World Journal of Advanced Research and Reviews
eISSN: 2581-9615 CODEN (USA): WJARAI
Cross Ref DOI: 10.30574/wjarr
Journal homepage: https://wjarr.com/

(REVIEW ARTICLE)

Check for updates

DSA Battle Royale: A real -time competitive coding combat

Kavitha Soppari, Rajesh Gundu, Vinay Donthula and Shiva Ramavath *

Department of CSE (AI and ML) of ACE Engineering College, Hyderabad, India.

World Journal of Advanced Research and Reviews, 2026, 29(03), 1075-1080

Publication history: Received on 05 February 2026; revised on 14 March 2026; accepted on 16 March 2026

Article DOI: https://doi.org/10.30574/wjarr.2026.29.3.0650

Abstract

DSA Battle Royale :- Real-Time Competitive Coding Combat is a gamified platform designed to enhance problem-solving skills in Data Structures and Algorithms (DSA) through an interactive competitive environment. The system introduces innovative gameplay mechanics such as 1 vs 1 coding duels with health-based progression, group-based contests with room-based participation, and tiered difficulty levels (easy, medium, hard). A rating and leaderboard system ensures fair competition and motivates continuous improvement, while performance analytics provide personalized insights into coding strengths and weaknesses. By merging gamification principles with real-time coding evaluation, the platform not only encourages learners to practice coding regularly but also transforms traditional contests into an engaging and adaptive learning experience.

Keywords: Competitive Programming; Gamification; Real-Time Coding; Problem-Solving; Leaderboards; Contest Management; Rating System; DSA Practice

1. Introduction

Traditional competitive programming platforms primarily focus on problem archives, scheduled contests, and static leaderboards, which can feel abstract and demotivating for many students. Learners often struggle to maintain consistency because the feedback loop between solving problems and seeing tangible progression is slow and non-interactive. Recent real-time coding battle platforms and head-to-head challenge systems have demonstrated that live competition, time pressure, and visually rich feedback significantly increase motivation and practice frequency.

DSA Battle Royale addresses this gap by reimagining DSA practice as a real-time battle experience where each correct or incorrect submission directly influences in-game health bars, damage, and survival. Players can participate in 1 vs 1 duels or join invite-only group contests, with both modes mapped to an Elo-based rating system that reflects player skill over time. The platform is especially suited for use in college classrooms, coding clubs, and online communities, where instructors or hosts can create private contests, share access codes, and observe live leaderboards to monitor student performance. By tightly integrating game mechanics with algorithmic problem-solving, DSA Battle Royale aims to make rigorous DSA practice more immersive while preserving the seriousness of competitive programming evaluation.

2. Literature Survey

2.1. Mishra & Pokalwar (2023) – JediCode

Mishra and Pokalwar (2023) introduce JediCode, a gamified competitive coding platform that uses synchronized challenges, real-time leaderboards, and achievements to increase engagement and learning in programming. Implemented with ReactJS and NestJS, it supports dynamic UIs, instant scoreboard updates, and a scalable backend

* Corresponding author: Ramavath Shiva

handling authentication, APIs, and concurrent users. Gamification features such as badges, levels, rankings, and adaptive matchmaking significantly improve motivation, participation, and long-term retention compared with standard contest systems.

2.1.1. Methodologies and Algorithms

JediCode combines REST APIs and WebSockets to support live coding duels and immediate statistics updates. Adaptive matchmaking algorithms select opponents based on performance metrics, while gamification logic assigns experience points, ranks, and badges after each match. Encrypted communication and token-based authentication ensure security, illustrating how a gamified architecture can successfully merge learning and competition at scale.

2.2. Zhan et al. (2022) – Meta-Analysis on Gamification

Zhan et al. (2022) conduct a meta-analysis of 21 studies on gamification in programming education, assessing effects on motivation, engagement, and performance. They conclude that elements like points, badges, leaderboards, levels, and competitive challenges substantially enhance persistence, enjoyment, and cognitive understanding among programming students. Social competition and visible progress strengthen feelings of achievement, motivating learners to attempt more and harder problems.

2.2.1. Methodologies and Algorithms

The authors apply quantitative synthesis and effect-size calculations to variables such as engagement rate, task completion, and performance gains. Gamification components are grouped (points, badges, leaderboards, progress tracking, social competition), and meta-regression shows that dynamic feedback and competition have the strongest motivational effect. They recommend hybrid gamified frameworks combining real-time feedback, adaptive difficulty, and progression mechanics to optimize educational impact.

