Fanfinity CTO Case Study

Sanabil Venture Studio | Technical Leadership Assessment

Task 1

Company Context

Fanfinity is a Sanabil Studio-backed sports technology venture building the digital backbone of Saudi Arabia's fan engagement ecosystem. We create platforms that connect fans, clubs, and media through live match feeds, predictor and trivia games, club CRMs, loyalty systems, and data-driven engagement tools extending into gaming and stadium technology using computer vision.

We're at an inflection point: we've proven product-market fit with early club partnerships and growing fan engagement, but need to transform from startup MVPs into an enterprise-grade platform capable of powering millions of fans across the Kingdom.

The Challenge

As our Chief Technology Officer, you'll inherit a fragmented technical foundation built during our rapid growth phase. Your mission is to establish the technology platform and engineering organization that can scale Fanfinity into the dominant fan engagement platform across Saudi Arabia and eventually the broader MENA region.

Present a comprehensive plan covering the following:

Part 1: Your First 90 Days

How will you assess the current state, build credibility, and create momentum?

Address

- Your approach to understanding our technology landscape, team capabilities, and business priorities
- Critical decisions you'll make in your first quarter
- Key relationships you'll build and how
- Early wins you'll pursue to establish credibility
- Deliverables we should expect by day 30 and Day 90

Part 2: 270-Day Strategic Foundation

What technology platform will you build to support our next phase of growth?

Address across these critical areas

- **Platform Architecture:** How will you build for scale, resilience, and 10x growth?
- **Product Velocity:** How will you accelerate innovation while maintaining quality?
- **Data & Intelligence:** How will data and AI power personalization and insights?
- **Security & Compliance:** How will you establish trust and meet Saudi regulatory requirements?
- **Team & Culture:** How will you build a world-class engineering organization?
- **Partnerships & Ecosystem:** How will you integrate clubs, stadiums, media partners, and third parties?

Part 3: Success Metrics & Investment

How will you measure success and what resources do you need?

Address

- Key metrics you'll track (business impact, platform performance, team health)
- Critical milestones for months 3 and 12
- Technology investment required (team size, infrastructure, tools)
- Major risks and your mitigation strategies

Constraints & Assumptions

- Assume we have 3-5 engineers currently with mixed skill levels
- Current architecture is across multiple technology stacks mainly Swift, React, Kotlin etc
- We're focusing on both B2B and B2C both.
- You'll have backing from Sanabil Studio but need to demonstrate ROI
- Saudi Vision 2030 creates massive opportunity but also regulatory requirements
- Competition is intensifying from both regional and international players

What We're Evaluating

- **Execution Realism:** Are your timelines, milestones, and priorities credible?
- **Technical Depth:** Do your architectural choices demonstrate sound engineering judgment?
- **Leadership Maturity:** Can you build, inspire, and scale a high-performing team?
- **Communication:** Can you make complexity simple for board and business stakeholders?

- **Incident Response:** Describe your approach when the platform experiences major downtime during a peak match. Who do you communicate to, how, and what changes post-incident?

Task 2: Coding Challenge

Overview

*Build a **Simple Real-Time Fan Engagement Analytics Service** that demonstrates your hands-on technical capabilities in the context of Fanfinity's platform needs. This microservice will process live match events and generate engagement metrics, core functionality for our fan engagement platform.*

The Challenge

Business Context

During a major match (e.g., Al-Hilal vs. Al-Nassr), Fanfinity expects:

- *50,000+ concurrent users*
- *Event spikes of 1,000+ requests/second during goals*
- *Sub-200ms API response times*
- *Zero data loss during traffic surges*

Your service should demonstrate architectural decisions that address these requirements.

What You'll Build

Create a standalone microservice that:

1. ***Ingests match events** via REST API (goals, cards, substitutions, etc.)*
2. ***Processes events in real-time** and calculates engagement metrics*
3. ***Exposes metrics via API** for consumption by frontend applications*
4. ***Demonstrates scalability patterns** suitable for handling match-day traffic spikes*

Technical Requirements

Core Functionality

```HTML
POST /api/events

- Accept match events (goal, card, substitution, match_start, match_end)

- Validate event data

- Process asynchronously if needed

- Return 202 Accepted with event ID

GET /api/matches/{matchId}/metrics

- Return real-time engagement metrics:

  - Total events

  - Events by type

  - Peak engagement periods (events per minute)

  - Response time percentiles (p50, p95, p99)

GET /metrics

- Prometheus-compatible metrics

- Request rates, error rates, latency

- Business metrics (events processed, matches active)
```

