Lila Backend Assignment A — Multiplayer Tic-Tac-Toe with Nakama

Problem Statement Build a production-ready multiplayer Tic-Tac-Toe game with a server-authoritative backend using Nakama.

Technical Requirements

Frontend (User Front End)

- Choose any modern stack (React, React Native, Flutter, Unity, etc.)
- Implement responsive UI optimized for mobile and desktop
- Render real-time game state updates from the server
- Show player information, match status, and turn indicators
- Provide UX for creating/joining matches and returning to lobby

Backend (Nakama)

Core Requirements

- Server-authoritative game logic: all state managed and validated on the server
- Validate moves server-side; reject invalid or out-of-turn actions
- Prevent client manipulation; never accept client-computed results
- Broadcast validated state updates to all participants

Matchmaking System

- Allow players to create new game rooms and invite/auto-match
- Implement automatic matchmaking to pair players
- Support room discovery and joining; handle reconnects
- Handle player connections/disconnections gracefully (forfeit or pause rules)

Deployment

- Deploy Nakama to a cloud provider (AWS/GCP/Azure/DigitalOcean)
- Deploy the frontend publicly or provide installable mobile build
- Include deployment configuration and instructions

Optional Features (Bonus Points)

Concurrent Game Support

- Support multiple simultaneous sessions with proper room isolation
- Ensure scalability for concurrent players

Leaderboard System

- Track wins, losses, streaks; maintain global rankings
- Display top players with statistics; persist performance data

Timer-Based Game Mode

- Add per-turn time limits (e.g., 30s) with countdown UI
- Auto-forfeit on timeout; matchmaking supports classic vs. timed modes

Deliverables

- Source code repository (GitHub/GitLab)
- Deployed and accessible game URL or installable mobile app
- Deployed Nakama server endpoint

- README including:
    - Setup and installation instructions
    - Architecture and design decisions
    - Deployment process documentation
    - API/server configuration details
    - How to test the multiplayer functionality

How to test the multiplayer functionality

Environment Setup

- Start Nakama locally or use your cloud deployment; note the server endpoint
- Start two frontend clients (two browser windows/devices or two simulators)
- Ensure both clients authenticate and join the same match (auto-match or room code)

Happy-Path Gameplay

- Player X makes a valid move; verify server accepts and broadcasts state to both clients
- Player O responds; continue until win/draw; verify winner/draw broadcast and match closure
- Confirm board resets or lobby navigation according to your UX

Invalid Move Handling

- Attempt to play on an occupied cell; expect server rejection and no state change
- Attempt to play out of turn; expect server rejection and informative error

Server Authority & Cheat Prevention

- Tamper with client-side state (e.g., force three in a row locally); verify server ignores
- Try sending a crafted client payload; verify server validation rejects it

Disconnections & Reconnects

- Disconnect one player mid-game; verify server rule: pause/forfeit/reassign
- Reconnect the player; server should restore session or enforce forfeit per rules

Concurrency

- Create multiple matches with different players; verify isolation and no cross-room leakage

Timer Mode (if implemented)

- Enable timed mode; verify countdown UI and server-side timeout enforcement
- Let timer expire; expect auto-forfeit and broadcast of result

Matchmaking

- Test auto-match: two clients queue into a match; verify pairing and lobby flow
- Test room discovery/joining: one creates, second discovers/joins; verify permissions

Leaderboard (if implemented)

- Complete several matches; verify stats recorded and rankings updated server-side
- Refresh UI to confirm leaderboard reflects latest results