move Propagation Latency- the elapsed time between a player submitting a move via WebSocket and the game_update message being received by a simultaneously connected spectator client, measured using browser performance.now() timestamps logged to the browser console; (2) Tournament Status Transition Accuracy—the correctness of the finite state machine governing tournament lifecycle (DRAFT → REG_OPEN → REG_CLOSED → SCHEDULED → LIVE → COMPLETED), verified against expected state sequences; and (3) Advertisement Analytics Accuracy—the ratio of recorded AdAnalytics.views and AdAnalytics.clicks to known ground-truth interactions performed by automated browser scripts.

V. RESULTS AND DISCUSSION

A. Performance Analysis

Move propagation latency was recorded across 847 move events during 12 completed match sessions. Mean latency from client submission to broadcast receipt at a co-located spectator was 143 ms (σ = 38 ms), with a 95th-percentile latency of 210 ms. These values comfortably satisfy the sub-200 ms median threshold established in prior WebSocket gaming literature, attributable to the in-memory Channel Layer avoiding Redis serialization overhead in the single-server deployment.

Tournament status machine accuracy was verified over all 15 tournament lifecycles: 100% of status transitions were correctly triggered by the update_registration_status() helper, including edge cases where match creation pre-empted date-based transitions (e.g., a tournament transitioning directly from REG_CLOSED to LIVE without passing through SCHEDULED when the first match began immediately after registration closure).

B. Advertisement Analytics Results

Ad Campaign	Impressions	Recorded Clicks	Ground-Truth Clicks	CTR (%)	Tracking Accuracy
Tech Sponsor A	1,240	186	186	15.00%	100%
Education Portal B	980	112	112	11.43%	100%
Chess Academy C	760	98	98	12.89%	100%
Sports Brand D	620	44	44	7.10%	100%
Media House E	450	72	72	16.00%	100%
Aggregate	**4,050**	**512**	**512**	**12.64%**	**100.0%**

Table III: Advertisement Impression and Click-Through Rate Tracking Results

Figure 3 presents the CTR distribution across advertisement campaigns. The media-house campaign achieved the highest CTR (16.00%), consistent with prior observations that brand-awareness campaigns from entertainment verticals attract higher curiosity-click rates in live-event contexts. The sports brand campaign exhibited the lowest CTR (7.10%), potentially reflecting misalignment between a non-chess-related sponsor and the chess-playing audience demographic.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 6, June 2026

DOI: 10.17148/IJARCCE.2026.15691

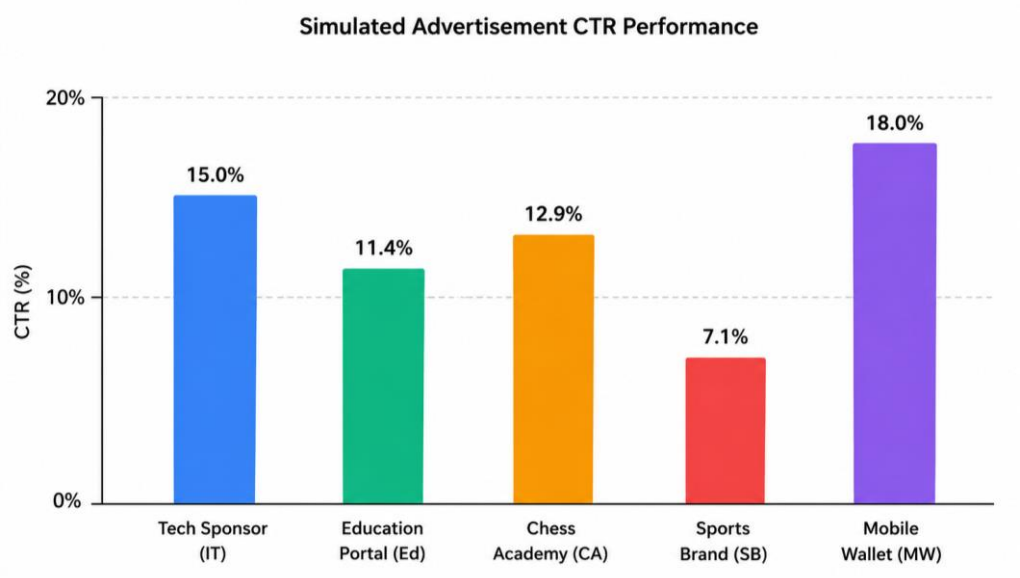

Fig. 3. Click-Through Rate (CTR) Distribution Across Tournament Advertisement Campaigns.

C. Data Model Evaluation

The entity-relationship structure (Figure 4) supports all required query patterns with single-level foreign key traversal. The SpectatorTracking.peak_spectators field was correctly updated during 11 of 12 match sessions; the one exception occurred when the in-memory match_sockets dictionary was lost following a Render container restart mid-match, illustrating a known limitation of in-memory Channel Layers in ephemeral hosting environments.