2.3. Pinto & Terroso (2022) – Gamified Programming Course

Pinto and Terroso (2022) implement gamification in an Algorithms and Data Structures course, using levels, challenges, and rewards to make programming more interactive. Students work in teams under time constraints, earning points and badges for solving coding tasks, which improves attendance, completion rates, and coding performance. Level-based progression and continuous feedback reduce anxiety, help beginners build confidence, and position programming as structured skill-building rather than a purely academic obligation.

2.4. Swacha & Szydłowska (2023) – FGPE Framework

Swacha and Szydłowska (2023) evaluate the Framework for Gamified Programming Education (FGPE), an open-source platform that adds points, badges, leaderboards, and levels to programming courses. FGPE offers modular, API-based integration with LMS systems, rule-based reward allocation, and analytics dashboards to monitor participation and performance. Results show higher motivation, persistence, and problem-solving accuracy for students using FGPE, while also indicating the need for more adaptive, real-time features in future gamified systems.

Table 1 Comparison Table of Literature Survey

S.No	Author(s)	Title	Methodology Used	Findings from the Reference Paper
1.	Mishra, A., & Pokalwar, S. (2023)	JediCode — A Gamified Approach to Competitive Coding	ReactJS (frontend), NestJS (backend), RESTful APIs, WebSocket communication, Adaptive Matchmaking	Developed a real-time gamified coding platform enhancing motivation and engagement. Results showed improved participation, retention, and user satisfaction through real-time duels and rewards.
	Zhan, Z., He, L., Tong, Y., Liang, X., Guo, S., & Lan, X. (2022)	The Effectiveness of Gamification in Programming	Quantitative Meta-Analysis, Statistical Modeling, Meta-Regression	Synthesized results from 21 studies; confirmed gamification significantly improves motivation, engagement, and learning outcomes. Leaderboards and feedback mechanisms increased

2.		Education: Evidence from a Meta-Analysis		persistence and problem-solving accuracy.
3.	Pinto, M., & Terroso, T. (2022)	Learning Computer Programming: A Gamified Approach	Problem-Based Learning (PBL), Real-Time Scoring Algorithms, Adaptive Difficulty Adjustment	Integrated gamification into programming courses, leading to higher motivation, teamwork, attendance, and reduced anxiety. Adaptive levels and feedback loops enhanced learning effectiveness and confidence.
4.	Swacha, J., & Szydłowska, J. (2023)	Does Gamification Make a Difference in Programming Education? Evaluating FGPE-Supported Learning Outcomes	FGPE Framework, API-Based Modular Architecture, Rule-Based Reward Algorithms, Analytics Dashboards	FGPE platform improved student motivation, engagement, and self-directed learning. However, static components limited adaptability; authors suggest enhancing real-time personalization and dynamic difficulty mechanisms.

3. System Architecture

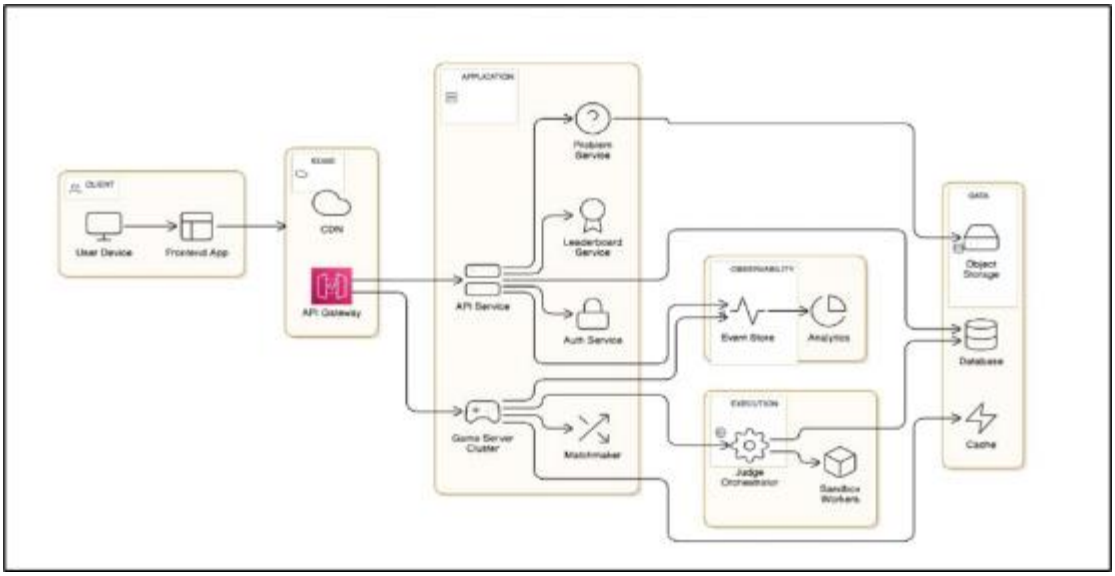

Figure 1 System Architecture

The architecture of DSA Battle Royale is organized into four primary layers: client interface, real-time game orchestration, secure judging subsystem, and persistence and analytics. The client interface is implemented using React with Tailwind CSS, presenting players with live battle views containing health bars, timers, problem statements, in-browser code editors, and dynamic animations for damage events, streaks, and rank changes. The frontend maintains a persistent WebSocket connection (via Socket.io) to receive low-latency updates about opponent actions, health adjustments, and leaderboard changes.

