Solana Protocol Architecture Diagram & Completion

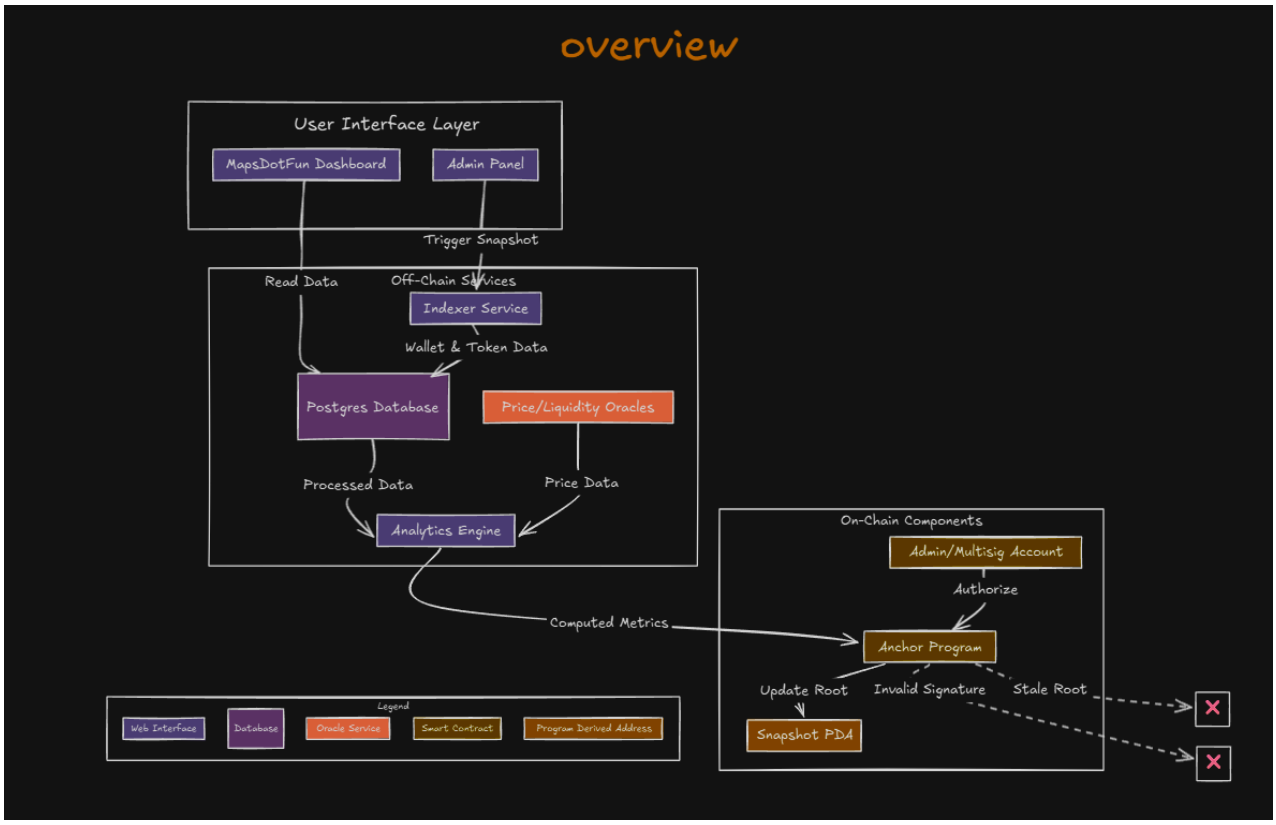
Name: Anudeep

Address: *2SZyg3ZgvmpE2FkqKnhtyz5xj4twnVcv8vaDyrVJszuv*

Protocol POC Requirements:

Objective: A Solana analytics protocol that maps and analyzes wallet clusters, token ownership, and transaction patterns to expose centralization and risk indicators.

