

- DECENTRALIZED
CHARITY
CROWDFUNDING
PLATFORM

Our Team



NURZHAN ZHUMABEKOV



ABILKHAYIR SARSENBAY



ABYLAY ABDYKASSYMOV

Project Overview

This project is a decentralized charity crowdfunding platform developed as part of the Blockchain 1 final examination. The goal of the project is to demonstrate practical blockchain development skills, including smart contracts, frontend interaction, and MetaMask integration. The main idea of this project is to create a decentralized application that allows users to support charity campaigns directly through the blockchain. Instead of relying on a centralized platform, all rules and transactions are handled by smart contracts.

Problem & Motivation

Traditional crowdfunding platforms are centralized, which means users must trust a third party to manage funds correctly. This can lead to transparency and trust issues. Our project addresses this problem by using blockchain technology.

Project Goals

01

Raise public awareness about the importance of life insurance.

02

Expand market share across demographics.

03

Design strategies to boost product sales.

04

Enhance customer satisfaction through quality service.

Process

MARKET RESEARCH



Collect data on target audiences and competitors.

STRATEGY DEVELOPMENT



Create a comprehensive marketing plan to attract customers.

IMPLEMENTATION



Apply marketing strategies and evaluate their effectiveness.

Architecture

User → MetaMask → Frontend → Smart
Contracts → Ethereum Test Network

The system follows a decentralized architecture. The frontend only provides the interface, while MetaMask signs transactions and smart contracts execute all core logic on the Ethereum test network. No funds are stored on the frontend.

Smart Contracts

Abilkhayr was responsible for implementing all smart contracts using Solidity. This includes the crowdfunding contract, the ERC-20 reward token, and generating the ABI files that allow the frontend to interact with the blockchain.

```
5   contract CharityCrowdfunding {
6     struct Campaign {
7       address creator;
8       string title;
9       uint256 goalWei;
10      uint256 deadline;
11      uint256 totalRaisedWei;
12      bool finalized;
13      bool goalReached;
14    }
15
16
17    CharityToken public rewardToken;
18    uint256 public campaignCount;
19
20    mapping(uint256 => Campaign) public campaigns;
21    mapping(uint256 => mapping(address => uint256)) public contributions;
22
23    event CampaignCreated(
24      uint256 indexed id,
25      address indexed creator,
26      string title,
27      uint256 goalWei,
```

```
> contracts > 📄 CharityToken.sol > ...
-License-Identifier: MIT
solidity ^0.8.20;

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

t CharityToken is ERC20, Ownable {
  constructor(address initialOwner)
    ERC20("Charity Reward Token", "CHAR")
    Ownable(initialOwner)

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

Frontend

Decentralized Charity Crowdfunding Platform

Address: 0x90f7...b906 Wallet: GoChain Testnet GO Balance: 9873.9991 GO Your Contributions: 126.00 GO

Connect wallet: Connect

CREATE NEW CAMPAIGN

Title of campaign:

Need to collect ___ GO:

Must be collected within ___ days:

Create

CAMPAIGN LIST:

There are no campaigns

Refresh

CONTRIBUTE CAMPAIGNS:

Campaign ID:

Amount:

Contribute

CAMPAIGNS INFORMATION:

Total contributions: 0
Total campaigns: 0

Refresh info

TRANSACTION OUTCOMES:

[02:03:40] Wallet connected!

Abilkhayir developed the frontend part of the application. This includes the website interface, campaign display, input forms, and buttons that allow users to create campaigns and initiate contributions.

```
1 <!DOCTYPE html>
2 <html>
3
4 <head>
5   <title>
6     Decentralized Charity Crowdfunding Platform
7   </title>
8
9   <link rel="stylesheet" href="style.css">
10 </head>
11 <body>
12   <h1 id="heading">Decentralized Charity Crowdfunding Platform</h1>
13   <div class="wallet-container">
14     <h3>Connect wallet:</h3>
15     <button onclick="connectWallet()">Connect</button>
16   </div>
17
18   <div id="walletContainer">
19     <div class="walletContainerP">
20       <p>Address: <span id="walletAddress">Not connected</span></p>
21     </div>
22     <div class="walletContainerP">
23       <p>Wallet: <span id="network">Not connected</span></p>
24     </div>
25     <div class="walletContainerP">
26       <p>GO Balance: <span id="ethBalance">0</span></p>
27     </div>
28     <div class="walletContainerP">
29       <p>Your Contributions: <span id="tokenBalance">0</span></p>
30     </div>
31   </div>
```

