📘 REGENEDAG — FULL PROJECT DOCUMENTATION

Prepared by: Mustapha Ismail Ogunleye (Mioha)

Project Name: RegenDAG — Decentralized Humanitarian & Climate Relief Protocol

Prepared by: Mustapha Ismail Ogunleye (Mioha)

Emails:

mustymioha0107@gmail.com

mustaphaismail0107@gmail.com

Version: 1.0

Date: November 2025

2. Project Overview

RegenDAG is a decentralized humanitarian relief and climate-resilience distribution system built on the BlockDAG Awakening Testnet.

The purpose of RegenDAG is to eliminate fraud, increase transparency, and ensure fair and efficient distribution of disaster relief funds using blockchain, DAO governance, and automated smart contracts.

3. Problem Statement

Traditional humanitarian aid suffers from:

Fraudulent claims

Double payments

Corruption

Manual dependency

No audit trail

Delayed distribution

No household-level tracking

High admin cost

These issues reduce the effectiveness of aid and prevent assistance from reaching those who genuinely need it.

4. Solution Summary

RegenDAG solves these challenges by offering:

On-chain transparency

Automated smart contracts

Tokenized aid (REGEN)

DAO-controlled verification

Household-level mapping

Fraud prevention engine (AliefVerification)

Instant and fair payouts

Immutable proof of distribution

5. System Architecture

RegenDAG uses a four-contract modular architecture:

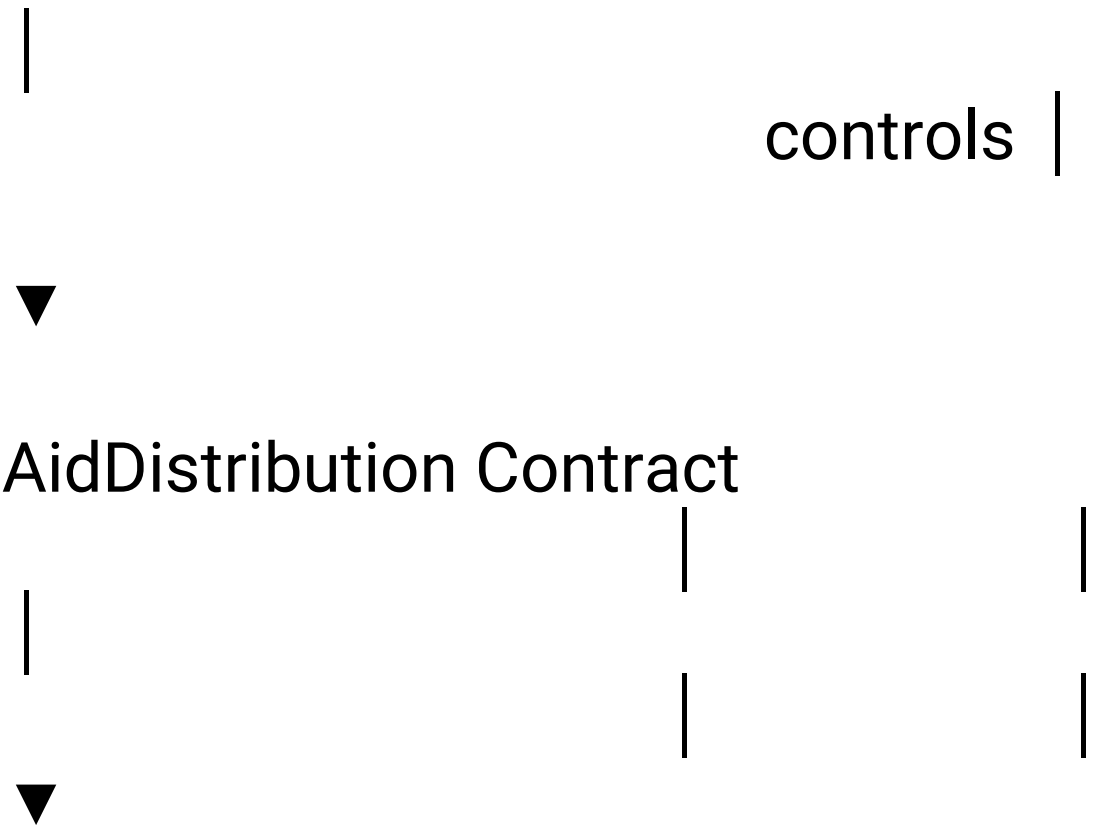
RegenToken.sol — ERC20 token for aid payouts