- The protocol shall allow indexing and ingestion of wallet and token data from the Solana blockchain using external APIs such as Helius, SolanaFM.
- The protocol shall construct Merkle trees from aggregated token holder snapshots to produce verifiable Merkle roots.
- The protocol shall allow anchoring of computed Merkle roots on-chain via an Anchor-based Solana program.
- The protocol shall store each Merkle root in a Snapshot PDA, derived deterministically from the token mint address.
- The protocol shall restrict snapshot anchoring to an authorized admin account.
- The protocol shall maintain a Config Account to store protocol parameters, such as admin public key and update cadence.
- The protocol shall emit an on-chain event after successful snapshot anchoring for off-chain indexers to listen and update state.
- The protocol shall allow users to verify the ownership distribution of a given token by fetching the latest anchored root and corresponding Merkle proof.
- The protocol shall maintain an off-chain Postgres database to store raw snapshots, proofs, and derived analytics.
- The protocol shall integrate with an Analytics Engine to compute clustering metrics, liquidity concentration, and ownership risk indices.
- The protocol shall allow users to access analytics and visualization data through a frontend dashboard.
- The protocol shall use admin-signed validation to prevent unauthorized or stale Merkle root submissions.
- The protocol shall include timestamp and nonce checks to prevent replay or outdated snapshot anchoring.
- The protocol shall expose a proof verification function allowing client-side validation between off-chain and on-chain data.
- The protocol shall maintain clear separation of on-chain (Anchor program + PDAs) and off-chain (indexer + analytics) responsibilities.



1. Protocol Overview

1 - Indexer Fetches On-Chain Data

- The off-chain Indexer Service fetches wallet and token data from Helius.
- The service aggregates token ownership and transaction activity for a specific token mint.
- The raw snapshot data is normalized and stored in the Postgres database.

2 - Merkle Root Generation

- The Analytics Engine processes the collected wallet data.
- It builds a Merkle tree representing all token holders and their balances.
- A Merkle root is generated, representing the state of ownership at that timestamp.

3 - Root Anchoring on Solana

- The admin submits the Merkle root to the Anchor Program.
- The Anchor Program creates or updates a Snapshot PDA for that token mint.
- The PDA stores the mint address, Merkle root, timestamp, and admin signature.
- The program emits an event signaling a new anchored snapshot.

4 - Risk Analysis and Clustering

- The Analytics Engine computes ownership concentration, liquidity distribution, and behavioral clustering.
- The computed risk metrics are linked to the corresponding Merkle root in the database.
- External oracle data (price/liquidity) used for enhanced analytics.

5 - User Queries and Visualization

- A user visits the MapsDotFun dashboard or interacts via API.
- The frontend requests the latest anchored Merkle root and off-chain analytics for a token.
- The dashboard visualizes ownership clusters, risk indices, and liquidity maps using D3.js.

6 - Proof Verification

- When a user verifies ownership data, the frontend fetches the Merkle proof from the backend.
- The client-side verifier recomputes the proof and compares it to the on-chain Merkle root.
- If verified, data is confirmed authentic; otherwise, a “Data Integrity Warning” is displayed.

