Dynamic-Reward-Allocation-Formula-DRAF

This document outlines the approach to implementing the requested features for the Solana-based project involving a Dynamic Reward Allocation Formula (DRAF) and associated fee logic. The project will consist of one immutable smart contracts:

- **Token Contract**: Manages token creation and allocation, and implements the reward distribution mechanism.
- **Transaction Contract**: Handles transaction-based fee deductions, reward distribution, and liquidity pool (LP) management.

Objectives

The project aims to achieve:

1. Implementation of a token with the Dynamic Reward Allocation Formula (DRAF).
2. Integration of a custom fee structure for minting and transactions.
3. Monitoring and enforcement of a fallback condition for the Liquidity Pool (LP) balance.

Dynamic Reward Allocation Formula (DRAF)

The DRAF is used to distribute rewards fairly among token holders based on their token holdings, proportional share, and holding time. Below is the formula:

```
\# Dynamic Reward Allocation Formula (DRAF)

def calculate_reward(T_reward, token_holding, total_weighted_tokens, beta_min, beta_m
    # Proportional share
    proportional_share = token_holding / max_share

    # Progressive bonus (beta)
    beta = beta_min + (beta_max - beta_min) * (1 - proportional_share)

    # Regressive penalty (alpha)
    alpha = alpha_min + (alpha_max - alpha_min) * proportional_share

    # Holding time multiplier
    holding_multiplier = min(holding_time / max_holding_time, 1)

    # Weighted tokens
    weighted_tokens = token_holding * (1 + beta - alpha) * (1 + holding_multiplier)

    # Reward calculation
    reward = T_reward * (weighted_tokens / total_weighted_tokens)

    return reward
```

Example Parameters

- Total reward pool: $T_{reward} = 10000$
- Token holdings: $T_i = 1000$
- Total weighted tokens: 50000
- Progressive bonus range: $\beta_{min} = 0.05$, $\beta_{max} = 0.15$
- Regressive penalty range: $\alpha_{min} = 0.02$, $\alpha_{max} = 0.10$
- Holding time: 6 months
- Maximum holding time: 12 months
- Maximum share: 20%

Output

The reward for a participant is calculated as:

$$R_i = T_{reward} \cdot \frac{T_i \cdot (1 + \beta - \alpha) \cdot (1 + H_{holding})}{\text{total\_weighted\_tokens}}$$

Fee Structure

Custom fees will be applied as follows:

- **Minting Fee (optional)**: Charged in SOL or other tokens.
- **Custom Fees (mandatory)**: Deducted from minted or transferred tokens:
  - 2% for the Liquidity Pool (LP).
  - 2% for DRAF-based rewards.
  - 0.85% for the marketing wallet.

Fallback Condition

To ensure ecosystem sustainability:

- The LP balance will be monitored.
- If the LP balance falls below 1000 tokens within 3 months, the entire LP will be transferred to the marketing wallet.

Liquidity Pool (LP) Creation and Maintenance

The Liquidity Pool (LP) is an essential part of this project for ensuring liquidity and enabling users to easily exchange tokens.

1. **LP Creation:** At the project's launch, the token contract will include an initialization function to create a liquidity pool on a DEX. This will involve depositing an initial amount of the token and a stable coin (like USDC or SOL) into the pool to enable trading.
2. **Fee Deduction:** Each time a transaction occurs (minting, buying, or selling the token), the contract will automatically deduct 2% of the transaction value and add it to the LP, ensuring the pool grows over time.
3. **LP Balance Monitoring:** The LP balance will be monitored regularly. If it falls below 1000 tokens, the contract will trigger the fallback condition, transferring the remaining balance to the marketing wallet to maintain sustainability.

Development Phases

1. **Token Contract Implementation**:
   - Define token properties.
   - Implement DRAF for reward distribution.
2. **Transaction Contract Implementation**:
   - Integrate fee deductions during minting and transfers.
   - Implement LP creation and management.
   - Enforce fallback conditions for LP.
3. **Testing**