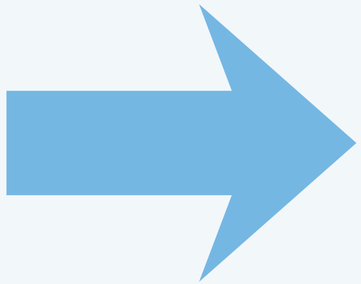
CROWDFUNDING MARKETPLACE

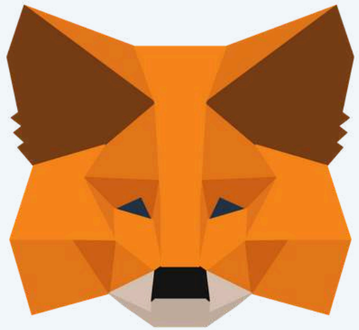
PRESENTATION

Presented By: Asylbek Abdibay, Tugelbayev Aidyn, Nugmash Bigali

PROJECT PURPOSE

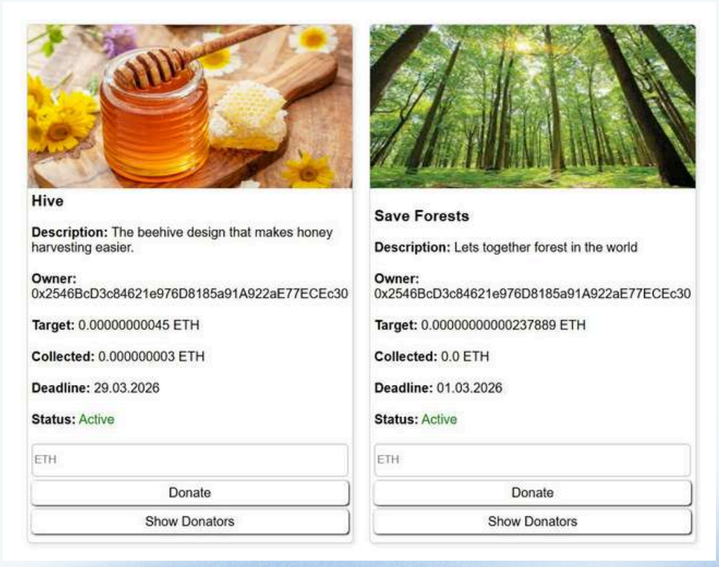
The purpose of this project is to demonstrate practical skills in blockchain technologies by developing a decentralized crowdfunding application. The project focuses on smart contracts, tokenization, MetaMask integration, and interaction with Ethereum test networks.

