CodeArena - Functional Requirements

Target stack: **Angular (Frontend) + Spring Boot (Backend)**

Document scope

This document defines the functional requirements for the CodeArena application, organized into 8 modules that can be implemented in parallel by a team.

Product overview

CodeArena is a competitive programming web application inspired by platforms like LeetCode, with an esports-style layer that enables real-time 1v1 coding battles, rankings, and tournaments.

The system supports solo practice, secure code execution with automated judging, real-time match rooms, leaderboards with ELO ratings, tournament brackets, analytics dashboards, and gamification.

Key actors

Actor	Description
Player	Registers, solves problems, participates in 1v1 battles and tournaments, views stats and rankings.
Admin	Manages problems, tournaments, and monitors analytics; has elevated permissions.

Functional requirements by module

Module 1 - User Accounts & Authentication

- **FR-1.1** - Users can register with email/username and password; the system validates uniqueness and password policy.
- **FR-1.2** - Users can log in and receive a JWT-based session token; the token is required for protected APIs.
- **FR-1.3** - Users can manage their profile: display name, avatar, bio, preferred programming languages.
- **FR-1.4** - Roles are supported: Player and Admin; Admin access is restricted by role-based authorization.
- **FR-1.5** - Users can view their match history and submission history filtered by date, mode, and outcome.
- **FR-1.6** - Optional: users can add friends and view friends' public profiles and recent activity (privacy controlled).

Implementation notes

- Angular: Reactive forms, route guards, interceptors for JWT.
- Spring Boot: Spring Security, JWT filter, password hashing (BCrypt).

Module 2 - Problem Management

- **FR-2.1** - Admins can create, update, archive, and delete problems with title, statement, constraints, difficulty, tags.
- **FR-2.2** - Problems support multiple test cases: public samples and hidden judge tests.
- **FR-2.3** - Users can browse problems with search and filters (difficulty, tags, solved/unsolved).
- **FR-2.4** - Users can open a problem page showing statement, examples, constraints, and allowed languages.
- **FR-2.5** - Problem visibility modes: public, private (tournament-only), and draft (admin-only).

Implementation notes

- Spring Boot: REST CRUD, pagination, validation (Jakarta Bean Validation).
- DB: store statements, metadata, and test cases with access control for hidden tests.

Module 3 - Code Submission, Execution & Judging

- **FR-3.1** - Users can submit code for a selected problem and language.
- **FR-3.2** - The platform runs code against public tests for 'Run' and hidden tests for 'Submit'.
- **FR-3.3** - Judge returns verdicts: Accepted, Wrong Answer, Time Limit Exceeded, Memory Limit Exceeded, Runtime Error, Compilation Error.
- **FR-3.4** - The platform records each submission with timestamp, language, verdict, runtime, memory, and output (for public tests).
- **FR-3.5** - Execution is isolated and resource-limited (time/memory) to prevent abuse.
- **FR-3.6** - Users can view detailed submission results (verdict + per-test feedback for public tests).

Implementation notes

- Implementation options: (A) internal Docker-based sandbox runner, or (B) integrate a hosted judge service (e.g., Judge0).
- Security: strict resource limits, container isolation, input/output size limits, and audit logging.

Module 4 - Real-Time 1v1 Battles

- **FR-4.1** - Players can create a private challenge or enter matchmaking for ranked 1v1.
- **FR-4.2** - When a match starts, both players receive the same problem, timer, and room identifier.
- **FR-4.3** - Real-time events are pushed to clients: join/leave, countdown, submissions status, and match end.
- **FR-4.4** - Winner rules: first Accepted submission wins; tie-breakers may include faster runtime or fewer penalties.
- **FR-4.5** - A match summary is generated: duration, submissions count, winner, rating change, and replay links.
- **FR-4.6** - Disconnection handling: reconnect window and fair forfeit rules.

Implementation notes

- Spring Boot: WebSocket/STOMP for real-time rooms and events.
- Angular: live match UI, timers, and state updates via WebSocket client.

