

Final Project: Blockchain Technologies-1

Group members: Zarina Kerimbay

Akbota Bekturgan

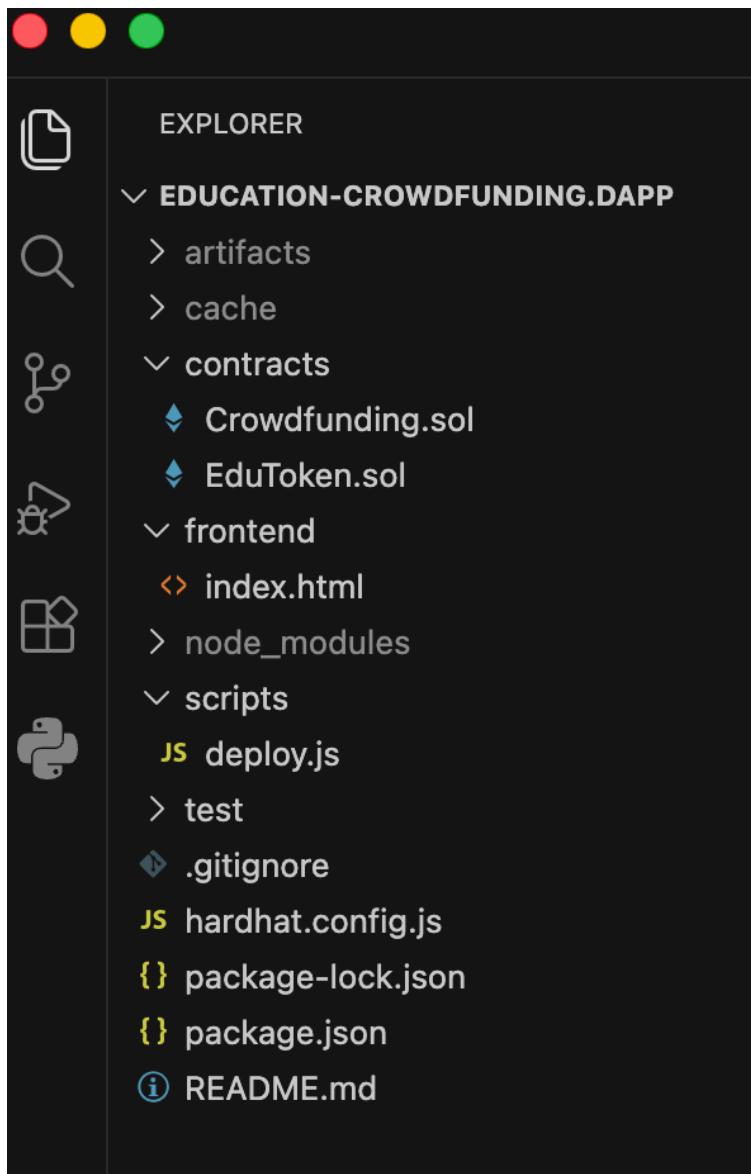
Rustam Saduakas

<https://github.com/bekturganakbota70-blip/education-crowdfunding.dapp.git>

Overview of Application Architecture

This section explains how the components of your system interact. A typical DApp follows a **Three-Tier Architecture**:

- **Frontend (The Client):** Built with frameworks like React, Vue, or Next.js. It handles the UI and user input.
- **Provider/Wallet (The Bridge):** Tools like MetaMask or WalletConnect act as the signer and the gateway to the blockchain.
- **Smart Contracts (The Backend):** Written in Solidity (or similar) and deployed on a blockchain (e.g., Ethereum Sepolia). This serves as your "immutable database" and logic layer.



Design and Implementation Decisions

The system was designed with the following considerations:

- Core logic is implemented in **smart contracts** to ensure transparency and immutability.
- The contracts are kept **modular and minimal** to reduce gas costs.
- A **test network** is used instead of Ethereum mainnet to allow safe testing.
- MetaMask was selected for wallet integration due to its wide adoption and ease of use.
- The frontend does not store sensitive data; all critical data is retrieved directly from the blockchain.

The screenshot shows a code editor interface with a dark theme. On the left is an Explorer sidebar containing project files: index.html, Crowdfunding.sol (selected), EduToken.sol, README.md, and deploy.js. The main pane displays the Solidity code for the 'Crowdfunding' contract. The code defines a struct 'Campaign' with fields owner, title, goal, deadline, amountRaised, and withdrawn. It also defines a public variable 'token' of type EduToken and a public array 'campaigns'. The constructor initializes the token variable. The 'createCampaign' function adds a new campaign to the array with initial values. The code editor has line numbers from 1 to 35.