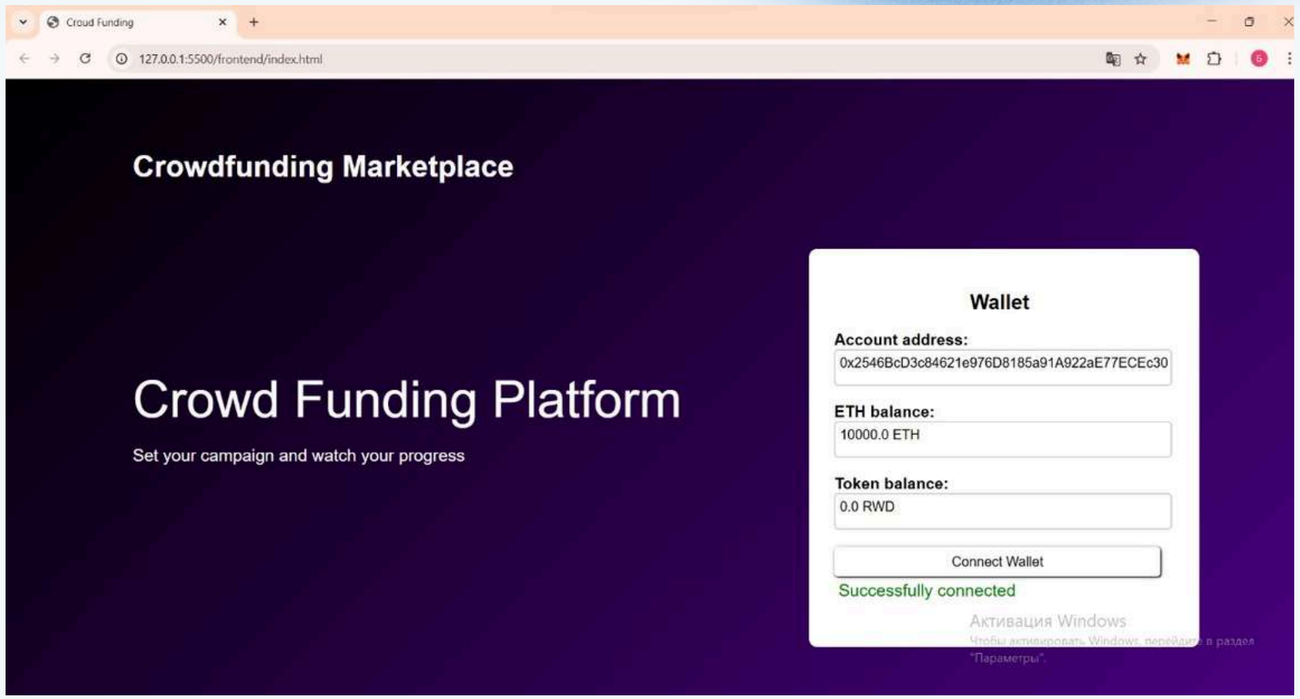




PROJECT OVERVIEW

The developed application is a decentralized crowdfunding platform that operates exclusively on an Ethereum test network.
Users can create crowdfunding campaigns, contribute test ETH, and receive internal ERC-20 reward tokens for participation.