Module 5 - Ranking, ELO & Leaderboards

- **FR-5.1** - The platform maintains an ELO rating for each player in ranked mode.
- **FR-5.2** - After each ranked match, ratings are updated and recorded with before/after values.
- **FR-5.3** - Leaderboards show top players with filters (global, friends, time range).
- **FR-5.4** - Tier system (optional): Bronze/Silver/Gold/Diamond/Master based on rating thresholds.
- **FR-5.5** - Anti-abuse rules: minimum games before appearing on leaderboards; detect repeated matches vs same opponent.

Implementation notes

- Backend provides rating and leaderboard APIs with pagination and caching.
- Frontend provides sortable, searchable leaderboard pages.

Module 6 - Tournaments

- **FR-6.1** - Admins can create tournaments with start/end dates, rules, and problem pool.
- **FR-6.2** - Tournament formats supported: single elimination bracket (baseline) and optional Swiss/round-robin.
- **FR-6.3** - Players can register for tournaments; registration can be open or invite-only.
- **FR-6.4** - Matches are scheduled and participants receive notifications in-app.
- **FR-6.5** - Tournament pages show bracket progression, match results, and final standings.

Implementation notes

- Bracket generation and progression logic implemented in backend services.
- Angular UI for bracket visualization and match pages.

Module 7 - Analytics & Statistics

- **FR-7.1** - Players have a dashboard showing solved counts by difficulty and tag.
- **FR-7.2** - Players can see performance metrics: win/loss ratio, average solve time, favorite language, streaks.
- **FR-7.3** - Admins have platform analytics: active users, submissions/day, match volume, error rates.
- **FR-7.4** - Charts and trends can be filtered by date range and mode (practice vs ranked vs tournaments).

Implementation notes

- Angular: charts (e.g., ngx-charts or Chart.js).
- Spring Boot: aggregated queries + optional scheduled jobs for daily stats snapshots.

Module 8 - Gamification & Notifications

- **FR-8.1** - XP system: users gain XP for accepted solutions, ranked wins, and tournament placements.
- **FR-8.2** - Achievements/badges: e.g., First Win, 10 Accepted, 7-day streak, Fast Solver, Tournament Champion.

- **FR-8.3** - Daily/weekly challenges: rotating tasks with rewards.
- **FR-8.4** - Notification center: match invites, tournament reminders, badge earned, rating changes.
- **FR-8.5** - Users can configure notification preferences (email optional, in-app required).

Implementation notes

- Backend: notification service + persistence; optional email integration.
- Frontend: notification drawer + unread counts.

Non-functional requirements

NFR-1 Security

- JWT authentication with role-based access control for all protected endpoints.
- Secure code execution with sandboxing and strict resource limits.
- Input validation, rate limiting on submission endpoints, and audit logs for admin actions.

NFR-2 Performance & Scalability

- Pagination on list endpoints (problems, submissions, leaderboards).
- Asynchronous judging queue for submissions and battles (recommended).
- Caching for leaderboards and problem lists where appropriate.

NFR-3 Reliability

- Graceful handling of judge failures: retries, clear error statuses, and incident logs.
- WebSocket reconnect logic during live matches.

NFR-4 Usability

- Responsive UI (desktop-first, mobile-friendly).
- Accessible forms and clear feedback for verdicts and match outcomes.

Expected deliverables

- Angular SPA with modules, routing, and state management for real-time battles.
- Spring Boot REST API + WebSocket endpoints, secured with Spring Security + JWT.
- Relational database schema (users, problems, tests, submissions, matches, tournaments, ratings, notifications).
- Docker-based local development (recommended) and deployment documentation.

Assumptions & constraints

- The platform will initially support a limited set of languages (e.g., Java, Python, C++) to simplify judging.

- The judge component will run isolated (Docker sandbox) and may be deployed as a separate service from the API.
- Practice mode and ranked mode use the same problem bank; tournament mode can use a restricted problem pool.
- Privacy: profiles are public by default, with optional settings to hide activity and match history.

Out of scope for MVP

- Mobile native apps (iOS/Android) - web responsive UI only for this phase.
- Monetization (subscriptions/payments).
- Public discussion forums and full social network features.

Future enhancements

- Spectator mode with delayed live view of battles.
- Anti-cheat features (plagiarism detection, suspicious behavior scoring).
- Team battles (2v2) and 'arena seasons' with resets and rewards.
- Code editor enhancements: templates, linting, and saved snippets.