# Tóm tắt README – Dự án amm-tool (Chinh Vuong)

## 1. Giới thiệu

Mục tiêu của «amm-tool»: phân tích và theo dõi dữ liệu AMM / token trên blockchain Solana. Hỗ trợ các tính năng như: tìm token, theo dõi giao dịch, holder distribution, đẩy thông báo realtime,…

## 2. Kiến trúc hệ thống (Technical Architecture)

- Client Layer: giao diện người dùng (web app) gửi yêu cầu tới backend.  
- Application Layer: gồm REST API server + WebSocket server.  
- Cache Layer: sử dụng Redis để cache dữ liệu, Pub/Sub và lock phân tán để tránh trùng lặp.  
- Data Layer: dùng TimescaleDB (phần mở rộng của PostgreSQL) để lưu lịch sử giao dịch, snapshot…  
- External APIs: tương tác với các API bên ngoài như Helius, Jupiter,… để lấy dữ liệu on‑chain và thị trường.

## 3. Luồng dữ liệu chính (Data Flow Scenarios)

- Cache Hit – Fresh Data: khi dữ liệu trong cache còn mới (< 30s) → API trả ngay trong vài chục ms.  
- Cache Hit – Stale Data + Push: khi cache có dữ liệu nhưng đã “cũ” (30‑60s) → trả dữ liệu cũ ngay, backend cập nhật async và đẩy update qua WebSocket.  
- Cache Miss + Distributed Lock: khi cache hết hạn hoặc không có dữ liệu → dùng lock phân tán để chỉ một request fetch dữ liệu từ external API.

## 4. WebSocket & Real‑Time Updates

- Client subscribe theo token hoặc chủ đề cụ thể.  
- Backend publish event vào Redis Pub/Sub khi có dữ liệu mới.  
- WebSocket server broadcast tới các client đã subscribe.  
- Hệ thống hỗ trợ heartbeat, auto‑reconnect, quản lý subscription state qua Redis.

## 5. Chiến lược Cache & Database

- Cache (Redis) là nguồn dữ liệu nhanh: TTL chia thành fresh (0‑30s), stale (30‑60s), expired (>60s).  
- Khi expired: cần fetch mới, lock được bật.  
- DB (TimescaleDB/PostgreSQL) lưu lịch sử dài hạn: giao dịch ví, snapshot pool, holder distribution…

## 6. Lưu trữ dữ liệu và định nghĩa bảng

- Các bảng chính: wallet\_transactions, pool\_snapshots, holder\_snapshots, tokens, api\_logs…  
- Chính sách lưu trữ: giao dịch chi tiết lưu 30 ngày, snapshot pool lưu 7 ngày realtime, 30 ngày aggregate.

## 7. Bảo vệ Rate‑Limiting & Distributed Locking

- Trước khi fetch external API, kiểm tra cache. Nếu cần fetch mới, dùng Redis SETNX (lock) để chỉ một request thực hiện fetch.  
- Sau fetch: update cache, DB, publish event, release lock.  
- Giúp tránh gọi API bên ngoài nhiều lần, giảm rate‑limit violation và tối ưu độ trễ.

## 8. Triển khai & Yêu cầu

- Hệ thống stateless backend – dễ scale theo chiều ngang.  
- Cấu hình Redis, TimescaleDB, external API keys (Helius, Jupiter,…).  
- Frontend kết nối WebSocket để nhận cập nhật realtime.