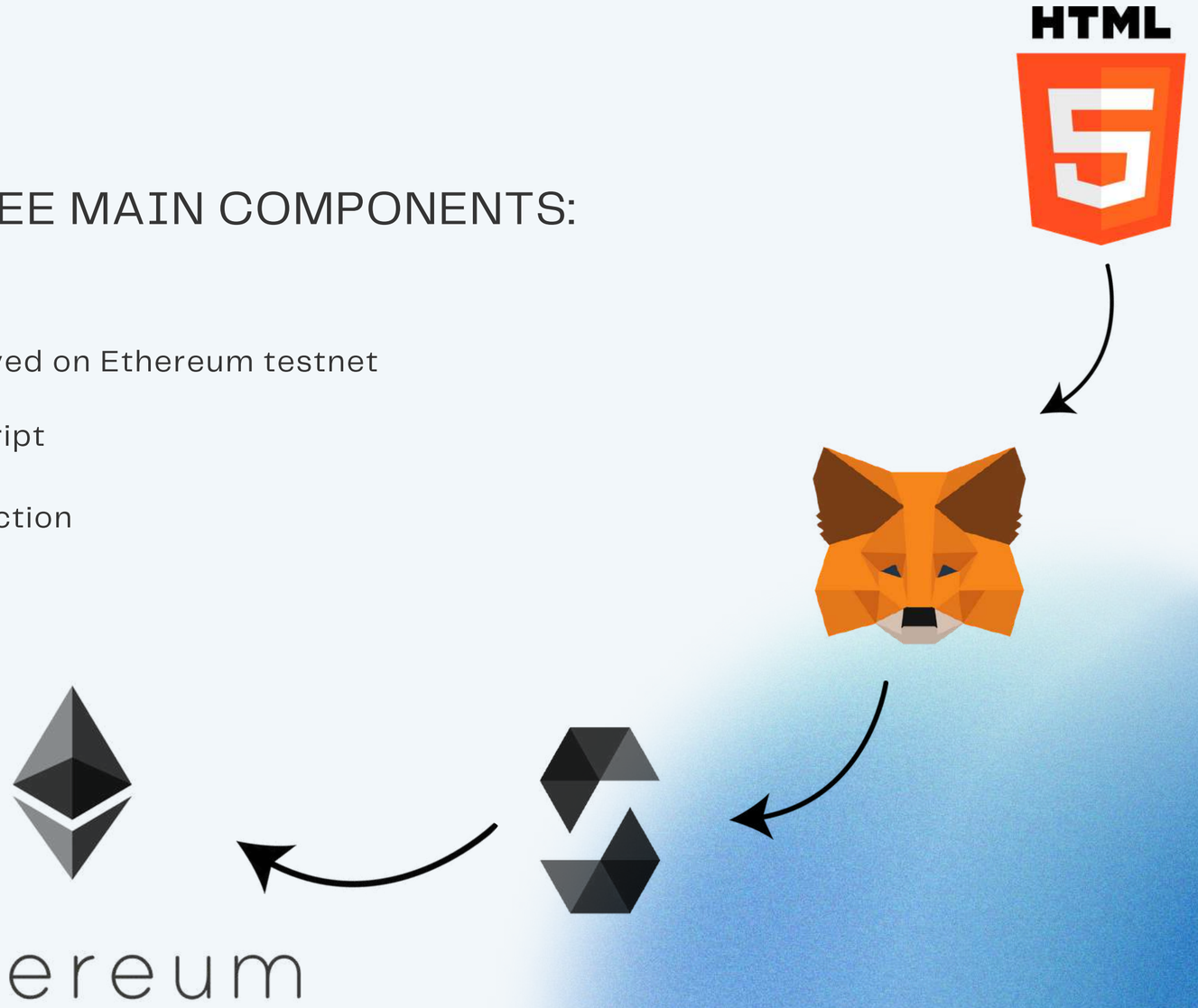




SYSTEM ARCHITECTURE

THE SYSTEM CONSISTS OF THREE MAIN COMPONENTS:

- Smart contracts written in Solidity and deployed on Ethereum testnet
- Frontend application developed using JavaScript
- MetaMask wallet for secure blockchain interaction



SMART CONTRACT FUNCTIONALITY

The smart contract provides the following features:

- Creation of crowdfunding campaigns with title, goal, and deadline
- Contribution of test ETH to active campaigns
- Tracking of individual user contributions
- Finalization of campaigns after the deadline

```
function donateToCampaign(uint256 _id) public payable {
    uint256 amount = msg.value;

    Campaign storage campaign = campaigns[_id];

    campaign.donators.push(msg.sender);
    campaign.donations.push(amount);

    (bool sent, ) = payable(campaign.owner).call{value: amount}("");

    if (sent) {
        campaign.amountCollected = campaign.amountCollected + amount;
    }

    reward_tokens.mint(msg.sender, msg.value * 100);
}
```

```
function createCampaign(address _owner,string memory _title,string memory _description,uint256 _target,uint256 _deadline,string
    require(_deadline > block.timestamp,"The deadline should be a date in the future.");

    Campaign storage campaign = campaigns[numberOfCampaigns];

    campaign.owner = _owner;
    campaign.title = _title;
    campaign.description = _description;
    campaign.target = _target;
    campaign.deadline = _deadline;
    campaign.amountCollected = 0;
    campaign.image = _image;

    numberOfCampaigns++;

    return numberOfCampaigns - 1;
}
```

