Food Cooperative & Group Buying System - Tokenomics & Smart Contract Design

📌 Overview

This document outlines the **tokenomics model and smart contract logic** for a **decentralized food cooperative** that integrates **group buying and global project funding**.

🛒 1. System Components

◇ Token Structure

Token	Function	Purpose
Food-USD (Stablecoin)	Used for food purchases & funding projects	Ensures **stable pricing**, preventing speculation.
GroToken (Governance & Rewards)	Used for **voting, staking, and earning discounts**	Encourages **long-term participation & fair decision-making**.

◇ Key Features

- **Group Buying Smart Contracts** → Allows members to **pool funds for bulk food purchases**.
- **Decentralized Project Funding** → Community votes to **fund local and global food sustainability initiatives**.
- **Staking & Rewards** → Members earn **GroTokens for participation**, ensuring **engagement and liquidity stability**.
- **Governance DAO** → **GroToken holders vote** on food purchases, supplier selection, and funding allocations.

📜 2. Smart Contract Design

🛒 Group Buying Contract (Solidity Example)

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

contract GroupBuy {
    struct GroupOrder {
        address[] participants;
        uint totalCollected;
        uint targetAmount;
        bool completed;
    }

    mapping(uint => GroupOrder) public orders;
    uint public orderCounter;
```

```
    function createGroupBuy(uint _targetAmount) public {
        orderCounter++;
        orders[orderCounter] = GroupOrder(new address[](0), 0, _targetAmount,
false);
    }

    function joinGroupBuy(uint _orderId) public payable {
        require(msg.value > 0, "Must send Food-USD to participate.");
        require(orders[_orderId].completed == false, "Group buy already
completed.");

        orders[_orderId].participants.push(msg.sender);
        orders[_orderId].totalCollected += msg.value;

        if (orders[_orderId].totalCollected >= orders[_orderId].targetAmount) {
            orders[_orderId].completed = true;
            executeGroupBuy(_orderId);
        }
    }

    function executeGroupBuy(uint _orderId) internal {
        address supplier = 0xSupplierAddress; // Placeholder
        payable(supplier).transfer(orders[_orderId].totalCollected);
    }
}
```

Voting & Governance Smart Contract

```
contract FoodCoopDAO {
    struct Proposal {
        string description;
        uint voteCount;
        bool executed;
    }

    mapping(uint => Proposal) public proposals;
    mapping(address => bool) public hasVoted;
    uint public proposalCounter;

    function createProposal(string memory _description) public {
        proposalCounter++;
        proposals[proposalCounter] = Proposal(_description, 0, false);
    }

    function voteOnProposal(uint _proposalId) public {
        require(hasVoted[msg.sender] == false, "Already voted.");
        proposals[_proposalId].voteCount++;
        hasVoted[msg.sender] = true;
    }
```

```
    function executeProposal(uint _proposalId) public {
        require(proposals[_proposalId].voteCount > 10, "Not enough votes.");
        proposals[_proposalId].executed = true;
    }
}
```

📊 3. Economic Model & Simulation

◇ Example: Group Buy Economics

- **Target Food Purchase:** 1,000 lbs of rice at $1.50/lb.
- **Minimum Buy-In Per User:** $15 (10 lbs minimum).
- **Participants Needed: 100 people** to reach the goal.
- **Savings vs. Retail: Retail price = $1.90/lb**, bulk buy gets it for **$1.50/lb** (21% savings).
- **Bonus:** Each participant **earns 5 GroTokens** for participating.

◇ Example: Governance & Project Funding

- **Community votes to fund a permaculture farm project**.
- **Required Funding:** $5,000.
- **Total Voters:** 500 people.
- **Vote Weighting:** Each voter **stakes at least 10 GroTokens**.
- **Funding Release:** The farm **receives the grant** if **300+ people vote YES**.

📦 4. Next Steps & Development Plan

Phase 1: Validate the Math

- **Run simulations** with different participation levels.
- **Ensure sustainability** of rewards & food pricing.

Phase 2: Develop Token & DAO System

- **Decide on blockchain infrastructure** (Ethereum L2, Solana, or Polygon).
- **Create Food-USD & GroToken smart contracts** for transactions & governance.
- **Test a small-scale DAO for voting & funding decisions**.

Phase 3: Launch Prototype

- **Pilot a group-buying event using a simple web platform**.
- **Test governance with a small food-related project vote**.
- **Refine logistics, partnerships, and onboarding process**.

🚀 Conclusion

This **Food Cooperative + Group Buying + DAO Funding** model is designed to: ✔ **Provide affordable food access through bulk buying.**

- ✔ **Empower communities with democratic decision-making.**
- ✔ **Support regenerative agriculture & sustainability.**
- ✔ **Create a transparent, fair economic system for food security.**

Would you like further refinement of the **mathematical models, governance rules, or tokenomics strategy** before implementation? 🚀