The real-time game orchestration layer is handled by a Node.js/Express backend combined with Socket.io namespaces and rooms for duels and contests. When players initiate a 1 vs 1 duel or join a contest room, the backend performs matchmaking, assigns both players the same problem set, and instantiates a match session that tracks health values, submission events, and timers. All code submissions are forwarded to a dedicated judge service rather than executed directly on the application servers. The judging subsystem is implemented as a sandboxed microservice, typically backed by Docker-based containers or an external judge API, which compiles and runs user code against predefined test cases under strict time and memory limits. The results (Accepted, Wrong Answer, Time Limit, Runtime Error) are returned to the orchestrator, which then updates health, scores, and match state.

Persistent data such as user profiles, ratings, match histories, and contest configurations are stored in a relational database like PostgreSQL, while Redis is used for caching hot data including live match states, queue information, and frequently accessed leaderboards. A background worker component periodically recalculates aggregates such as streaks, achievement unlocks, and rank transitions based on recent match outcomes. This architecture separates real-time event handling from persistent storage and resource-intensive judging, enabling scalable deployment on cloud platforms similar to existing competitive coding battle systems.

4. Proposed Methodology

4.1. Game Modes and Mechanics

DSA Battle Royale defines two primary game modes: 1 vs 1 duels and group contests. In 1 vs 1 duels, two matched players enter a battle arena where each begins with a fixed health value (for example, 100 HP), and both receive the same set of DSA problems drawn from difficulty-tagged pools. Correct submissions inflict damage on the opponent by reducing their health in discrete amounts calibrated to problem difficulty (for instance, larger damage for harder problems), while incorrect submissions cause a smaller self-inflicted penalty to discourage random guessing. If a player's health reaches zero before the timer expires, they lose the duel; otherwise, the system uses total solved problems, penalty counts, and remaining health as tie-breakers.

Group contests are created and hosted by a user who acts as the organizer, typically a teacher or community leader. The host configures contest parameters including duration, problem set composition across Easy, Medium, and Hard categories, and access controls such as room password or invite-only links. Participants join the contest room through these private credentials, and the system presents a shared problem set with a global leaderboard that updates in real time based on problems solved and submission times. Unlike duels, group contests do not use health mechanics; instead, they follow standard competitive programming scoring rules where faster correct solutions rank higher.

4.2. Rating, Ranks, and Progression

Player skill is quantified using an Elo-based rating system that updates after each duel or contest based on opponent strength and match outcomes. For every match between players A and B with ratings R_A and R_B, the system computes expected scores E_A and E_B using the standard logistic formula and then updates ratings using $R'_A = R_A + K(S_A - E_A)$, where S_A is 1 for a win, 0 for a loss, and 0.5 for a draw. Different K-values may be used for low-rated versus high-rated players to stabilize ratings for experienced participants. On top of continuous rating values, the platform defines discrete rank divisions such as Bronze, Silver, Gold, Platinum, Diamond, and Legend, which are derived from rating thresholds and displayed on profiles and leaderboards.

To further encourage regular play, DSA Battle Royale implements a progression system based on experience points (XP), levels, achievements, and streaks. Players gain XP for each match, with bonuses for winning against higher-rated opponents or performing well in contests. Achievements such as "First Win," "5-Win Streak," "Contest Champion," and "Survivalist" are unlocked by satisfying specific conditions, and win streaks are highlighted in the UI to reward consistent performance. A dedicated profile dashboard aggregates rating trajectories, rank history, win/loss records, recent problems solved, and contest participation, providing players with a clear view of their algorithmic development over time.

4.3. Secure Code Evaluation and Fair Play

Given that players submit arbitrary code, the system emphasizes security and fairness in the judging pipeline. All code is executed inside isolated containers or sandboxed environments with restricted CPU, memory, file system, and network access to prevent malicious behavior and resource abuse. Language runtimes are pre-installed in base images, and each submission runs against a fixed set of hidden test cases defined per problem. Time and memory limits are enforced, and results are communicated back to the main backend via authenticated APIs. To mitigate cheating, the platform monitors unusual patterns such as extremely fast identical submissions across accounts, repeated copy–paste behavior, or abnormal success rates, and can flag suspicious accounts for review.

