EDUCATIONAL CROWDFUNDING DAPP

BY

Yerassyl A, Shynbergen Y, Erasyl U

Project Purpose & Overview

Objective: To build a transparent, decentralized crowdfunding platform (DApp) on the Ethereum blockchain.

Key Goals:

- Eliminating intermediaries in fundraising.
- Ensuring full transparency of contributions.
- Implementing an automated reward system using custom tokens.

Network: Exclusively deployed on **Ethereum Sepolia Testnet.**

Technical stack

Smart Contracts: Solidity (v0.8.20).

Development Framework: Hardhat (for compilation, deployment, and testing).

Frontend: HTML5, CSS3, JavaScript.

Blockchain Library: Ethers.js (to connect the UI with smart contracts).

Wallet Integration: MetaMask.

System Architecture

Decentralized Approach: No central database; all campaign data and funds are managed by smart contracts.

Components:

Crowdfunding Contract: Logic for managing campaigns.

Reward Token Contract: ERC-20 token for contributors.

Web Interface: Direct interaction via Provider (MetaMask).

Smart Contract: Crowdfunding.sol

Core Logic:

struct Campaign: Stores title, goal, deadline, amount raised, and status.

createCampaign(): Allows any user to start a new fundraiser.

contribute(): Accepts ETH and triggers the reward token minting.

finalizeCampaign(): Closes the campaign once the goal is met.

crowdfinding.sol

```solidity
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.20;

import "@openzeppelin/contracts/token/ERC20/ERC20.sol";

contract RewardToken is ERC20 {
    address public owner;

    event OwnershipTransferred(address indexed previousOwner, address indexed newOwner);

    constructor() ERC20("CrowdReward", "CRT") {
        owner = msg.sender;
        emit OwnershipTransferred(address(0), owner);
    }

    modifier onlyOwner() {
        require(msg.sender == owner, "Only owner");
        _;
    }

    function transferOwnership(address newOwner) external onlyOwner {
        require(newOwner != address(0), "Zero address");
        emit OwnershipTransferred(owner, newOwner);
        owner = newOwner;
    }

    function mint(address to, uint256 amount) external onlyOwner {
        _mint(to, amount);
    }
}
```

Smart Contract: RewardToken.sol (ERC-20)

Token Name: RewardToken (CRT).

Standard: OpenZeppelin ERC-20 implementation.

Security Feature (Access Control): * The contract uses a restricted minting mechanism.

- **Only** the Crowdfunding contract has the permission to mint new CRT tokens when a contribution is made.

rewardtoken.sol

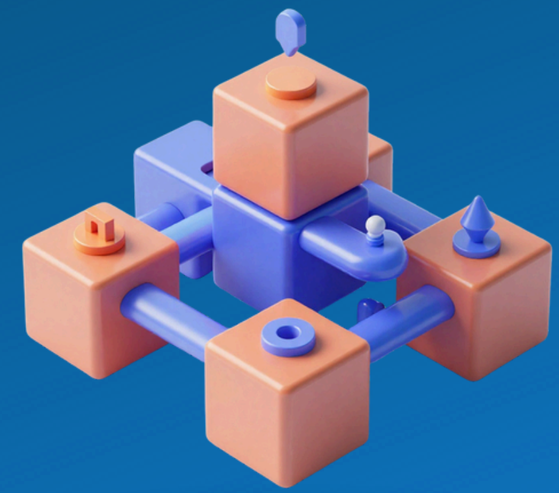
```
function setRewardToken(address _tokenAddress) external onlyOwner {
    require(_tokenAddress != address(0), "Zero address");
    require(address(rewardToken) == address(0), "Token already set");
    rewardToken = RewardToken(_tokenAddress);
    emit RewardTokenSet(_tokenAddress);
}

function createCampaign(
    string memory _title,
    uint256 _goal,
    uint256 _durationInDays
) external {
    require(_goal > 0, "Goal must be > 0");
    require(_durationInDays > 0, "Duration must be > 0");
    require(bytes(_title).length > 0, "Title required");

    uint256 deadline = block.timestamp + (_durationInDays * 1 days);

    campaigns.push(
        Campaign({
            title: _title,
            creator: payable(msg.sender),
            goal: _goal,
            collectedAmount: 0,
            deadline: deadline,
            finalized: false
        })
    );

    emit CampaignCreated(campaigns.length - 1, msg.sender, _title, _goal, deadline);
}

function contribute(uint256 _campaignId) external payable {
    require(_campaignId < campaigns.length, "Invalid campaignId");

    Campaign storage campaign = campaigns[_campaignId];
    require(block.timestamp <= campaign.deadline, "Campaign ended");
    require(!campaign.finalized, "Campaign finalized");
    require(msg.value > 0, "Send some ETH");

    contributions[_campaignId][msg.sender] += msg.value;
    campaign.collectedAmount += msg.value;

    uint256 rewardAmount = 0;

    if (address(rewardToken) != address(0)) {
        rewardAmount = msg.value * 100;
        rewardToken.mint(msg.sender, rewardAmount);
    }

    emit Contributed(_campaignId, msg.sender, msg.value, rewardAmount);
}
```

MetaMask & Blockchain Interaction

Connection: The DApp requests permission to access the user's wallet via window.ethereum.

Network Validation: The application checks if the user is connected to Sepolia (Chain ID 11155111) to prevent loss of funds.

Transaction Flow: Every action (creating/donating) requires a cryptographic signature from the user.