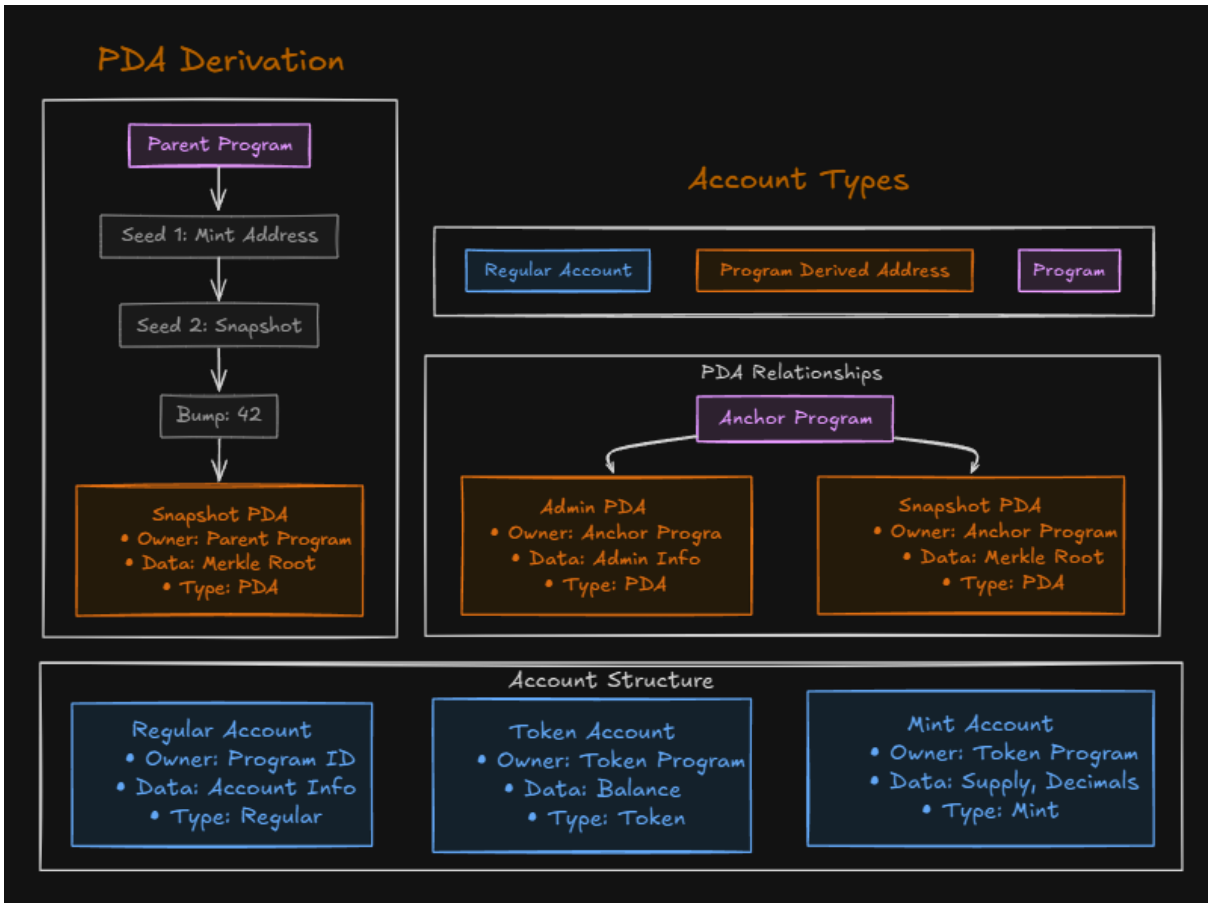
7 - Periodic Updates and Fee Handling

- The Indexer periodically fetches fresh on-chain data to produce updated snapshots.
- Each update triggers a new Merkle root anchoring transaction.
- Any minor protocol fees (if applicable) are sent to a separate fee vault account for protocol sustainability.

8 - Governance and Admin Controls

- The Config Account maintains admin addresses.
- Admins can rotate authorities or update configuration parameters through authorized transactions.