Data Model (Suggested)

```None
Match Event:

- eventId (UUID)

- matchId (string)
- eventType (enum: goal, yellow_card, red_card, substitution,
etc.)

- timestamp (ISO 8601)

- teamId (string)

- playerId (optional string)

- metadata (JSON object for extensibility)
```

Technology Stack (Your Choice)

Language Options: *Choose one that best demonstrates your expertise*

- *Go (preferred for high-performance services)*
- *Python (FastAPI/Django)*
- *Node.js (TypeScript/Express)*
- *Java/Kotlin (Spring Boot)*
- *Rust (if you want to impress)*

Required Components

- *REST API framework*
- *Data persistence (PostgreSQL, MongoDB, or Redis)*
- *Containerization (Docker)*
- *CI/CD pipeline (GitHub Actions)*
- *Infrastructure as Code (optional but valued)*

Deliverables

1. Source Code Repository

Must include

- *Complete, working application code*
- `README.md` *with:*
  - *Architecture overview and key decisions*
  - *Setup instructions*
  - *API documentation*
  - *Assumptions and trade-offs made*

    - *Known limitations and production considerations*
- `Dockerfile` *with multi-stage build*
- `docker-compose.yml` *for local development*
- `.env.example` *with configuration template*

2. CI/CD Pipeline

GitHub Actions workflow that:

- *Runs on push to main and pull requests*
- *Executes linting and code quality checks*
- *Runs unit tests with coverage reporting (>70% target)*
- *Builds Docker image*
- *Tags images with commit SHA and version*

4. Testing & Quality

Demonstrate

- *Unit tests for core business logic*

5. Documentation

Include brief written responses (1-2 paragraphs each)

1. ***Architecture Decisions:*** *Why did you choose this language, framework, and data store?*
2. ***Scalability Approach:*** *How does your design handle 10x traffic? What would break first?*
3. ***Production Readiness:*** *What's missing for production? What would you add with more time?*

Evaluation Criteria

Code Quality (25%)

- *Clean, maintainable code following language conventions*
- *Proper error handling and logging*
- *Configuration management (12-factor app principles)*
- *Code organization and structure*

DevOps Practices (25%)

- *Effective CI/CD pipeline*

- *Container optimization (image size, layers, security)*
- *Infrastructure as Code quality*
- *Deployment strategy*

System Design (25%)

- *API design and REST principles*
- *Performance considerations*
- *Scalability patterns*
- *Observability (logging, metrics, tracing)*

Testing & Reliability (15%)

- *Test coverage and quality*
- *Error handling and resilience*
- *Health checks and graceful degradation*
- *Load testing approach*

Documentation (10%)

- *Clear setup instructions*
- *Architectural reasoning*
- *Trade-off explanations*
- *Production considerations*

Constraints

- ***Time Commitment:*** *Maximum 3 hours (we're timing demonstration of skills, not perfection)*
- ***Scope:*** *Focus on one microservice only*
- ***Complexity:*** *Don't over-engineer, show you can balance pragmatism with quality*
- ***External Services:*** *Mock all external dependencies*

Submission

1. ***Create a public GitHub repository*** *with your solution*
2. ***Create a brief video*** *(3-5 minutes, Loom/unlisted YouTube) walking through:*
   - *Quick code overview*
   - *Running the application locally*
   - *Triggering the CI/CD pipeline*
   - *Demonstrating key API endpoints*
   - *Discussing one interesting technical decision*
3. ***Submit repository URL and video link***

Bonus Points (Optional)

These are NOT required but demonstrate advanced capabilities:

- *Comprehensive API documentation (OpenAPI/Swagger)*

What We're Looking For

This task reveals

- ***Can you ship?** Working code that actually runs*
- ***Production mindset:** Do you think beyond "works on my machine"?*
- ***DevOps fluency:** Are you comfortable with the full stack?*
- ***Pragmatic engineering:** Can you balance speed with quality?*
- ***Communication:** Can you explain your decisions clearly?*

Remember: We'd rather see a simple solution executed excellently than a complex solution executed poorly. Show us you can build the foundation Fanfinity needs.

Deliverables

Submit: Candidates may submit their work in any format they feel best showcases their thinking, technical depth, problem solving, and communication style. This can be a written document, a presentation deck, a technical architecture outline, or a mix of formats.

Present: Candidates will present their work in a dedicated session with up to 45 minutes for the presentation followed by 30 minutes of discussion with the CEO and Studio leadership.

Timeline: Submit 72 hours before interview; be prepared to defend trade-offs and adapt your plan based on constraints we'll introduce during discussion

This case study assesses your ability to think strategically, execute pragmatically, and lead effectively at the intersection of technology and business. Show us why you're the right CTO to build Fanfinity's technology foundation.