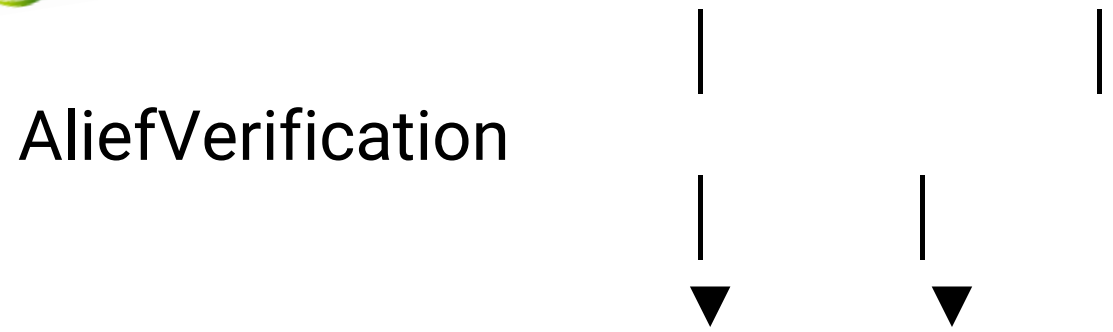
AliefVerification.sol — Fraud/eligibility engine

AidDistribution.sol — Registers, verifies, and pays beneficiaries

RegenDAO.sol — DAO owner for system governance

Architectural Diagram





---

6. Smart Contract Architecture

6.1 RegenToken.sol

Implements ERC20 standard

Mintable by owner

Used for relief fund transfers

6.2 AliefVerification.sol

Stores flagged accounts

Ensures eligibility

Prevents fraud

6.3 AidDistribution.sol

Registers beneficiaries

Verifies identity

Prevents household double-claims

Transfers REGEN tokens

6.4 RegenDAO.sol

Owns and controls AidDistribution

Executes registration, verification & payouts

7. Technical Components

Languages & Libraries

Solidity 0.8.20

OpenZeppelin v5

Hardhat

Ethers.js v6

Node.js 22

Ganache (local testing)

BlockDAG Testnet

Tools

Hardhat for compiling/deploying

Ganache for local blockchain

BlockDAG IDE (alternative deployment)

GitHub for source control

8. User Workflows

8.1 Beneficiary Registration

1. DAO registers wallet + householdId

2. System stores beneficiary data

3. Emits event Registered

8.2 Verification

1. DAO verifies with verifyBeneficiary

2. Updates verification status

8.3 Aid Payout

1. DAO calls releaseAid

2. All eligibility checks run

3. Token transfer is executed

4. Emits AidReleased

9. Smart Contract Functions Explained

RegenDAO.sol

setAidDistribution(address) ′n links main distribution contract

daoRegister(address,uint256) ′n register beneficiary

daoVerify(address,bool) ′n verify beneficiary

daoRelease(address,uint256) ′n release aid

AidDistribution.sol

register(address,uint256)

verifyBeneficiary(address,bool)

releaseAid(address,uint256)

setAlief(address)

AliefVerification.sol

setFlag(address,bool) ’n flag suspect accounts

isEligible(address,uint) ’n returns fraud check result

RegenToken.sol

mint(address,uint)

ERC20 functions inherited

10. Security Model

✔ Access Control

Only DAO has admin rights.

✔ Reentrancy Protection

Payout function uses nonReentrant.

✔ Household Integrity

One householdId ’n One payout.

✔ Immutable Logs

Events generate on-chain audit trails.

✔ Future Audits

Before mainnet, contracts will undergo a full audit.

11. Deployment Guide

11.1 Hardhat Deployment

Install:

npm install

Compile:

npx hardhat compile

Deploy:

npx hardhat run scripts/deploy_all.mjs --network ganache

11.2 Ganache Deployment

1. Open Ganache

2. Copy Private Key

3. Add to .env

4. Run deploy script

5. Deployment addresses appear in terminal

11.3 BlockDAG Deployment (when RPC stable)

Paste contracts into BlockDAG IDE

Compile

Deploy to Awakening Testnet

Check deployment at bdagscan.com

12. Folder Structure

```
RegenDAG/
├── contracts/
│   ├── RegenToken.sol
│   ├── AliefVerification.sol
│   ├── AidDistribution.sol
│   └── RegenDAO.sol
├── scripts/
│   └── deploy_all.mjs
├── docs/
│   └── architecture-diagram.png
(future)
├── .env
├── .env.example
├── hardhat.config.js
├── package.json
└── README.md
```

13. Testing Guide

Test the following:

✔ Registration

✔ Verification

✔ Release Aid

✔ Fraud flagging

✔ Household double-claim prevention

✔ Token transfer

Use Hardhat tests for automation (Wave 3+).

14. Future Enhancements (Roadmap)

Wave 3

Frontend interface, dashboards, wallet connect.

Wave 4

AI/Oracle integration, governance voting.

Wave 5

Localization, optimization, documentation.

Wave 6

Final demo, public testnet launch.

15. Glossary

REGEN Token — Token used for aid distribution
DAO — Decentralized governance body
Beneficiary — Aid recipient
Household ID — Family identifier
Eligibility Check — Fraud prevention logic
AliefVerification — Fraud detection contract

16. Appendices

Appendix A: Contract Addresses

(To be added after final deployment)

Appendix B: Architecture Diagram

(To be added to PDF)

Appendix C: Deployment Logs

(From Hardhat or Ganache)