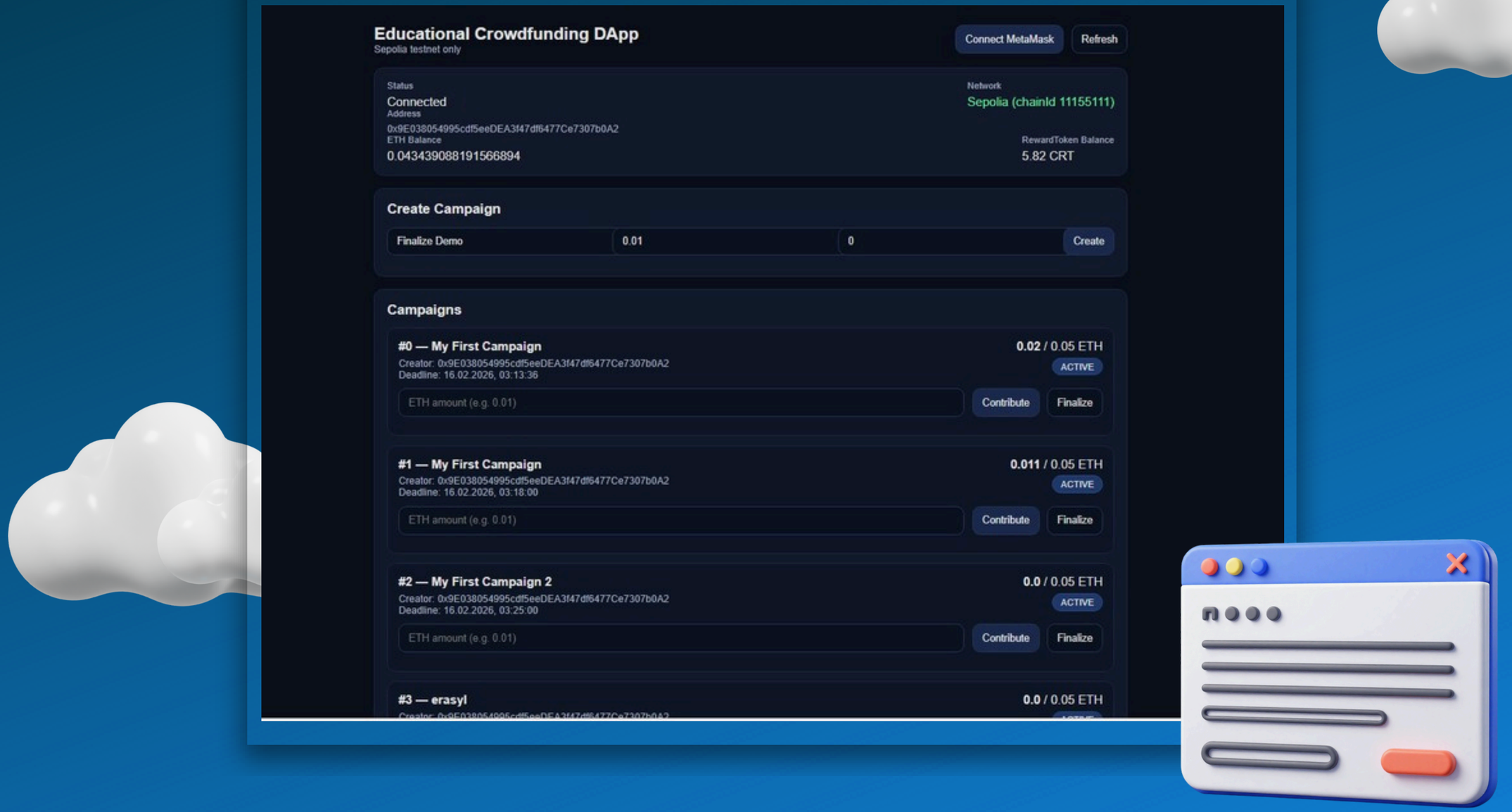
Frontend Implementation

User Experience:

- Real-time display of active campaigns.
- Dynamic progress bars for each funding goal.
- Dashboard showing user's ETH balance and CRT token balance.

Integration: Using Ethers.js to listen for blockchain events and update the UI.

frontend


Educational Crowdfunding DApp
Sepolia testnet only
Connect MetaMask
Refresh
Status
Connected
Address
0x9E038054995cdf5eeDEA3f47df6477Ce7307b0A2
ETH Balance
0.043439088191566894
Network
Sepolia (chainId 11155111)
RewardToken Balance
5.82 CRT
Create Campaign
Finalize Demo
0.01
0
Create
Campaigns
#0 — My First Campaign
0.02 / 0.05 ETH
Creator: 0x9E038054995cdf5eeDEA3f47df6477Ce7307b0A2
Deadline: 16.02.2026, 03:13:36
ACTIVE
ETH amount (e.g. 0.01)
Contribute
Finalize
#1 — My First Campaign
0.011 / 0.05 ETH
Creator: 0x9E038054995cdf5eeDEA3f47df6477Ce7307b0A2
Deadline: 16.02.2026, 03:18:00
ACTIVE
ETH amount (e.g. 0.01)
Contribute
Finalize
#2 — My First Campaign 2
0.0 / 0.05 ETH
Creator: 0x9E038054995cdf5eeDEA3f47df6477Ce7307b0A2
Deadline: 16.02.2026, 03:25:00
ACTIVE
ETH amount (e.g. 0.01)
Contribute
Finalize
#3 — erasyl
0.0 / 0.05 ETH

Deployment & Verification

- **Hardhat Scripts:** Used for automated deployment to Sepolia.
- **Contract Verification:** Both contracts are verified on Etherscan.
- **Why it matters:** Transparency allows anyone to read the source code and verify the logic independently.

Conclusion & Results

- **Functional Requirements Met:** Successful campaign creation, ETH contribution, and token distribution.
- **Security:** Implemented "on-chain" data storage and immutable transaction history.
- **Summary:** The project demonstrates a fully functional DApp that leverages the core benefits of blockchain: trust, security, and decentralization.

Thank you for listening!