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

import "./EduToken.sol";

contract Crowdfunding {
    struct Campaign {
        address owner;
        string title;
        uint goal;
        uint deadline;
        uint amountRaised;
        bool withdrawn;
    }

    EduToken public token;
    Campaign[] public campaigns;

    mapping(uint => mapping(address => uint)) public contributions;

    constructor(address tokenAddress) {
        token = EduToken(tokenAddress);
    }

    function createCampaign(string memory _title, uint _goal, uint _duration) public {
        campaigns.push(Campaign({
            owner: msg.sender,
            title: _title,
            goal: _goal,
            deadline: block.timestamp + _duration,
            amountRaised: 0,
            withdrawn: false
        }));
    }
}
```

This screenshot continues the view of the 'Crowdfunding.sol' contract from the previous screen. It shows the completion of the 'createCampaign' function and the start of the 'contribute' and 'withdraw' functions. The 'contribute' function allows users to add funds to a campaign, update the total amount raised, and mint tokens. The 'withdraw' function allows campaign owners to withdraw funds if certain conditions are met. The code editor has line numbers from 35 to 58.

```
contract Crowdfunding {
    struct Campaign {
        ...
    }

    EduToken public token;
    Campaign[] public campaigns;

    mapping(uint => mapping(address => uint)) public contributions;

    constructor(address tokenAddress) {
        ...
    }

    function createCampaign(string memory _title, uint _goal, uint _duration) public {
        ...
    }

    function contribute(uint id) public payable {
        Campaign storage c = campaigns[id];
        require(block.timestamp < c.deadline, "Ended");

        c.amountRaised += msg.value;
        contributions[id][msg.sender] += msg.value;

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

    function withdraw(uint id) public {
        Campaign storage c = campaigns[id];
        require(msg.sender == c.owner, "Not owner");
        require(c.amountRaised >= c.goal, "Goal not reached");
        require(block.timestamp >= c.deadline, "Not finished");
        require(!c.withdrawn, "Already");

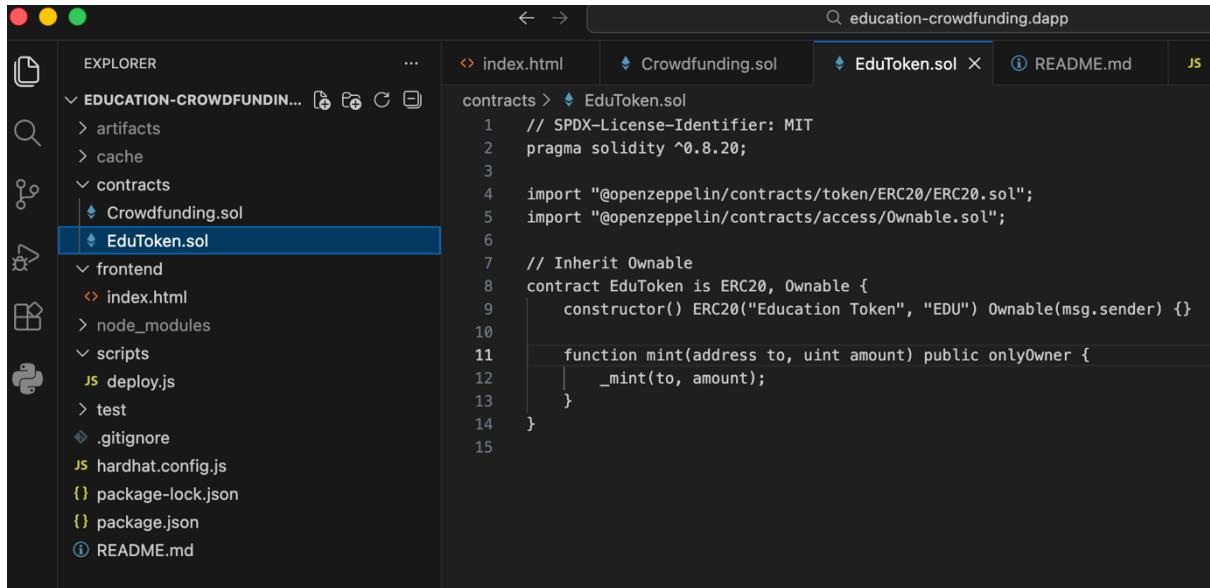
        c.withdrawn = true;
        payable(c.owner).transfer(c.amountRaised);
    }
}
```

Description of Smart Contract Logic

The smart contract defines:

- State variables to store application data.

- Functions that allow users to interact with the contract, such as creating records, updating state, or transferring value.
- Validation logic using `require()` statements to prevent invalid transactions.
- Access control mechanisms where necessary.
- Events that are emitted after successful state changes to notify the frontend.



The screenshot shows a code editor interface with a dark theme. On the left is a file explorer sidebar titled "EXPLORER". Under the "EDUCATION-CROWDFUNDIN..." project, the "contracts" folder contains "Crowdfunding.sol" and "EduToken.sol", with "EduToken.sol" currently selected and highlighted in blue. The main workspace displays the Solidity code for "EduToken.sol". The code defines a contract that inherits from "Ownable" and implements the "ERC20" interface. It includes a "mint" function that allows the owner to issue tokens to another address.