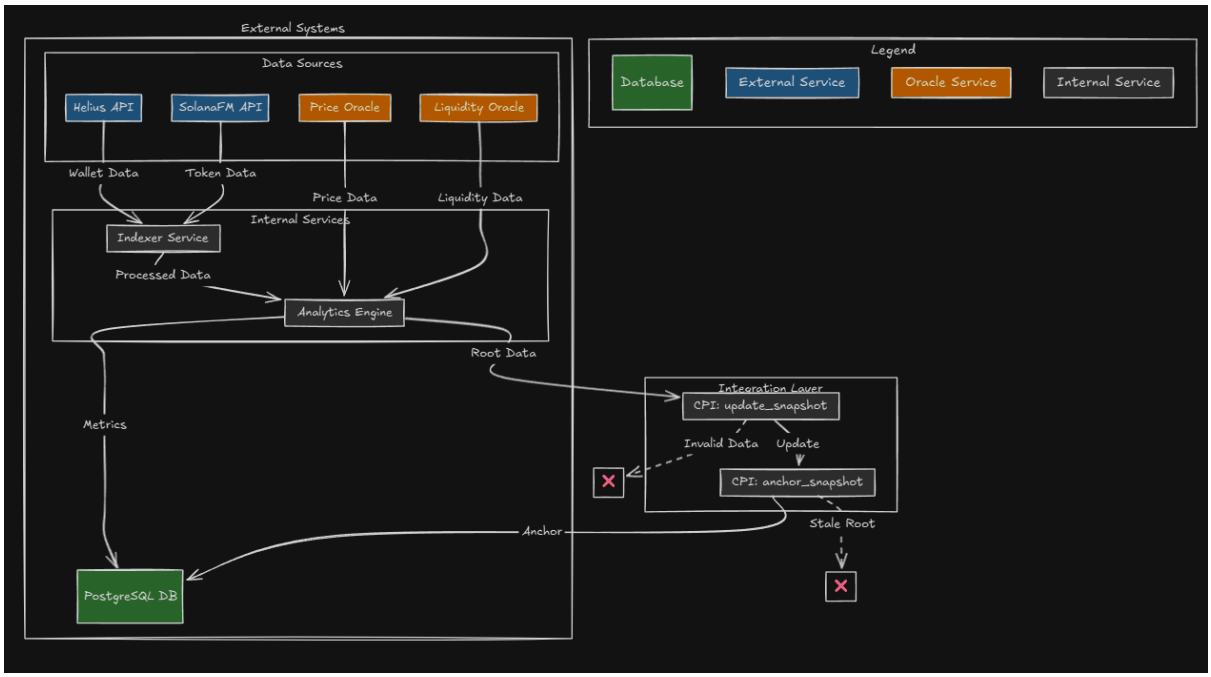
2. Account Structure Mapping



Key aspects of the account structure diagram:

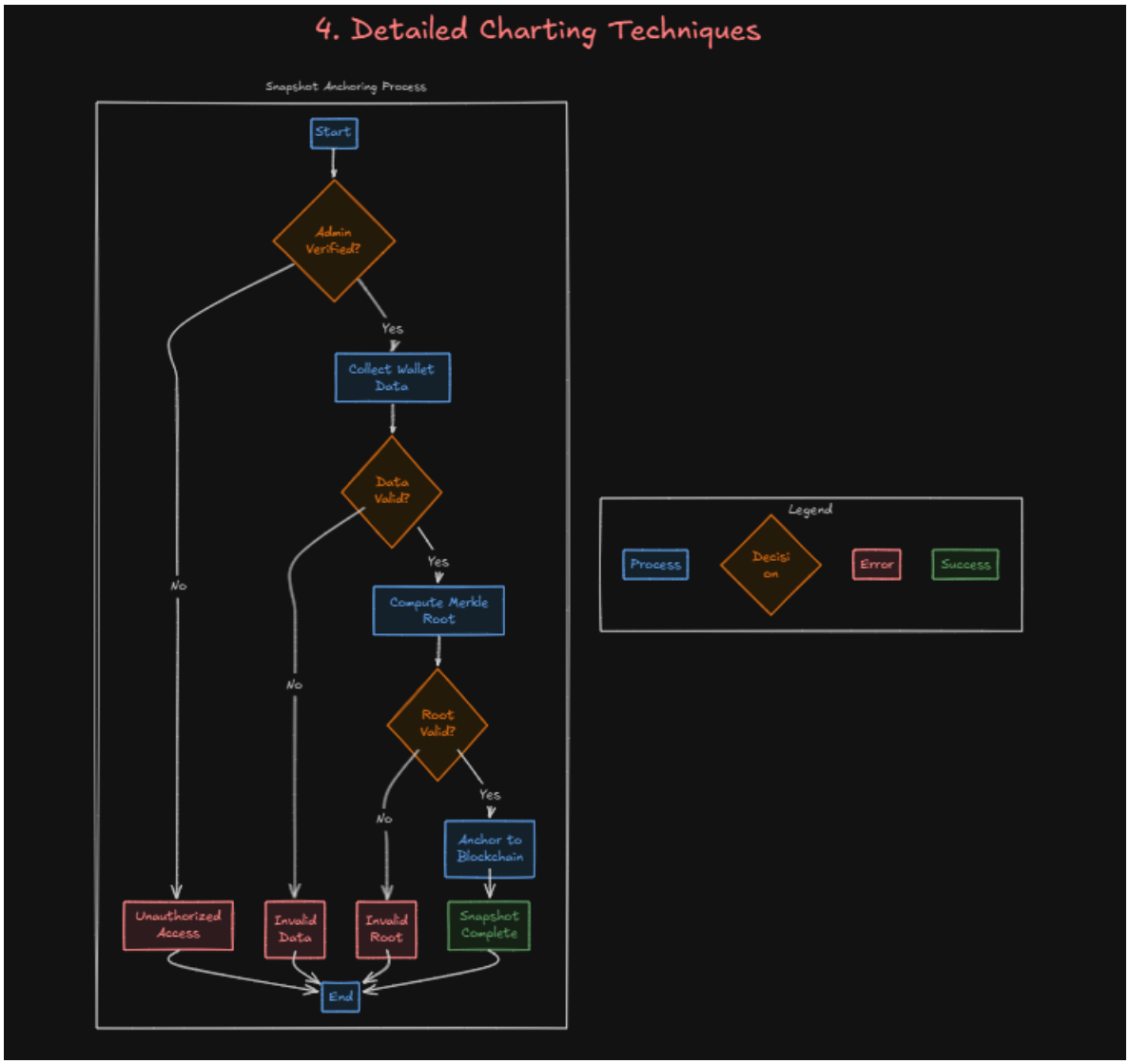
1. **Account Types and Structure**:
    - Regular accounts store basic program data and are owned by their respective programs
    - Token accounts and mint accounts follow the SPL token standard
    - PDAs are special accounts that can be deterministically derived from seeds
2. **PDA Derivation Process**:
    - The bump value (42) ensures the derived address lies off the Ed25519 curve
    - This mathematical property prevents private key generation for PDAs
    - Only the parent program can authorize operations for its PDAs
3. **PDA Relationships**:
    - PDAs are owned by their parent program, enabling secure program control
    - The Admin PDA and Snapshot PDA are derived from the Anchor Program
    - This structure allows MapsDotFun to maintain secure, program-controlled operations

3. External Dependencies and Integrations



1. **Integration Layer (CPI)**:
    - CPI (Cross-Program Invocation) enables secure communication between services
    - Two-step CPI process ensures data validation before on-chain anchoring
    - First CPI validates metrics, second CPI performs the actual anchoring
2. **Data Flow Patterns**:
    - External data sources (Helius, SolanaFM) feed raw data to Indexer Service
    - Analytics Engine aggregates data from multiple sources before processing
    - Processed data flows through CPI layer for on-chain operations
    - Database stores both metrics and anchored data for verification
3. **Error Handling**:
    - Invalid Data: Prevents corrupted metrics from reaching the blockchain
    - Stale Root: Ensures snapshot updates are based on current state
    - Both errors are caught at the CPI layer before on-chain execution

4. Detailed Charting Techniques



5. Detailed Architecture Sections

A. User Interaction Flow

1. Data Discovery and Visualization

- The user searches for a token on the MapsDotFun dashboard.
- The frontend requests ownership data and the latest Merkle root from the backend and on-chain PDA.
- The ownership distribution, risk index, and wallet clusters are displayed in an interactive visual map.

