CROWDFUNDING DAPP ON ETHEREUM TESTNET

BLOCKCHAIN TECHNOLOGIES 1 — FINAL EXAMINATION PROJECT

Students:
Nauryzbay Tomiris,Aisulu Azimkhan,Medina Klyumova
Instructor: Sayakulova Zarina

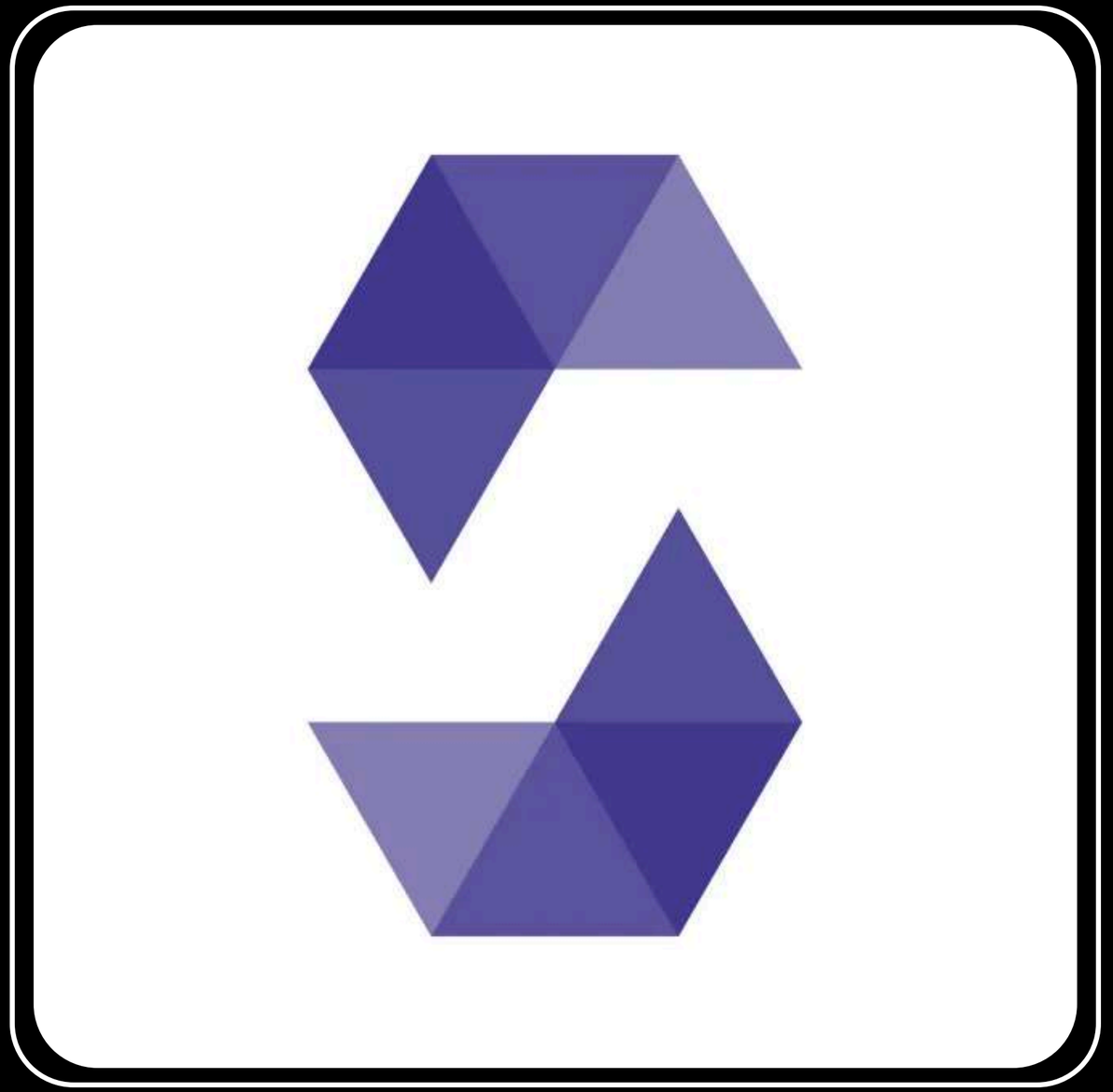
ASTANA IT UNIVERSITY

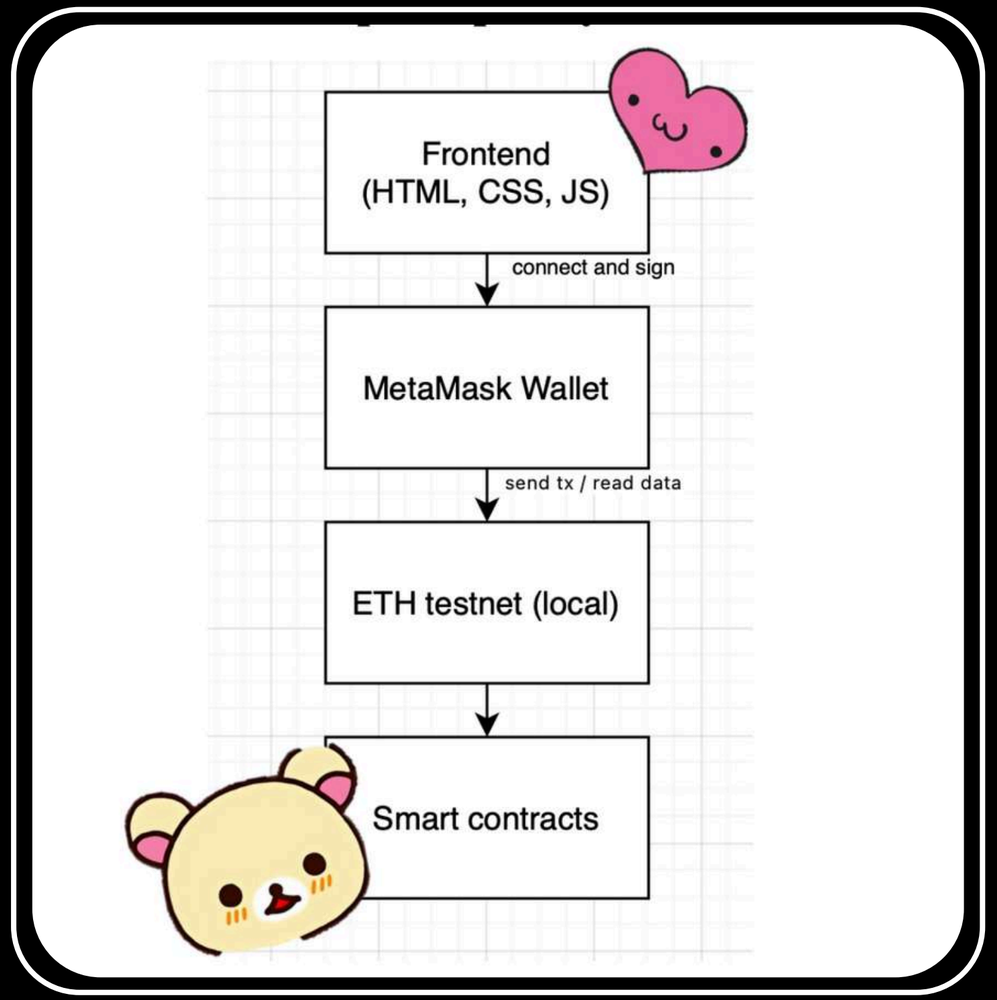
PROJECT GOAL

- Build a decentralized crowdfunding application
- Replace centralized intermediaries with smart contracts
- Enable campaign creation and ETH contributions
- Use Ethereum test networks for safe experimentation

TECHNOLOGY STACK

- Solidity — smart contract development
- JavaScript — frontend logic and interaction
- MetaMask — wallet connection and transaction signing
- Ethereum (Testnet / Localhost) — blockchain execution
- Hardhat — development and deployment framework





