Cortex Clash - Technical Audit Report

1. Executive Summary

Project Status: Late-Stage Beta / Pre-Production

The Cortex Clash platform demonstrates a robust implementation of core tournament features, authentication, and real-time updates. The architecture is sound, utilizing a microservices-lite approach for AI features. However, critical deployment blockers exist regarding the ML service integration and environment configuration. The project is not yet production-ready due to hardcoded local URLs and missing deployment infrastructure for the Python service.

2. Architecture Overview

System Diagram & Interaction

1. Frontend (Client): React + Vite
   - Serves as the UI for Players, Organizers, and Admins.
   - Communicates with Backend via REST API.
   - Real-time updates via Socket.IO.
   - Hosting: GitHub Pages.

2. Backend (Server): Node.js + Express
   - Orchestrates logic, DB, and Auth.
   - Database: MongoDB Atlas.
   - Hosting: Render.

3. ML Service: FastAPI (Python)
   - Win probability and anomaly detection.
   - Hosting: Pending (Local-only).

3. Feature Completion Status

Feature Cluster	Status	Notes / Issues
Authentication	Complete	JWT/Bcrypt/Roles implemented.
Tournament Sys	Complete	Creation, Brackets, Results.
Real-time	Partial	Socket.IO implemented, needs CORS test.
Ranking System	Complete	Elo, Seasons. CRITICAL: Hardcoded ML URL.

Cortex Clash - Technical Audit Report

Game Logic	Complete	Generic Game model supported.
Seasons	Complete	Active season tracking.
Admin Panel	Complete	Full management features.
AI Prediction	Partial	Model ready. Deployment MISSING.
Integrity Sys	Complete	Anomaly detection logic ready.
Analytics	Partial	Basic stats only.

4. Production Readiness Evaluation

PROD READY: Frontend Build, DB Schema, API Security.

NEEDS IMPROVEMENT: CORS Config, Error Handling.

CRITICAL BLOCKERS:

1. Hardcoded ML Service URL (localhost stuck in code).
2. No deployment config for ML Service.
3. Missing environment variables.

5. Technical Debt & Risks

- Critical: Node backend coupled to local ML service. Fails in prod.
- Critical: JWT_SECRET hardcoded fallback is unsafe.
- Scalability: Synchronous ML calls block request threads.

6. Requirements for v1.0

1. Deploy ML Service to public URL.
2. Add ML_SERVICE_URL to backend config.
3. Enable Security Headers (xss-clean).
4. Finalize Frontend deployment to GitHub Pages.

7. Completion Estimate

Backend: 90% | Frontend: 85% | ML: 80% | DevOps: 40%

OVERALL PROJECT STATUS: 75%