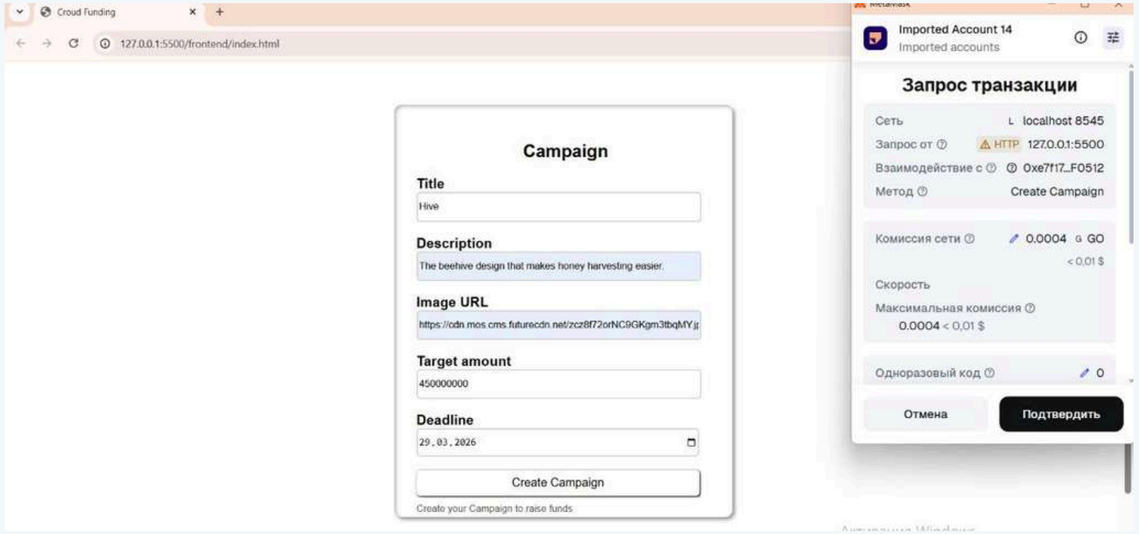
ERC-20 REWARD TOKEN

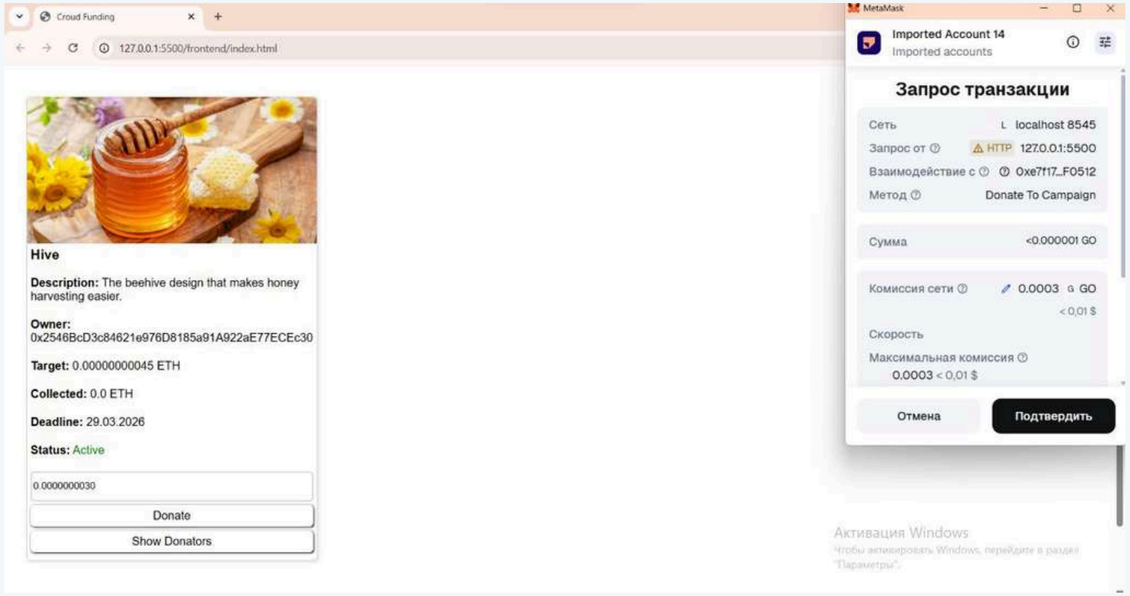
The project includes a custom ERC-20 token used as a reward mechanism.
Tokens are automatically minted when users contribute to a campaign.
The token has no real monetary value and is used only for educational purposes.

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

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

contract Reward_Tokens is ERC20,Ownable{

    constructor() ERC20("Reward Tokens","RWD") Ownable(msg.sender){}

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

METAMASK INTEGRATION

METAMASK IS USED FOR SECURE INTERACTION WITH THE BLOCKCHAIN. THE APPLICATION REQUESTS USER PERMISSION TO ACCESS WALLET ACCOUNTS, VERIFIES THE ACTIVE TEST NETWORK, AND EXECUTES TRANSACTIONS THROUGH METAMASK.