```
1 const CROWDFUNDING_ADDRESS = "0xa513E6E4b8f2a923D98304ec87F64353C4D5C853";
2 const TOKEN_ADDRESS = "0x0165878A594ca255338adfa4d48449f69242Eb8F";
3
4 const crowdfundingABI = [
5   {
6     "inputs": [
7       { "name": "title", "type": "string" },
8       { "name": "goalWei", "type": "uint256" },
9       { "name": "durationSeconds", "type": "uint256" }
10    ],
11    "name": "createCampaign",
12    "outputs": [],
13    "stateMutability": "nonpayable",
14    "type": "function"
15  },
16  {
17    "inputs": [{ "name": "id", "type": "uint256" }],
18    "name": "contribute",
19    "outputs": [],
20    "stateMutability": "payable",
21    "type": "function"
22  },
23  {
24    "inputs": [{ "name": "id", "type": "uint256" }],
25    "name": "finalizeCampaign",
26    "outputs": [],
27    "stateMutability": "nonpayable",
28    "type": "function"
29  },
30  {
31    "inputs": [{ "name": "id", "type": "uint256" }],
```

STUDENT 3: METAMASK & INTEGRATION

```

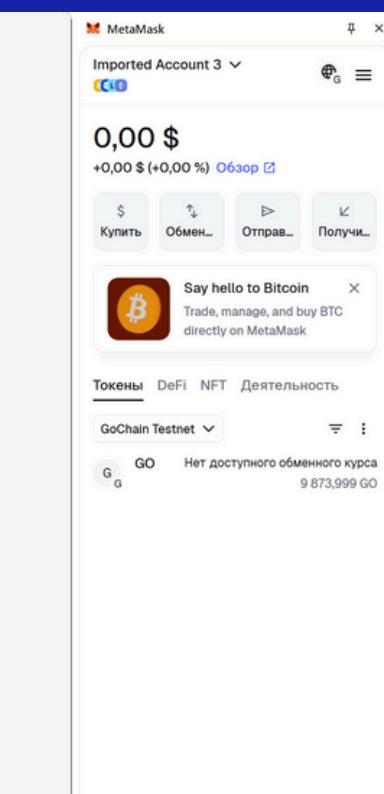
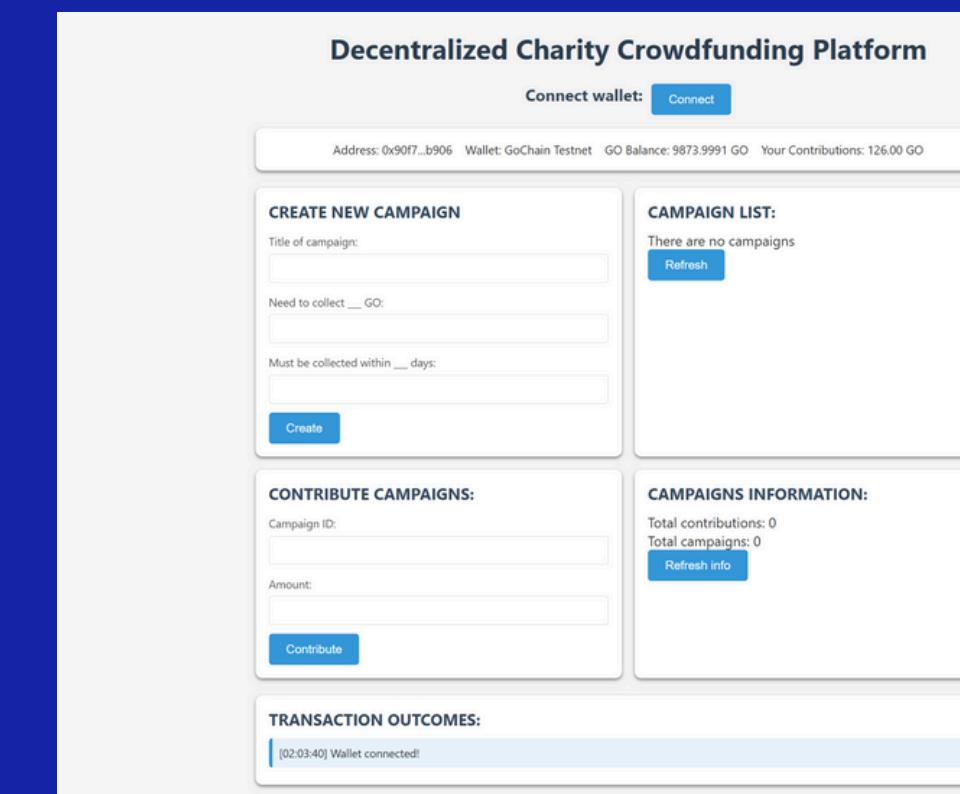
74 let provider, signer, userAddress, crowdfundingContract, tokenContract;
75
76 async function connectWallet() {
77   if (!window.ethereum) { alert("MetaMask not installed!"); return; }
78   try {
79     await window.ethereum.request({
80       method: 'wallet_switchEthereumChain',
81       params: [{ chainId: '0x7A69' }]
82   });
83 } catch (switchError) {
84   if (switchError.code === 4902) {
85     await window.ethereum.request({
86       method: 'wallet_addEthereumChain',
87       params: [
88         {
89           chainId: '0x7A69',
90           chainName: 'GoChain Testnet',
91           rpcUrls: ['http://127.0.0.1:8545'],
92           nativeCurrency: { name: 'GO', symbol: 'GO', decimals: 18 }
93         }
94       ]
95     });
96   }
97   provider = new ethers.BrowserProvider(window.ethereum);
98   const accounts = await provider.send('eth_requestAccounts', []);
99   userAddress = accounts[0];
100  signer = await provider.getSigner();
101
102  crowdfundingContract = new ethers.Contract(CROWDFUNDING_ADDRESS, crowdfundingABI, signer);
103  tokenContract = new ethers.Contract(TOKEN_ADDRESS, tokenABI, provider);
104
105}

```