2. Verification Process

- When the user selects a specific wallet or data point, the protocol fetches the corresponding Merkle proof from the off-chain database.

- The frontend verifies the proof against the on-chain Merkle root using client-side logic.
- A visual indicator confirms authenticity or flags a mismatch.

3. Periodic Updates

- Users receive updated metrics as the Indexer refreshes snapshots periodically (every few hours or daily).
- Notifications or visual refreshes indicate when new snapshot data is anchored.

B. Program Interaction Matrix

1. Cross-Program Calls (CPIs)

- The Anchor Program interacts with the Config Program for authorization validation.
- Anchor emits events that can trigger off-chain indexer updates.
- Optional CPI hooks may be added for interacting with external vault or fee management programs.

2. Data Transmission

- Off-chain → On-chain: Merkle root submission via anchor_snapshot() instruction.
- On-chain → Off-chain**:** Event logs parsed by indexers to update UI and analytics.
- Frontend → Backend**:** User queries for proofs, analytics, or token search results.

3. Control Flow

- Control originates from user or admin actions → flows through the frontend → backend services → on-chain programs.
- Access control enforced through signature checks and PDA ownership validation.

C. Account Management

1. Account Creation

- Snapshot PDA is created deterministically for each token mint (seed = [b"snapshot", mint_pubkey]).
- Config Account initialized once to store admin authority.

2. State Updates

- When new snapshots are anchored, the PDA's stored Merkle root and timestamp are updated.

- Off-chain analytics database mirrors this state for faster query access.

3. Ownership Transfer

- If admin rotation or authority transfer occurs, the Anchor Program allows secure update of the Config Account via set_admin() instruction.
- Old admin privileges are revoked once the transaction is confirmed.

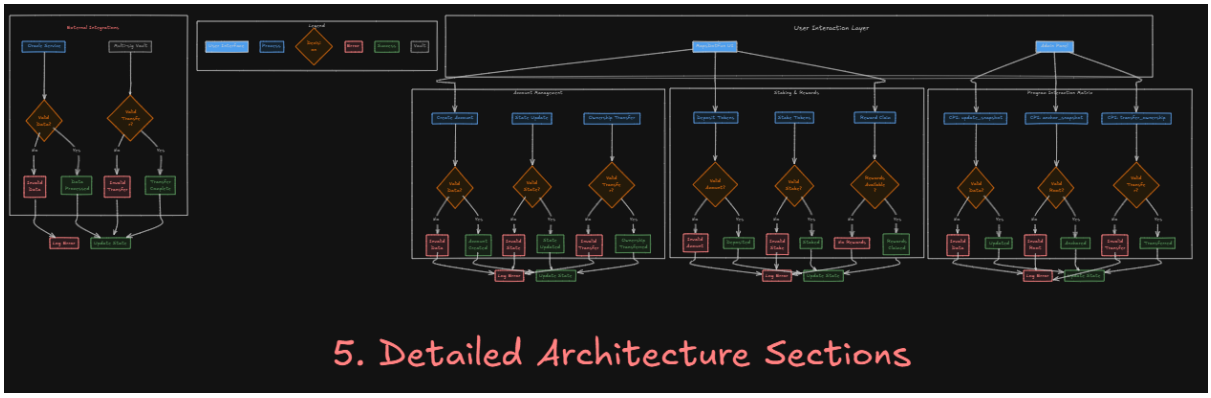
D. External Integrations

1. Compliance Checks

- Before anchoring data, the backend may run compliance checks (e.g., via OFAC or wallet risk APIs).
- Suspicious or sanctioned wallets are flagged and excluded from snapshots.

2. Oracle Interactions

- The Analytics Engine may fetch token price and liquidity data from Pyth or Switchboard oracles.
- This data is used to compute real-time liquidity concentration and market risk metrics.



Architecture Diagram: View all Live Diagrams on Exaclidraw