**Blockchain-Based Drought Index Insurance Platform**

## 1. Introduction

This project presents a blockchain-based drought index insurance platform specifically designed for maize farmers in drought-prone regions of Malawi. The platform leverages smart contracts and Chainlink’s decentralized oracle network to automate and secure insurance processes including policy creation, subscription, and automatic payouts triggered by real-world weather data.

### Why Malawi?

Malawi is a country highly dependent on agriculture, with maize being the staple crop. Its farmers are severely affected by unpredictable rainfall patterns and frequent droughts. By offering index-based insurance using rainfall data, we aim to provide a fast, transparent, and tamper-proof way to support these communities.

## 2. Scope

* Target Audience: Maize farmers in rural areas of Malawi.
* Weather Index: Rainfall data measured in millimeters.
* Trigger: Drought conditions based on rainfall falling below predefined thresholds.
* Blockchain: Ethereum (testnet) or Polygon for scalability.
* Oracle Layer: Chainlink Functions & Chainlink Automation.

## 3. Core Features

### 3.1 User Roles

#### ● Farmer:

* Register and submit farm information (name, contact, location, farm size).
* Subscribe to insurance policies after admin approval.
* Pay premiums via smart contract.
* View active policies and claim history.
* Receive automated payouts when rainfall data triggers policy conditions.

#### ● Admin:

* Approve or reject farmer policy subscription requests.
* Manage and define new policy templates.
* Monitor system funds, payouts, and drought reports.
* Pause/Resume system in emergencies.

## 4. Smart Contract Features

* **Policy Creation:** Farmer details, premium amounts, and payout thresholds are stored immutably on-chain.
* **Payout Triggering:** Chainlink Functions fetch off-chain weather data from public APIs (e.g., OpenWeatherMap). This data is returned to the smart contract, which compares the reported rainfall value to the drought threshold. If the value is below the threshold, the payout is automatically triggered.
* **Automation:** Chainlink Automation is used to periodically call the weather-checking function (e.g., daily), ensuring that payout conditions are monitored consistently without user input.
* **Event Logging:** Emits events like PolicyCreated, PayoutIssued, WeatherChecked for frontend integration and transparency.
* **Emergency Pause:** Admin can pause all contract functions such as new subscriptions, premium payments, and automated payouts in the event of a system failure or suspicious activity. This helps prevent further damage or exploitation until the issue is resolved.

## 5. Chainlink-Specific Features

### 5.1 Chainlink Functions

* Fetches real-time rainfall data from external weather APIs using a JavaScript function executed off-chain within the Chainlink DON.
* The function validates and processes the data before sending it to the smart contract.
* Developers may implement checks such as:
  + Ensuring the rainfall field exists and is a valid number.
  + Ignoring out-of-range or extreme values.
  + Falling back to defaults if the API fails.
  + (Optional) Comparing multiple API sources for consistency.
* The smart contract uses the validated rainfall metric to decide whether a payout should be triggered. This logic is fully custom and does not rely on predefined Chainlink Data Feeds.

### 5.2 Chainlink Automation

* Periodically executes the rainfall-checking function (e.g., once per day) using Chainlink’s decentralized automation network.
* Ensures that the contract regularly checks if rainfall levels fall below the predefined drought threshold during the policy period. If so, a payout is automatically triggered.
* The automation is responsible for scheduling and executing the rainfall check automatically based on time intervals, ensuring consistent evaluation of drought conditions.
* Frequency can be configured to align with the insurance policy duration or specific phases of the maize crop season. For example:
  + During sensitive growth phases like flowering and grain filling, rainfall checks can occur daily to capture critical drought conditions.
  + In less sensitive phases, checks may be reduced (e.g., weekly) to optimize gas usage.
* This flexibility helps balance accuracy in drought detection with efficient resource usage, even for developers unfamiliar with agricultural science.

## 6. Technical Architecture

* **Frontend:** Next.js + Wagmi for wallet interaction.
* **Smart Contract:** Solidity + Foundry.
* **Oracle Layer:** Chainlink Functions (API fetch) + Chainlink Automation (scheduling).
* **Blockchain Network:** Ethereum Sepolia / Polygon Mumbai (testnet).
* **Storage:**
  + On-chain: Stores minimal essential data like policy ID, owner, thresholds, status. These values are required by the contract for logic execution and payout calculation.
  + IPFS: Full policy metadata (e.g., farmer’s full name, contact info, detailed farm description, optional documents) is stored in IPFS due to its large size and non-essential nature for smart contract execution. This approach helps reduce gas costs and keeps the blockchain state optimized. The IPFS CID (content identifier) is stored in the smart contract, enabling anyone to fetch the metadata if needed.
  + NFT: Each policy is minted as an ERC-721 NFT representing the insured contract. It allows traceability and verifiable ownership.

## 7. Future Enhancements

* Support for other crops and disaster indices (floods, pests).
* NFT-based policy certificates for traceability and future secondary features like trading or transferring ownership.
* Decentralized governance via DAO: Farmers and stakeholders can vote on policy parameters, thresholds, and contract upgrades using governance tokens or staked NFTs. DAO mechanisms would be implemented via open frameworks like OpenZeppelin Governor or Snapshot integration.

## 8. Security Considerations

* Role-based access control for admin functions.
* Data validation and fallback in oracle response.
* Smart contract audits and test coverage.
* Emergency pause mechanism for halting all sensitive operations.

## 9. Summary

This platform showcases the power of combining blockchain, smart contracts, and decentralized oracle networks to provide a transparent and efficient insurance system. It empowers vulnerable communities through automated risk management and sets a foundation for scalable agricultural protection across developing nations.

Project Name Suggestion: **“AgriLink” or “RainSure”**