```

101 crowdfundingContract = new ethers.Contract(CROWDFUNDING_ADDRESS, crowdfundingABI, signer);
102 tokenContract = new ethers.Contract(TOKEN_ADDRESS, tokenABI, provider);
103
104 document.getElementById("walletAddress").textContent = userAddress.slice(0, 6) + "..." + userAddress.slice(-4);
105
106 const network = await provider.getNetwork();
107 document.getElementById("network").textContent = "GoChain Testnet";
108
109 const balance = await provider.getBalance(userAddress);
110 document.getElementById("ethBalance").textContent = parseFloat(ethers.formatEther(balance)).toFixed(4) + " GO";
111
112 const count = await crowdfundingContract.campaignCount();
113 let totalContributed = 0;
114 for (let i = 0; i < Number(count); i++) {
115   const amount = await crowdfundingContract.contributions(i, userAddress);
116   totalContributed += amount;
117 }
118 document.getElementById("tokenBalance").textContent = parseFloat(ethers.formatEther(totalContributed)).toFixed(2) + " GO";
119
120 showStatus("Wallet connected!");
121
122 async function createCampaign() {
123   const title = document.getElementById("campaignTitle").value;
124   const goal = document.getElementById("campaignGoal").value;
125   const duration = document.getElementById("campaignDuration").value;
126
127   if (!title || !goal || !duration) { alert("Fill all fields"); return; }
128
129   showStatus("Creating campaign...");

```



NURZHAN WAS RESPONSIBLE FOR CONNECTING THE FRONTEND TO THE BLOCKCHAIN. THIS INCLUDES INTEGRATING METAMASK, CONFIGURING THE SCRIPT.JS FILE, HANDLING WALLET CONNECTION, AND IMPLEMENTING THE CONTRIBUTION LOGIC SO THAT TRANSACTIONS ARE SENT AND CONFIRMED THROUGH METAMASK.

Contribution Process

The screenshot shows a user interface for managing contributions. At the top, there's a header with 'Connect wallet:' and a 'Connect' button. Below it, a status bar displays: Address: 0x90f7...b906, Wallet: GoChain Testnet, GO Balance: 9873.9991 GO, and Your Contributions: 126.00 GO.

