Engineering Take-Home Assessment

Position: Full-Stack Engineer

Assessment Overview

You will implement a referral system backend and conduct performance audits of existing services. This assessment evaluates your ability to design scalable systems, implement complex business logic, and analyze existing codebases for performance bottlenecks.

Extension: Create a frontend to go along with this (intentionally ambiguous to test judgement, innovation and adaptability).

Referral System Implementation

Business Requirements

Core Concept

Build a commission-based referral system where users earn a percentage of trading fees generated by their referred users' trading activity (spot only).

Our SQE (spot mechanism) splits the fee taken to the Nika treasury, cash back, and the referral commissions. The way this works is that the fee is taken in a consistent manner (USDC on Arbitrum or Solana, depending on where the trade happened (EVM vs SVM). The referral commission is transferred to a merkle root smart contract which effectively keeps track of how much each user is able to claim (i.e. cashback & commission earnings).

- This test includes the SQE fee bundling logic to redirect fees to both the treasury and the chain-relevant contract (with the breakdown calculation applied and fed into the contract).
- Note: the claiming contract itself is not in scope for the test so XP will be used to demonstrate how the actual crypto breakdown will work.

Note: we have the concept of fee cashback, separate to fee tiers, which are applied on sign up (e.g. if Daniel referred me then I get 10% fee cashback).

- E.g. Daniel refers Dennis. Dennis is in the base fee tier (1%). Since Dennis signed up with Daniel's referral link he gets a 10% cashback on his fee tier. Dennis is still charge 1%, but gets 10% as cashback. Daniel's commission is 30% of Dennis's fee.
- E.g. Daniel refers Dennis who refers Joe. Joe is in the lowest fee tier (0.5%). Joe still pays 0.5% fee, but gets 10% of the fee paid as cashback. Dennis gets 30% of the 0.5% fee, and Daniel gets 3% of the 0.5% fee.
- TL;DR Total Commissions = 35% of Fee paid, Cashback = 10% of Fee Paid. Treasury Revenue = 55% of Fee Paid.

Commission Structure

Implement a 3-level cascade system:

● **Level 1 (Direct Referral):** 30% of trading fees ● **Level 2 (Referral's Referral):** 3% of trading fees ● **Level 3 (Third Level):** 2% of trading fees

Example: If User A refers User B, who refers User C, who refers User D:

- User A earns 30% from B's trades, 3% from C's trades, 2% from D's trades
- User B earns 30% from C's trades, 3% from D's trades
- User C earns 30% from D's trades
- Ability to have custom structures for certain people:
    - 1. KOLs with only 50% direct comm
    - 2. KOLs with custom level comm
    - 3. Users with waived fees and comms

Technical Requirements

Database Schema

Design and implement models for:

- **Referral tracking:** Store referral codes, referrer-referee relationships
- **Commission tracking:** Track earned commissions by user, level, and source
- **Claim management:** Track claimed vs unclaimed commissions
- **Performance metrics:** XP earned (acts as analogy for total crypto earned)

Required fields per user:

- Unique referral code
- Referrer ID (if applicable)
- List of direct referrals

- Commission balance (by token type)
- Claimed amount history
- Timestamp data for all actions

API Endpoints

The below endpoints act as examples, consider what makes the most sense and validate with DF before implementing:

1. **POST /api/referral/generate**
    - Generate unique referral code for authenticated user
    - Ensure idempotency (don't regenerate if exists)
2. **POST /api/referral/register**
    - Register a new user with a referral code
    - Validate referral code exists
    - Prevent circular references
    - Limit referral depth to 3 levels
3. **GET /api/referral/network**
    - Return user's referral network (3 levels deep)
    - Include user details and level
    - Paginate results for large networks
4. **GET /api/referral/earnings**
    - Show breakdown of earnings per referred user
    - Group by level
    - Include both claimed and unclaimed amounts
    - Support date range filtering
5. **POST /api/referral/claim** (UI only - no implementation needed)
    - Endpoint signature and validation only
    - Should validate claimable amount exists
6. **POST /api/webhook/trade** (Simulate trading activity)
    - Accept trade data (user_id, volume, fees)
    - Calculate and distribute commissions across referral level
    - Handle concurrent requests safely

Technical Constraints

- Handle race conditions for commission calculations
- Ensure accurate decimal arithmetic for financial calculations
- Implement proper database transactions for commission distribution
- Design for horizontal scaling (assume distributed system)
- Use database indexes appropriately for query optimization
- Each user can only be referred by one referrer
- Referrals happen via URL parameter and also popup on sign up

Deliverables

1. **Database schema** (SQL or ORM definitions)
2. **API structure & implementation** with proper error handling
3. **Unit tests** for commission calculation logic
4. **Integration tests** for critical paths
5. **Brief documentation** of design decisions

Questions?

If you have questions about requirements, please document your assumptions and proceed with your best interpretation. This simulates real-world scenarios where requirements may be ambiguous.

Good luck!