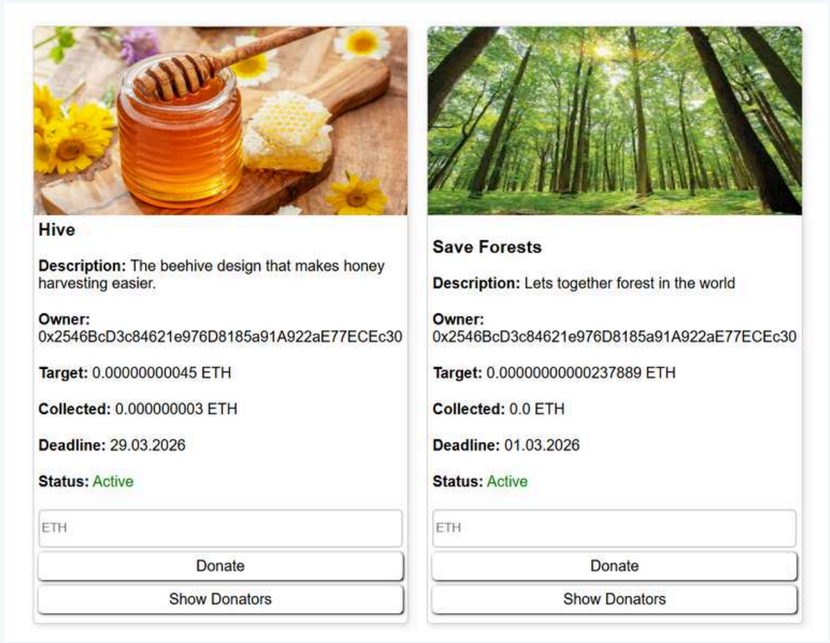


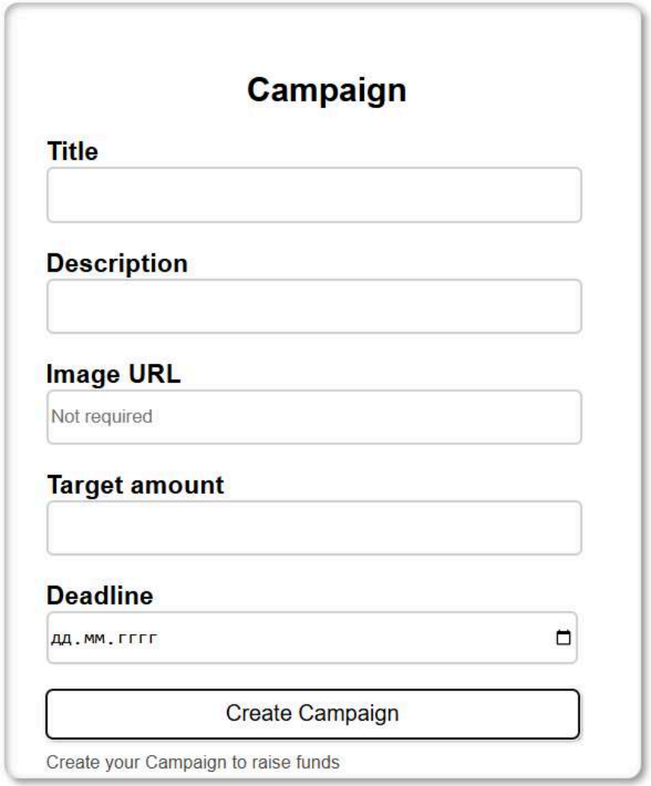


FRONTEND FEATURES

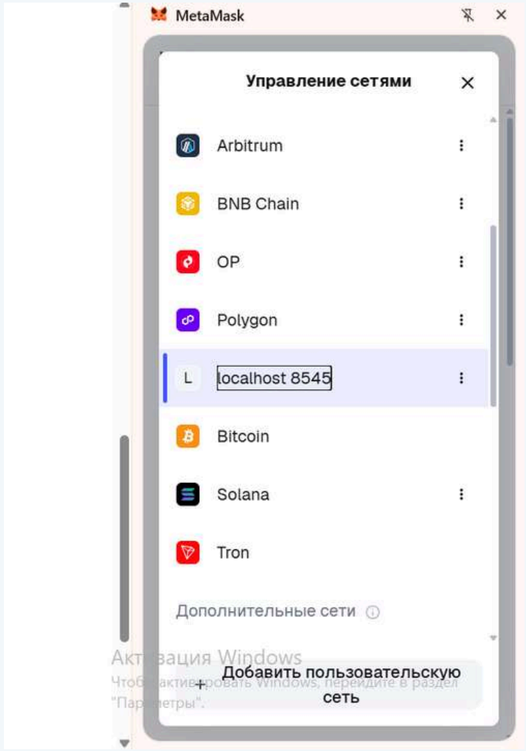
The frontend application allows users to:

