Project Title: Coding Battle Arena (Web + Android App)

Project Type: Full-Stack Development (Web + Mobile)

Inspired By: Chess.com, Clash Royale

🌎Overview

A gamified platform for competitive programming battles where players engage in 1v1 problem-solving duels. With rating tiers, clans, real-time emotes, and detailed post-match analysis, the app motivates learning through competition.

SRS (Software Requirements Specification)

1. Functional Requirements

1.1 User Authentication & Profiles

- Sign up/login with email, Google, or GitHub
- Create username, upload avatar
- View and edit profile, view tier badge, ranking, win streak
- Search users by usercode

1.2 Game Arena

- Real-time 1v1 matchmaking
- Unsolved problem served to both players
- Code editor with language support (C++, Python, Java)
- Timer system (Rapid, Blitz, Classical)
- Submit button locks answer
- Code remains hidden until match ends
- Result calculation:
- Winner by test cases passed
- Tie-breakers: Time complexity > Space complexity > Submission time
- Draw if all same

1.3 Battle History

- Log of previous matches
- Opponent info
- Test case results
- View both users' codes
- Option to analyze and optimize

1.4 Rating & Tiers

- Tier system: Iron, Bronze, Silver, Gold, etc. (I, II, III)
- Elo-based rating change on win/loss
- Draw: minor change or none
- World, country, and friend rankings

1.5 Friends & Chat

- Send/accept friend requests
- Show online/last active friends
- Invite for friendly battle
- Real-time emotes and messages during matches

1.6 Clans

- Create or join a clan
- Clan tier, rating, war history
- Intra-clan friendly battles
- Inter-clan wars (matchmaking and ratings)

1.7 Code Judging & Anti-Cheat

- Judge0/Docker-based execution
- Backend stores optimal solution for each question
- Compare time/space complexity and runtime
- No copy-paste allowed
- Plagiarism/AI detection
- Ban system for cheaters

---

2. Non-Functional Requirements

2.1 Performance

- Matchmaking <2 seconds
- Code execution <5 seconds

2.2 Security

- JWT auth or Firebase
- Secure REST APIs
- Input sanitization, sandboxed code execution

2.3 Scalability

- Support for 100k+ concurrent users
- Horizontal scaling (load balancer, database replica)

2.4 Availability

- Uptime > 99.5%

2.5 Maintainability

- Modular codebase
- RESTful API documentation (Swagger/Postman)

🐃Project Roadmap

Phase 1: Planning & Setup (Week 1-2)

- Finalize features & flow
- Choose tech stack
- Design schema & SRS (this doc)
- Setup GitHub repo & CI/CD pipelines

Phase 2: Core Backend (Week 3-6)

- Auth system (JWT/Firebase)
- Users, profiles, ratings, matchmaking APIs
- Problem delivery & history tables
- Judge0 integration for code execution

Phase 3: Frontend Web (Week 6-10)

- React UI: login, lobby, profile, battle arena
- Live timer, code editor, submission flow
- Socket.IO for real-time communication

Phase 4: Android App (Week 10-14)

- Flutter/Kotlin app mirroring web features
- Login, matchmaking, arena, ratings

Phase 5: Advanced Features (Week 15-18)

- Friends & clan system
- Emotes and in-game chat
- Problem tagging, difficulty filter, dynamic pool

Phase 6: Testing & Optimization (Week 19-20)

- QA, bug fixing, performance optimization
- Load testing for matchmaking & judge engine

Phase 7: Deployment (Week 21+)

- Host backend (Render, AWS, Railway)
- Vercel/Netlify for frontend
- Play Store beta release
- Monitor & iterate

🐊Suggested Tech Stack

Layer	Tech
Frontend	React + Tailwind CSS
Mobile	Flutter / Kotlin
Backend	Node.js + Express / Django
DB	PostgreSQL + Redis
Code Execution	Judge0 / Docker Sandbox
Real-Time	Socket.IO
Hosting	Vercel, Render, Firebase

Let me know when you're ready to begin with the first module (e.g., auth, matchmaking, arena)!