```

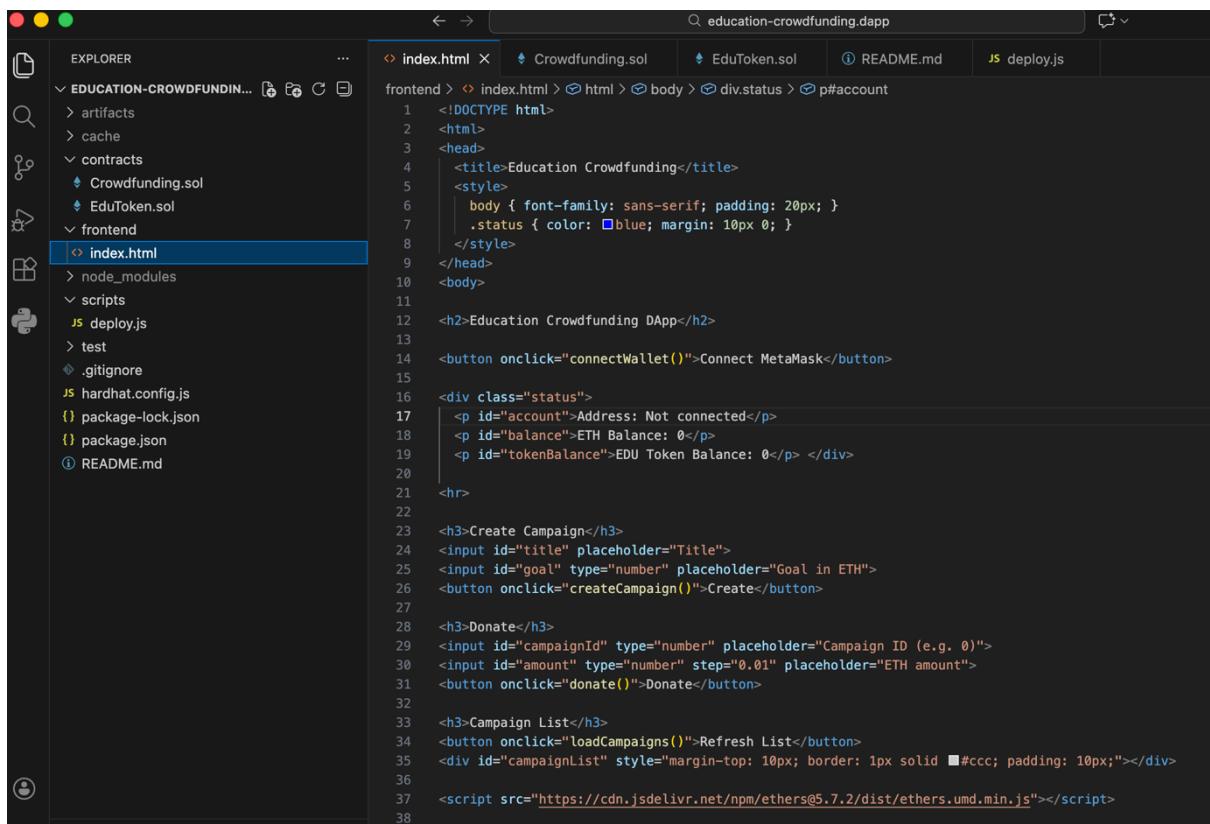
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.20;

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

// Inherit Ownable
contract EduToken is ERC20 {
    constructor() ERC20("Education Token", "EDU") Ownable(msg.sender) {}

    function mint(address to, uint amount) public onlyOwner {
        _mint(to, amount);
    }
}

```



The screenshot shows a code editor interface with a dark theme. On the left is a file explorer sidebar titled "EXPLORER". Under the "EDUCATION-CROWDFUNDIN..." project, the "frontend" folder contains "index.html", which is currently selected and highlighted in blue. The main workspace displays the HTML code for "index.html". The page has a simple layout with a title, a "Connect MetaMask" button, and sections for "status", "Create Campaign", and "Campaign List". It uses CSS to style the "status" section and includes JavaScript functions for interacting with the Ethereum blockchain via MetaMask.