- Connect their MetaMask wallet
- Create crowdfunding campaigns
- Contribute test ETH
- Monitor transaction status
- View ETH and reward token balances

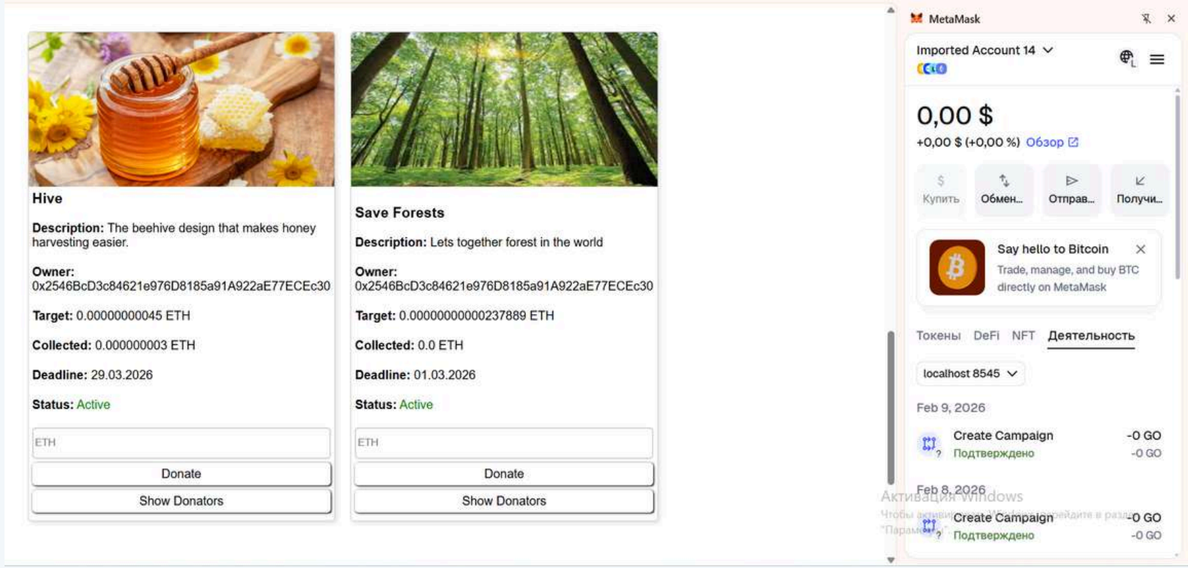




ETHEREUM TEST NETWORK

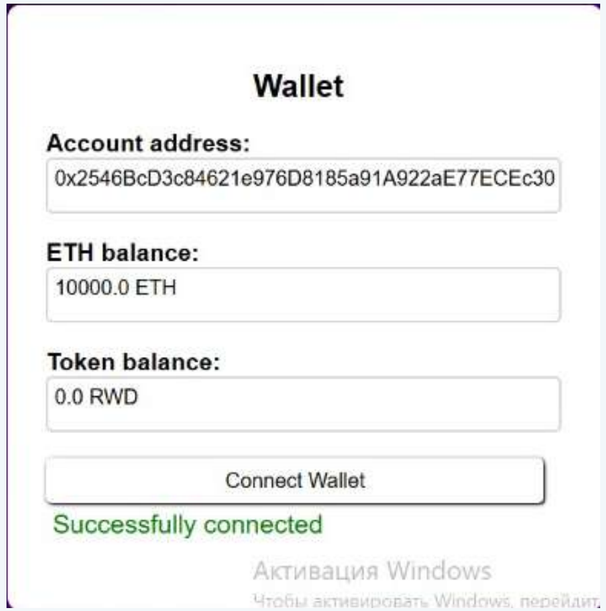


The application is deployed on an Ethereum test network (Hardhat).
Only free test ETH is used, and interaction with the Ethereum mainnet is strictly avoided.