SYSTEM ARCHITECTURE

WEB3 ARCHITECTURE

→ METAMASK

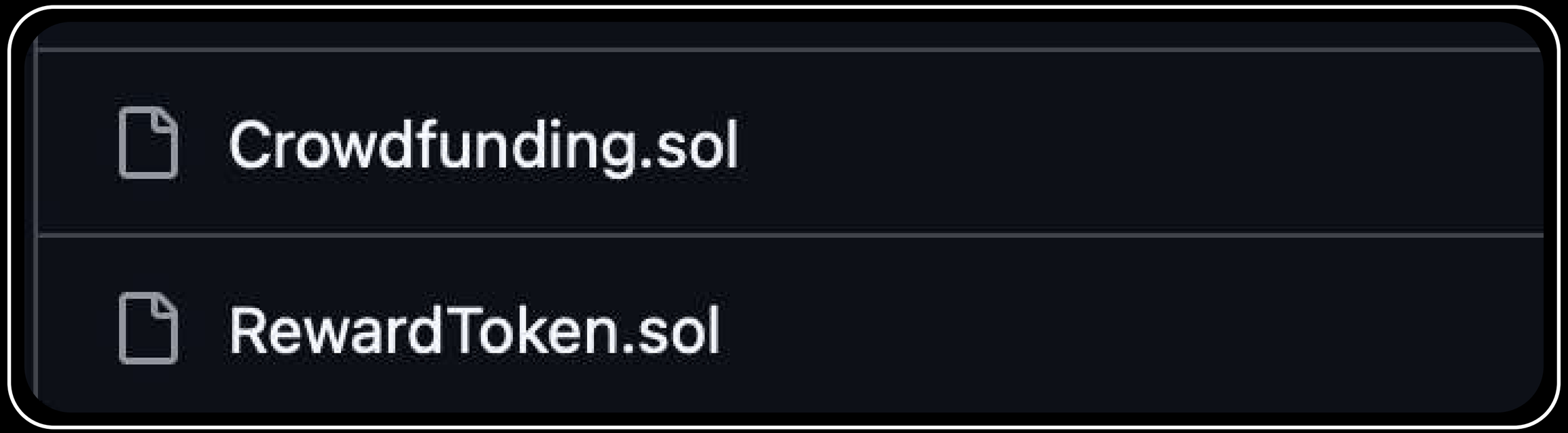
→ ETHEREUM TESTNET

→ Smart Contracts

SMART CONTRACTS OVERVIEW

Two smart contracts are used:

- Crowdfunding Contract
- ERC-20 Reward Token Contract



CAMPAIGN DATA STRUCTURE

EACH CAMPAIGN STORES:

- FUNDING GOAL
- DEADLINE
- OWNER ADDRESS
- TOTAL RAISED AMOUNT
- FINALIZATION STATUS

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

interface IRewardToken {
    function mint(address to, uint256 value) external returns (bool);
}

/// @title Crowdfunding - simple crowdfunding with reward token
contract Crowdfunding {
    struct Campaign {
        string title;
        uint256 goal;
        uint256 deadline;
        address owner;
        uint256 totalRaised;
        bool finalized;
    }

    uint256 public campaignCount;
    mapping(uint256 => Campaign) public campaigns;
    mapping(uint256 => mapping(address => uint256)) public contributions;

    IRewardToken public immutable rewardToken;

    // 1000 tokens per 1 ETH (1e18 wei)
    uint256 public constant TOKENS_PER_ETH = 1000;
```

CAMPAIGN DATA STRUCTURE

On-chain storage model for crowdfunding campaigns and contributions

```
struct Campaign {
    string title;
    uint256 goal;
    uint256 deadline;
    address owner;
    uint256 totalRaised;
    bool finalized;
}

uint256 public campaignCount;
mapping(uint256 => Campaign) public campaigns;
mapping(uint256 => mapping(address => uint256)) public contributions;

IRewardToken public immutable rewardToken;
```

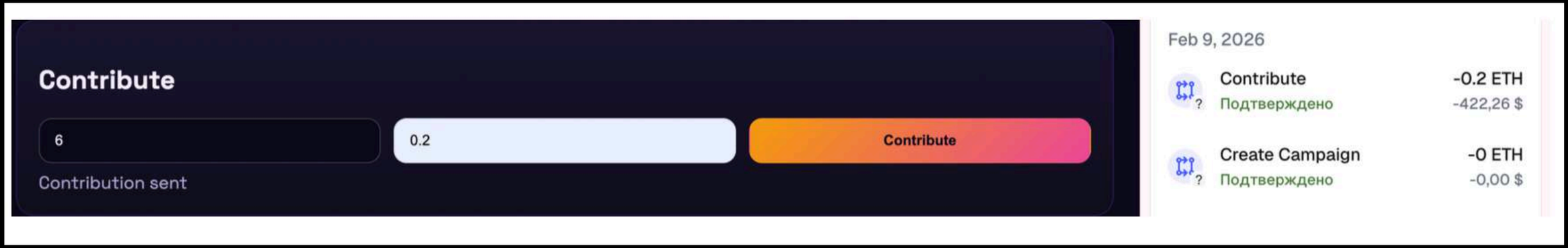
CAMPAIGN CREATION LOGIC

New campaigns are created on-chain with a funding goal and a time-based deadline.