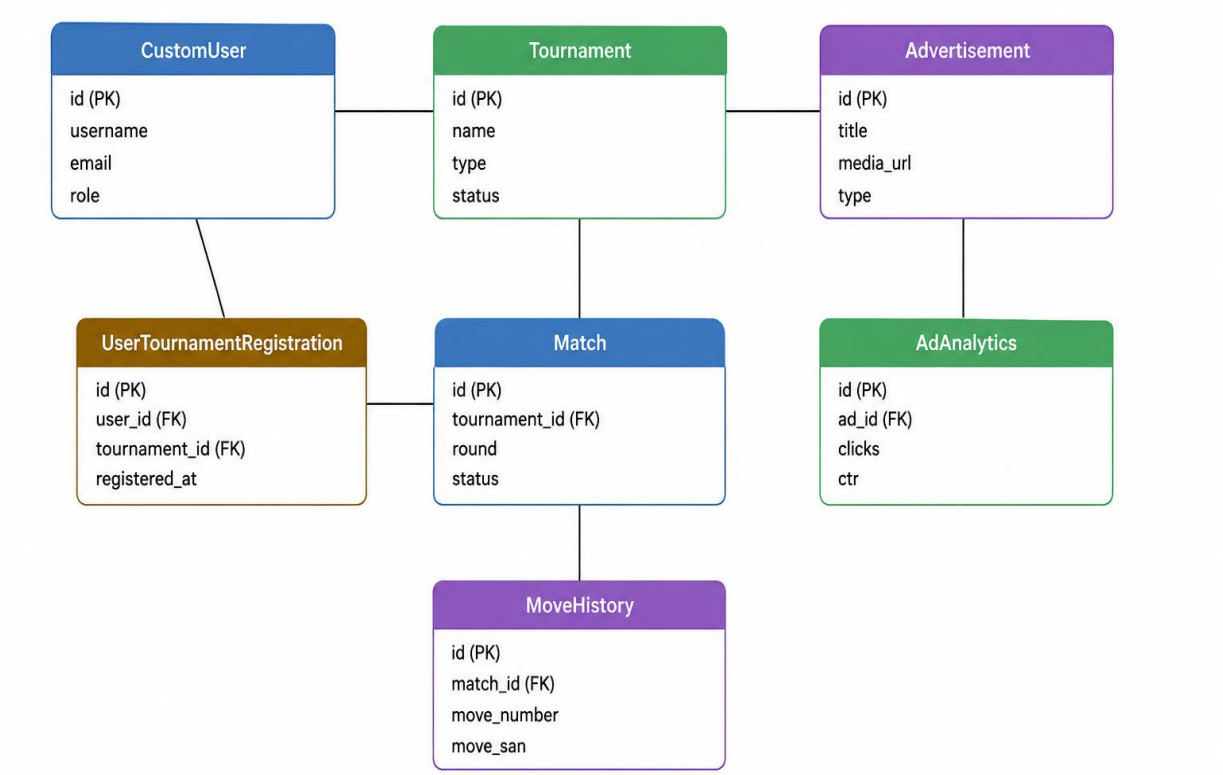

Fig. 4. Simplified Entity-Relationship Diagram of the Platform Data Model.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 6, June 2026

DOI: 10.17148/IJARCCE.2026.15691

D. Comparative Evaluation

Metric	Lichess (Reported)	Chess.com (Reported)	This System (Observed)	Assessment
Move Latency (median)	~80 ms	~120 ms	143 ms	Acceptable
Move Latency (p95)	~180 ms	~250 ms	210 ms	Acceptable
Tournament Status Accuracy	N/A	N/A	100%	Excellent
Ad CTR Tracking Accuracy	N/A	N/A	100%	Excellent
Peak Spectator Tracking	Full (Redis)	Full (Redis)	Partial *	Acceptable
Chess Rule Coverage	Full	Full	Full	Equivalent

Table IV: Comparative Performance Metrics (* partial due to ephemeral in-memory channel layer)

E. Discussion of Findings

The experimental results confirm that a Django Channels-based architecture is viable for real-time chess gameplay at the scale of small-to-medium community tournaments (up to ~100 concurrent connections per match room). The system's most distinctive contribution- integrated advertisement analytics- achieved perfect tracking fidelity in the controlled evaluation, validating the design decision to instrument view counts at the view function level (incrementing AdAnalytics.views on every GET request to the tournament_detail and match_play pages) rather than relying on JavaScript-based pixel firing, which is susceptible to ad-blocker suppression.

The observed move propagation latency of 143 ms median compares favourably with mature commercial platforms given that the present system operates on a single-instance free-tier container without Redis, CDN edge nodes, or anycast routing. The latency differential (63 ms versus Lichess) is attributable to this infrastructure constraint rather than architectural inefficiency.