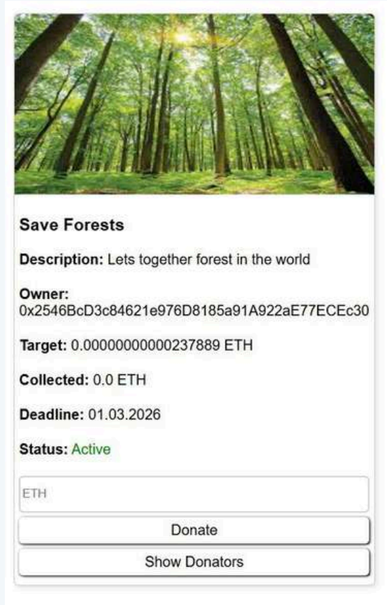


APPLICATION WORKFLOW

USER CONNECTS METAMASK WALLET:



USER CREATES A CROWDFUNDING CAMPAIGN



OTHER USERS CONTRIBUTE TEST ETH

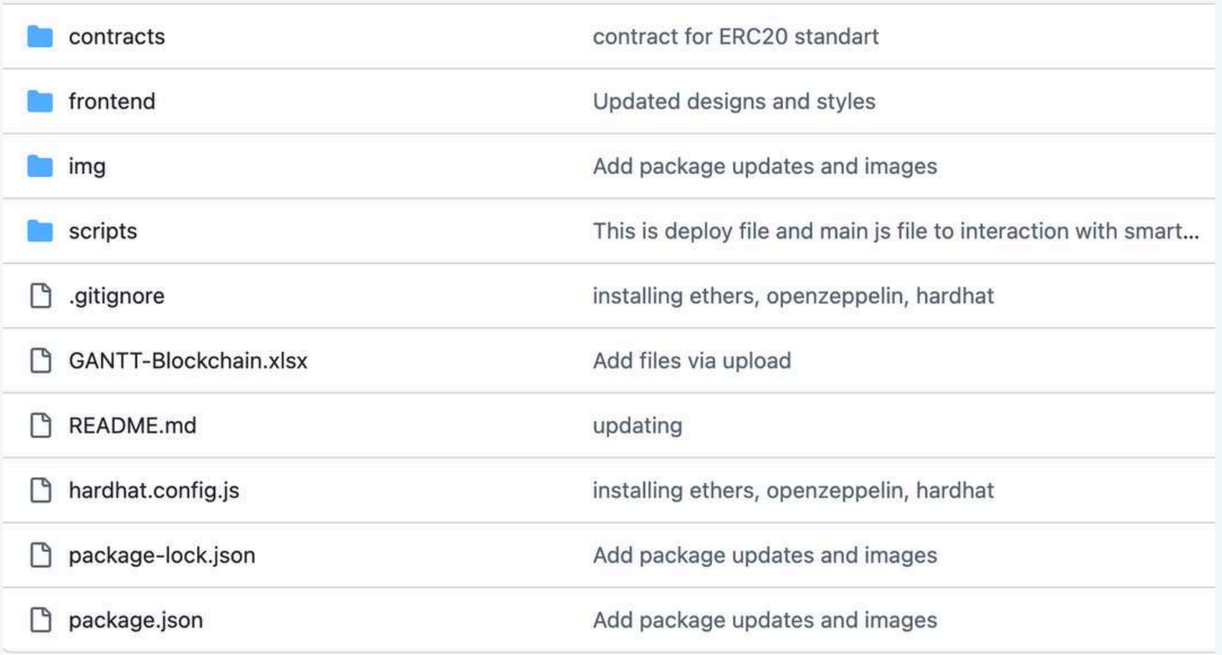
REWARD TOKENS ARE ISSUED AUTOMATICALLY

CAMPAIGN IS FINALIZED AFTER THE DEADLINE

REQUIREMENT COMPLIANCE

THE PROJECT FULLY SATISFIES ALL COURSE REQUIREMENTS:

- Ethereum test network usage
- Solidity smart contracts
- ERC-20 token implementation
- MetaMask integration
- Real blockchain interaction



contracts	contract for ERC20 standart
frontend	Updated designs and styles
img	Add package updates and images
scripts	This is deploy file and main js file to interaction with smart...
.gitignore	installing ethers, openzeppelin, hardhat
GANTT-Blockchain.xlsx	Add files via upload
README.md	updating
hardhat.config.js	installing ethers, openzeppelin, hardhat
package-lock.json	Add package updates and images
package.json	Add package updates and images

THANK YOU