```
function createCampaign(string calldata title, uint256 goal,
uint256 durationSeconds) external returns (uint256) {
    require(bytes(title).length > 0, "TITLE_EMPTY");
    require(goal > 0, "GOAL_ZERO");
    require(durationSeconds > 0, "DURATION_ZERO");

    uint256 id = ++campaignCount;
    uint256 deadline = block.timestamp + durationSeconds;

    campaigns[id] = Campaign({
        title: title,
        goal: goal,
        deadline: deadline,
        owner: msg.sender,
        totalRaised: 0,
        finalized: false
    });

    emit CampaignCreated(id, msg.sender, title, goal, deadline);
    return id;
}
```

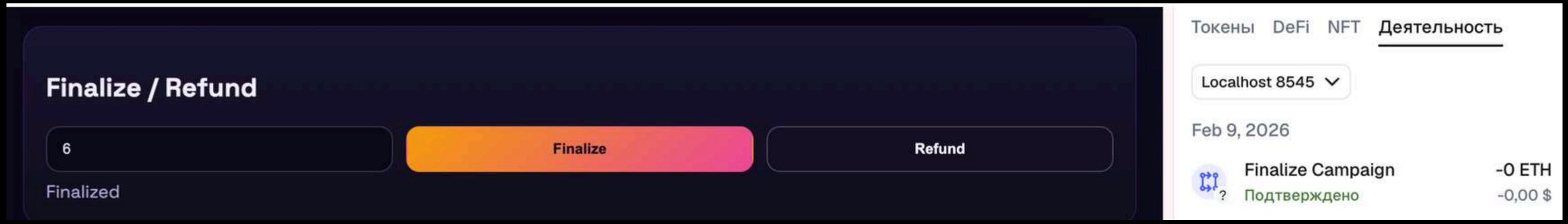
CONTRIBUTION AND REWARD LOGIC