5. Algorithmic Flow of a Duel Match

The core duel flow in DSA Battle Royale can be summarized as an algorithm that coordinates matchmaking, code evaluation, health updates, and rating adjustments:

- Receive duel request from Player A with selected difficulty range and queue preference.
- Match Player A with Player B based on Elo rating proximity and availability.
- Initialize match session with both players at full health and assign a shared problem set derived from the requested difficulty.
- Start match timer and broadcast initial state to both clients over Socket.io.
- When a player submits a solution, forward the code, language, and problem ID to the judge service.
- The judge executes the code in a sandbox, evaluates it against test cases, and returns the verdict (Accepted or specific error) plus performance metrics.
- If the verdict is Accepted, reduce the opponent's health by damage corresponding to problem difficulty and increment the solver's score; if incorrect, apply a small health penalty to the submitting player.
- Broadcast updated health values, scores, and any streak changes to both clients in real time.
- Check termination conditions after each event: a player's health reaches zero, the timer expires, or a maximum number of problems is solved.
- Determine winner using health, solved count, and penalties as tie-breaking criteria, then compute new Elo ratings for both players using the Elo update formula.
- Persist match results, rating changes, and XP progression to the database and invalidate or refresh relevant leaderboard caches.
- Notify players of the outcome, rating changes, and achievement unlocks through the UI.

This structured flow ensures consistent duel behavior, clear separation between judging and orchestration, and deterministic rating updates that remain interpretable to users familiar with Elo-based systems.

6. Conclusion

DSA Battle Royale presents a unified framework that merges the rigor of DSA problem-solving with the immediacy and excitement of real-time battle mechanics. By integrating 1 vs 1 duels, private group contests, Elo-based ratings, rank divisions, and achievements into a single platform, it addresses limitations of traditional competitive programming sites that rely mainly on static contests and offline practice. The proposed architecture—comprising a React-based live interface, a Node.js/Express real-time backend, a sandboxed judge service, and a PostgreSQL/Redis data layer—supports scalable and secure operation for classrooms and public communities alike.

Future work can explore AI-assisted difficulty recommendations, adaptive matchmaking that considers player learning curves, and richer analytics dashboards for educators to track student progress in specific DSA topics. Integration with existing coding platforms, expansion to team-based modes, and mobile-first interfaces could further increase accessibility and impact. By positioning competitive coding as a battle royale-style experience, DSA Battle Royale has the potential to significantly enhance motivation and sustained engagement in algorithmic learning.

Compliance with ethical standards

Disclosure of conflict of interest

The authors declare that there is no conflict of interest regarding the publication of this paper.

References

[1] AlgoArena Team, "AlgoArena: Real-Time 1v1 Competitive Coding Battles," AlgoArena Official Documentation and Platform Pages, 2025. Available: https://www.algoarena.net [Accessed: Mar. 2026].

[2] AlgoArena Community, "AlgoArena: Real-time 1v1 coding battles + 5000+ problem library (free beta)," r/leetcode, Reddit, Dec. 2025. [Online]. Available: https://www.reddit.com/r/leetcode/comments/1pc1mr6/algoarena_realtime_1v1_coding_battles_5000/.

[3] GeeksforGeeks, "Elo Rating Algorithm," GeeksforGeeks, 2018. [Online]. Available: https://www.geeksforgeeks.org/dsa/elo-rating-algorithm/.

[4] "Elo rating system," Wikipedia, 2002. [Online]. Available: https://en.wikipedia.org/wiki/Elo_rating_system.

[5] A. Aiera, "I Built a Real-Time 1v1 Coding Battle Platform in Days Using Kiro," DEV Community, Dec. 2025. [Online]. Available: https://dev.to/aieradev/i-built-a-real-time-1v1-coding-battle-platform-in-days-using-kiro-25de.

[6] CodeBattle Team, "CodeBattle: The Ultimate 1v1 Real-Time Coding Battle Platform," LinkedIn Article, Mar. 2025. [Online]. Available: https://www.linkedin.com/pulse/codebattle-ultimate-1v1-real-time-coding-battle-platform--k7kyf.

[7] D. Bisht, "Real-time 1v1 Coding Battle Platform Built with Next.js & Socket.io," LinkedIn Post, Dec. 2025. [Online]. Available: https://www.linkedin.com/posts/dipish-bisht-23755b291_building-a-live-coding-arena-real-time-activity-7408528487516508160-tiu5.

[8] AlgoArena Team, "AlgoArena: Beta Testers Wanted for Real-Time Coding Battles!," Announcement, Dec. 2025. [Online]. Available: http://tims.itf.gov.ng/aanews1/algoarena-beta-testers-wanted-for.