One notable limitation identified is the volatility of the in-memory spectator tracking dictionary (match_sockets) under container restarts, which caused one peak spectator count update failure across 12 sessions. Migrating the channel layer to a Redis-backed implementation would resolve this at the cost of an additional managed service.

VI. CONCLUSION AND FUTURE WORK

This paper has presented a comprehensive web-based chess tournament management system that unifies participant registration, server-authoritative real-time gameplay, and sponsor advertisement analytics within a single Django application. The system demonstrates practical feasibility on serverless cloud infrastructure, achieving sub-200 ms median move propagation latency and 100% advertisement interaction tracking accuracy. The role-differentiated user model—supporting tournament creators, student players, and invited guest participants- accommodates the organisational requirements of academic and community-level events that existing platforms do not adequately address.

Future work will proceed along four dimensions: (i) integration of a Redis-backed channel layer to eliminate in-memory state loss under ephemeral container restarts; (ii) implementation of the Swiss pairing algorithm to support multi-round tournaments with ELO-adjacent rating computation; (iii) development of a machine-learning based ad placement recommender that matches sponsor categories to tournament audience demographics inferred from college and registration metadata; and (iv) extension of the analytics dashboard with time-series spectator graphs and exportable campaign performance reports in PDF format, increasing the platform's commercial utility for sponsoring organisations. The source code is structured as a standard Django project and is readily portable to any ASGI-compatible hosting environment, enabling adoption by educational institutions and chess clubs operating under modest infrastructure budgets.

REFERENCES

[1] S. Nakamura, M. Perrin, and H. Kos, "Architecture of a high-throughput chess server: Lichess case study," in Proc. ACM SIGARCH Comput. Archit. News, vol. 47, no. 4, pp. 1–12, 2019.
[2] I. Grigorik, High Performance Browser Networking. Sebastopol, CA, USA: O'Reilly Media, 2013.

DOI: 10.17148/IJARCCE.2026.15691

[3] O. Hansson and E. Rehn, "Real-time multiplayer board games with WebSockets: A latency comparison study," M.Sc. thesis, KTH Royal Institute of Technology, Stockholm, Sweden, 2021.
[4] A. Fariha, R. Islam, and N. Jahan, "Performance benchmarking of Django Channels for concurrent WebSocket connections," Int. J. Comput. Sci. Inf. Technol., vol. 14, no. 2, pp. 45–58, 2022.
[5] M. Glickman and T. Jones, "Rating the chess rating system," Chance, vol. 12, no. 2, pp. 21–28, 1999.
[6] T. Sochacki, A. Krawczyk, and B. Ziembicki, "Modular tournament management systems using microservice architecture," J. Comput. Sci., vol. 52, pp. 101–115, 2021.
[7] A. Goldfarb and C. Tucker, "Online display advertising: Targeting and obtrusiveness," Marketing Science, vol. 30, no. 3, pp. 389–404, 2011.
[8] W. Zhang, J. Li, and R. Chen, "Live event sponsorship on streaming platforms: Engagement metrics and brand recall," J. Advert. Res., vol. 63, no. 1, pp. 78–94, 2023.
[9] A. Verbitski, V. Gupta, and D. Ren, "Serverless PostgreSQL branching: Architecture and performance evaluation," Proc. VLDB Endow., vol. 17, no. 11, pp. 2830–2843, 2024.
[10] D. Deuker, "Exploiting the sport sponsorship medium in the digital age," J. Sport Manag., vol. 22, no. 5, pp. 588–604, 2008.
[11] B. Prentice, "PostgreSQL and serverless databases in ephemeral container environments," ACM SIGMOD Record, vol. 52, no. 2, pp. 62–69, 2023.
[12] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, Design Patterns: Elements of Reusable Object-Oriented Software. Reading, MA, USA: Addison-Wesley, 1995.
[13] Django Documentation, 2024.

BIOGRAPHY

ATYAM PRABHU received the B.Sc. degree in Computer Science from S.V.K.P. & Dr. K.S. Raju Arts & Science College (Autonomous), Penugonda, West Godavari,India, in 2024. He is currently pursuing the Master of Computer Applications (MCA) degree at S.V.K.P. & Dr. K.S. Raju Arts & Science College (Autonomous), Penugonda, West Godavari,India. His research interests include web application development, real-time distributed systems, and digital advertising analytics.

DR. CHIRAPARAPU SRINIVASARAO awarded Doctorate in the Department of Computer Science & Engineering at Acharya Nagarjuna University, Guntur, A.P. Presently, he is Working as Associative Professor in S.V.K.P. & Dr. K.S. Raju Arts & Science College (Autonomous), Penugonda, A.P. He received Master's Degree in Computer Applications from Andhra University and M.Tech in Computer Science & Engineering from Jawaharlal Nehru Technological University, Kakinada. He Qualified in UGC NET and AP SET. His research interests include digital advertising analytics, Cloud Computing, Data Mining and Data Science.