Users send test ETH contributions via MetaMask. Each contribution is recorded on-chain and automatically triggers reward token minting.

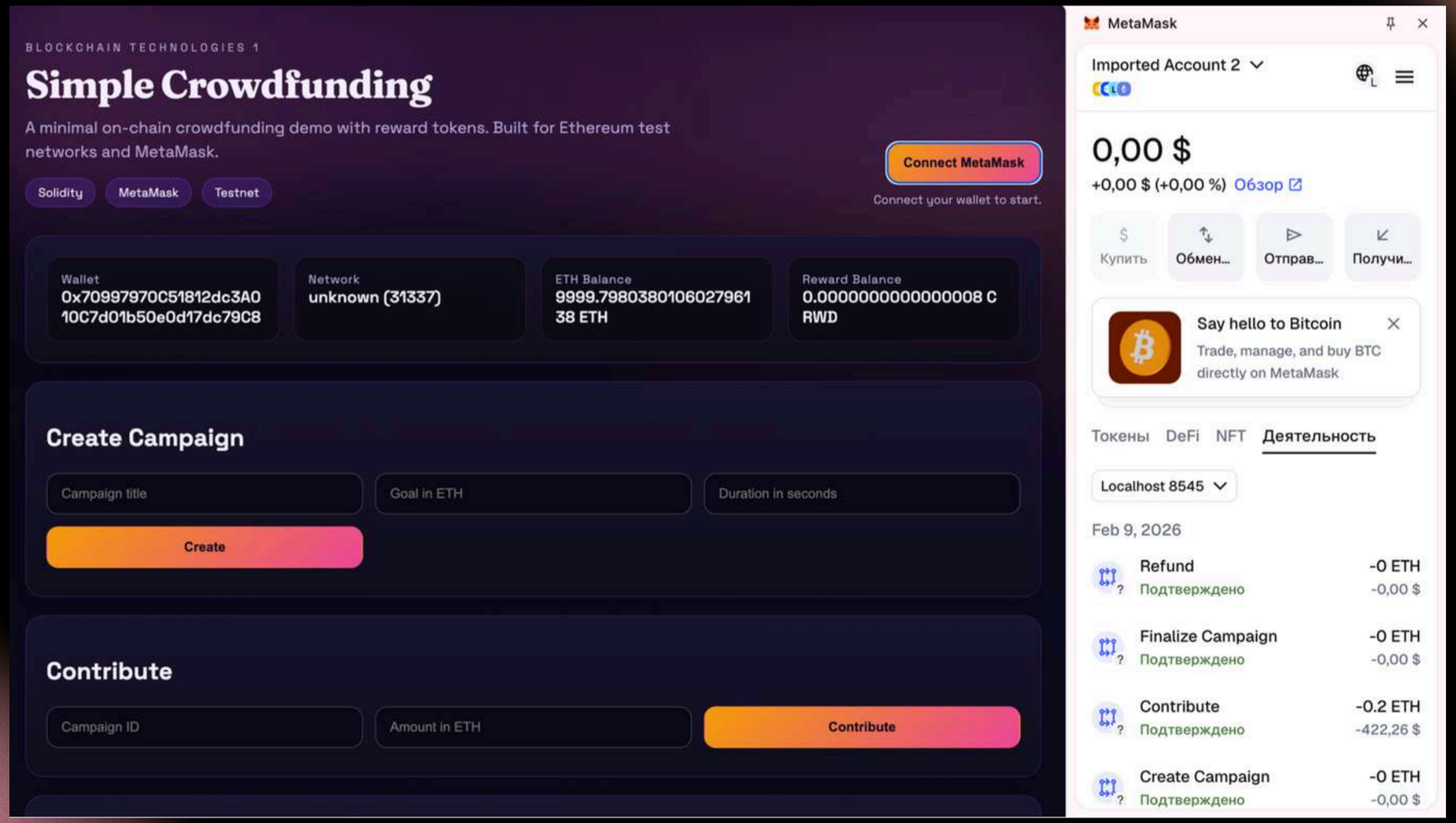


ERC-20 REWARD TOKEN

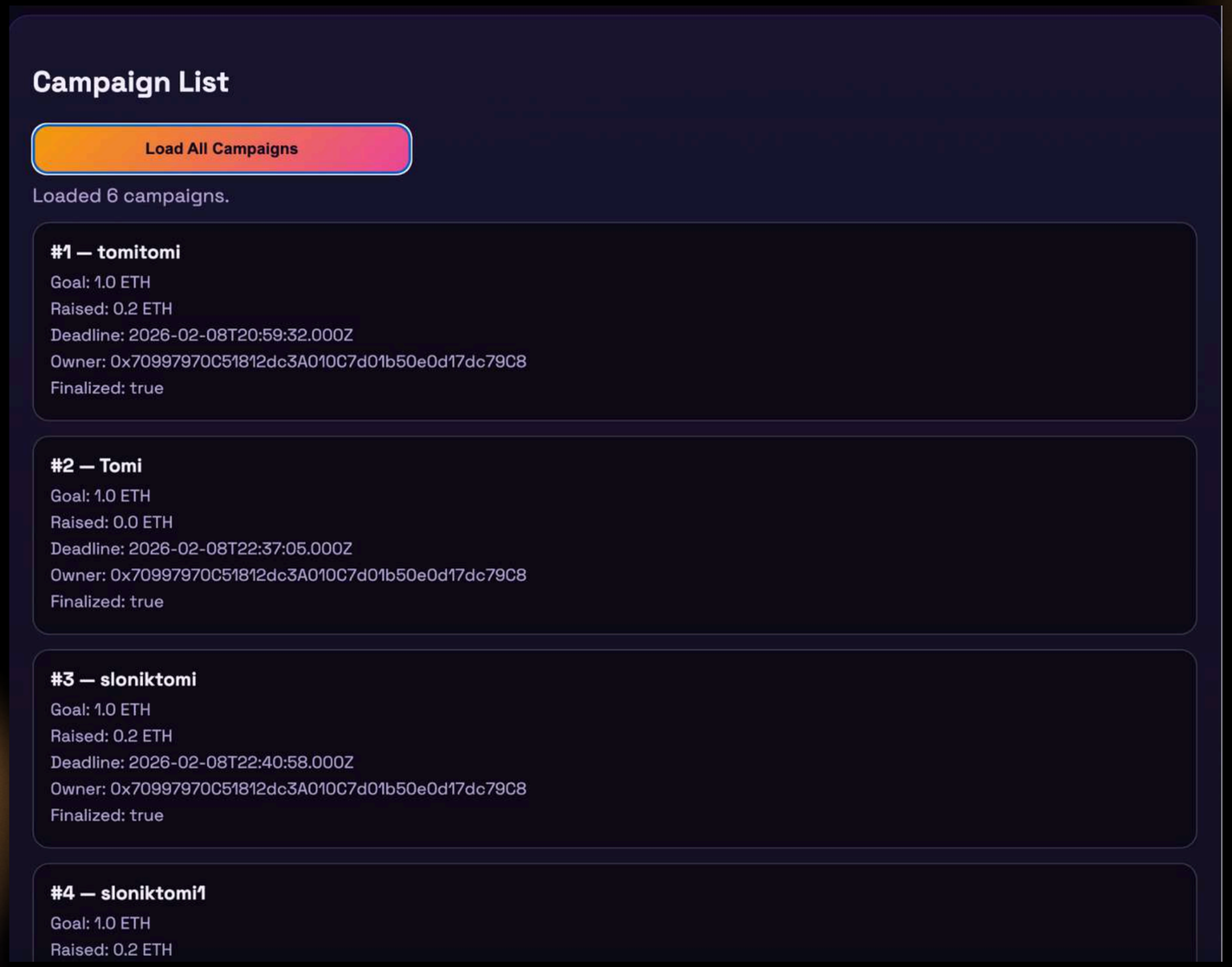
- Custom ERC-20 token is used as a reward mechanism
- Tokens are minted automatically during user contribution
- The token has no real monetary value
- Used only for educational and demonstration purposes



METAMASK INTEGRATION



FRONTEND INTERFACE

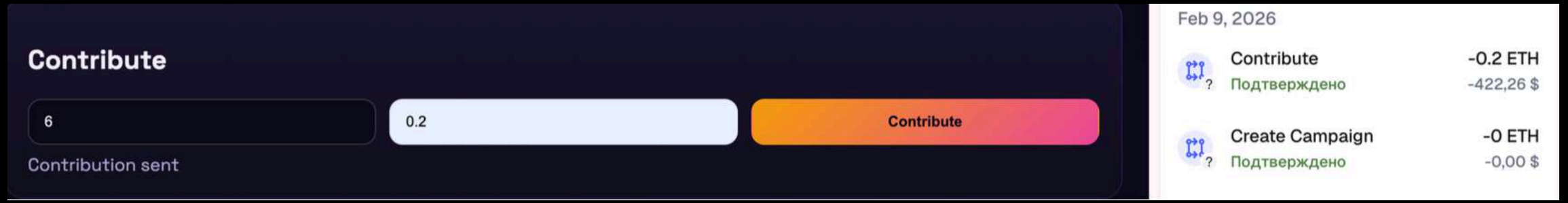


CAMPAIGN CREATION (DEMO)

This demo demonstrates how a user creates a crowdfunding campaign by specifying a title, funding goal, and duration. The transaction is sent via MetaMask and stored on the blockchain.

CONTRIBUTION RESULT (DEMO)

This demo shows a successful ETH contribution to an active campaign.
The transaction is signed and confirmed via MetaMask and recorded on the blockchain.



FINALIZATION & REFUND (DEMO)

After the campaign deadline, the campaign can be finalized.
If the funding goal is reached, funds are transferred to the campaign owner.
Otherwise, contributors can refund their ETH.

CONCLUSION

This project demonstrates a fully functional decentralized crowdfunding application. Smart contracts manage campaign logic, contributions, and refunds transparently. ERC-20 reward tokens illustrate tokenization concepts.

The application operates entirely on an Ethereum test network.