CREATE NEW CAMPAIGN: This section contains fields for 'Title of campaign:', 'Need to collect ___ GO:', and 'Must be collected within ___ days:', along with a 'Create' button.

CAMPAIGN LIST: This section lists three active campaigns:

- ID: 0 - y** Active
Goal: 6.0 GO | Raised: 6.0 GO
Deadline: 15.02.2026
- ID: 1 - charity** Active
Goal: 10.0 GO | Raised: 20.0 GO
Deadline: 19.02.2026
- ID: 2 - charity 2** Active
Goal: 100.0 GO | Raised: 100.0 GO
Deadline: 20.05.2026

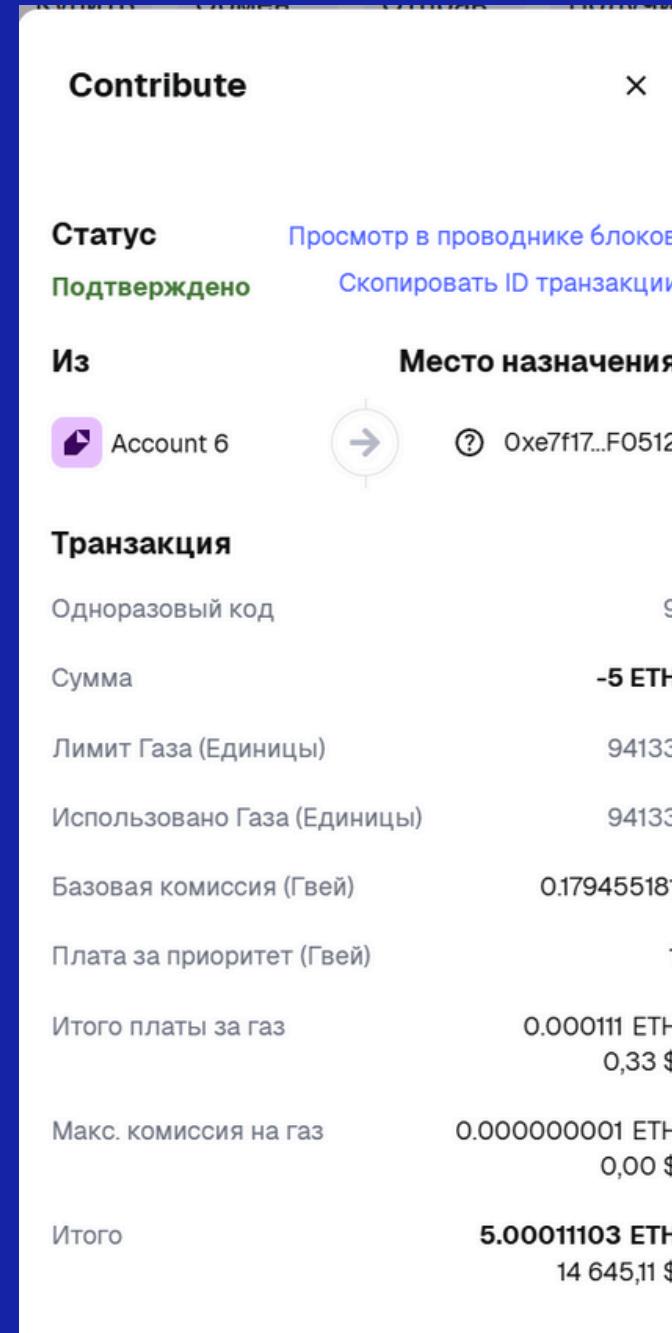
A 'Refresh' button is located below the campaign list.

CONTRIBUTE CAMPAIGNS: This section has fields for 'Campaign ID:' and 'Amount:', with a 'Contribute' button.

CAMPAIGNS INFORMATION: This section displays summary statistics: Total contributions: 126.0 GO and Total campaigns: 3, with a 'Refresh info' button.

THIS SLIDE SHOWS THE CONTRIBUTION PROCESS FROM THE USER'S PERSPECTIVE. WHEN A USER ENTERS AN AMOUNT AND CLICKS CONTRIBUTE, THE FRONTEND CALLS THE SMART CONTRACT FUNCTION AND CREATES A BLOCKCHAIN TRANSACTION.

Transaction Result



GoChain Testnet		
Feb 8, 2026		
	Contribute Подтверждено	-5 ETH -14 644,79 \$
	Create Campaign Подтверждено	-0 ETH -0,00 \$
	Contribute Подтверждено	-10 ETH -29 289,57 \$
	Create Campaign Подтверждено	-0 ETH -0,00 \$
	Contribute Подтверждено	-100 ETH -292 895,71 \$

Here we can see a successfully confirmed transaction. This confirms that the contribution was processed on the Ethereum test network and approved through MetaMask, demonstrating real blockchain interaction.

Conclusion & Compliance

In conclusion, the project fully complies with academic requirements. It operates on an Ethereum test network, uses only test ETH, and demonstrates smart contracts, frontend development, and MetaMask integration for educational purposes.

THANK YOU

FOR YOUR ATTENTION

REPOSITORY LINK:

[HTTPS://GITHUB.COM/NURZHAN-ZHUMABEKOV/BLOCKCHAIN-1-PROJECT/TREE/FEATURE/CONTRACTS-HARDHAT](https://github.com/nurzhan-zhumabekov/blockchain-1-project/tree/feature/contracts-hardhat)