Decentralized Voting System - Report

Final Exam - Blockchain Technologies 1

Team

Sagi Alpyssov
Aigerim Askarova
Sabira Kurbankhozha

Introduction

This project implements a decentralized crowdfunding and voting system on an Ethereum test network.

The main goal was to create a platform where users can create campaigns, contribute ETH, vote on campaign options, and receive reward tokens, all while maintaining transparency and security via smart contracts. The system operates entirely on a test network, without using real cryptocurrency, and demonstrates blockchain principles in practice.

Each campaign supports two voting options, and contributors can vote once per campaign. Contributions trigger minting of ERC-20 Reward Tokens proportional to the ETH contributed. The frontend provides real-time feedback on contributions, votes, and token balances.

2. System Architecture

The project is built using three main layers to handle data, security, and the user interface.

Smart Contracts

Two contracts were developed: CrowdFundingVoting.sol and RewardToken.sol.

RewardToken.sol is a standard ERC-20 token contract. Its mint function can only be called by the crowdfunding system to ensure controlled issuance of reward tokens:

rewardToken.mint(msg.sender, msg.value * 100);

After deployment, ownership of the token is transferred to the crowdfunding system:

await rewardToken.transferOwnership(await crowdfunding.getAddress());

This guarantees that tokens are only minted in response to valid contributions.

CrowdFundingVoting.sol manages campaign creation, contributions, voting, and finalization. Each campaign stores:

- Title
- Funding goal
- Deadline
- Total contributions
- Voting options (two candidates)
- Vote counts per option
- Record of contributors and whether they have voted

The smart contract ensures:

- Only valid campaigns can be created
- Only contributors can vote
- Duplicate voting is prevented
- Deadlines are enforced
- Campaigns are finalized automatically to prevent further contributions or votes after the deadline

Voting triggers token minting only if the user has contributed, reinforcing participation incentives.

Frontend Application

The frontend, built with JavaScript and ethers.js v6, allows users to:

- Connect MetaMask
- Create campaigns with a title, funding goal, duration, and two voting options
- Contribute ETH to campaigns
- Vote for one of the two options

View current vote counts, campaign details, contributions, and Reward Token balances

The interface updates automatically after each confirmed transaction. For example, after a contribution, the ETH balance, token balance, and total raised are updated immediately. Vote buttons are dynamically disabled for contributors who have already voted or for campaigns that are finalized.

A campaign listing feature allows users to see all existing campaigns and their IDs, preventing errors when contributing or voting.

MetaMask Integration

MetaMask serves as the bridge between users and the Ethereum network:

Detects MetaMask via window.ethereum
Requests account access for secure signing of transactions
Verifies that the user is on the local Hardhat network (chain ID 31337) to avoid sending transactions to the wrong network
All contributions, votes, and campaign creations are signed securely

Smart Contract Logic

Campaign Creation

Stores campaign title, goal, duration, and two voting options

require(_options.length == 2, "Two options required");

Contributions

Contributors send ETH to a campaign
Must contribute > 0 ETH
Reward tokens are minted proportional to contributions

Voting

Only contributors can vote
One vote per campaign per contributor

Vote counts are updated on-chain

Finalization

After the deadline, campaigns can be finalized
Funds are sent to the creator if the funding goal is reached

Frontend-to-Blockchain Interaction

Wallet Connection: Users must approve account access

Transaction Handling: All blockchain calls are asynchronous; UI updates after confirmation

State Synchronization: Vote counts, total raised, and token balances are updated automatically

Validation Feedback: Users are prevented from voting multiple times or contributing after a campaign ends

Campaign Listing: Users can view all campaigns with IDs, titles, goals, and total raised to prevent mistakes

Deployment and Test ETH

Deployment uses Hardhat local network:

1. Deploy RewardToken.sol
2. Deploy CrowdFundingVoting.sol, passing the token address
3. Transfer token ownership to the crowdfunding contract

Frontend addresses are updated with the deployed contract addresses.

Users connect MetaMask to the local Hardhat network (chain ID 31337). Pre-funded accounts provide test ETH for contributions.

For public testnets (e.g., Sepolia), test ETH can be obtained from faucets. All interactions are confined to test networks.

Testing

Automated and manual tests cover:

- Campaign creation
- Contribution logic and reward token minting
- Voting restrictions and vote counting
- Prevention of double voting
- Deadline enforcement
- Finalization of campaigns

Time manipulation using Hardhat confirmed correct behavior for deadlines.

Manual testing of the frontend verified dynamic updates for contributions, votes, balances, and campaign listings.

Conclusion

The Decentralized Crowdfunding and Voting System provides a secure and transparent way to run crowdfunding campaigns with optional voting The Decentralized Voting System provides a secure and transparent platform for elections, combining smart contracts, ERC-20 incentives, and a responsive frontend. Users can create elections, vote securely, and receive participation tokens, all on test networks. Throughout development, careful attention was given to validation, state synchronization, and user feedback. The system is fully tested, reliable, and ready for further extension. The project demonstrates a practical implementation of blockchain

principles, combining backend contract logic with frontend usability.