```

<!DOCTYPE html>
<html>
<head>
<title>Education Crowdfunding</title>
<style>
body { font-family: sans-serif; padding: 20px; }
.status { color: blue; margin: 10px 0; }
</style>
</head>
<body>
<h2>Education Crowdfunding DApp</h2>
<button onclick="connectWallet()">Connect MetaMask</button>
<div class="status">
<p id="account">Address: Not connected</p>
<p id="balance">ETH Balance: 0</p>
<p id="tokenBalance">EDU Token Balance: 0</p>
</div>
<hr>
<h3>Create Campaign</h3>
<input id="title" placeholder="Title">
<input id="goal" type="number" placeholder="Goal in ETH">
<button onclick="createCampaign()">Create</button>
<h3>Donate</h3>
<input id="campaignId" type="number" placeholder="Campaign ID (e.g. 0)">
<input id="amount" type="number" step=".01" placeholder="ETH amount">
<button onclick="donate()">Donate</button>
<h3>Campaign List</h3>
<button onclick="loadCampaigns()">Refresh List</button>
<div id="campaignList" style="margin-top: 10px; border: 1px solid #ccc; padding: 10px;"></div>
<script src="https://cdn.jsdelivr.net/npm/ethers@5.7.2/dist/ethers.min.js"></script>

```

```

<html>
<body>
<script>
const crowdfundingAddress = "0x0B306BF915C4d645ff596e518fAf3F9669b97016";
const tokenAddress = "0x9A676e781A523b5d0C0e437313A708CB607508";

const crowdfundingABI = [
  "function createCampaign(string _title, uint256 _goal, uint256 _duration)",
  "function contribute(uint256 id) payable",
  "function campaigns(uint256) view returns (address, string, uint256, uint256, uint256, bool)"
];

const tokenABI = [
  "function balanceOf(address owner) view returns (uint256)"
];

let signer;
let crowdfundingContract;
let tokenContract;

async function connectWallet() {
  if (!window.ethereum) {
    alert("Please install MetaMask!");
    return;
  }

  const provider = new ethers.providers.Web3Provider(window.ethereum);
  await provider.send("eth_requestAccounts", []);
  signer = provider.getSigner();

  const address = await signer.getAddress();
  document.getElementById("account").innerText = "Address: " + address;

  // Network Check Local Network
  const network = await provider.getNetwork();
  console.log("Connected to network:", network.name);
}

```

```

<html>
<body>
<script>
async function connectWallet() {
  if (network.chainId === 1) {
    alert("Critical error: Mainnet usage prohibited! Switch to Local.");
    return; // Stops the execution of the script
  }

  // Updating balances
  updateBalances(address, provider);

  // Initialization of contracts
  crowdfundingContract = new ethers.Contract(crowdfundingAddress, crowdfundingABI, signer);
  tokenContract = new ethers.Contract(tokenAddress, tokenABI, signer);

  alert("Connected to " + address);
}

async function updateBalances(address, provider) {
  // ETH Balance
  const ethBalance = await provider.getBalance(address);
  document.getElementById("balance").innerText = "ETH Balance: " + ethers.utils.formatEther(ethBalance);

  // EDU Token Balance
  if (tokenContract) {
    const eduBalance = await tokenContract.balanceOf(address);
    document.getElementById("tokenBalance").innerText = "EDU Token Balance: " + ethers.utils.formatUnits(eduBalance, 18);
  }
}

```

Frontend-to-Blockchain Interaction

The interaction process is as follows:

1. The user connects their wallet via MetaMask.
2. The frontend retrieves the user's Ethereum account.

3. A contract instance is created using the contract's ABI and deployed address.
4. Read-only data is fetched directly from the blockchain.
5. State-changing functions require user confirmation and gas fees.
6. Transaction results are displayed in the user interface.

Screenshot of a code editor showing the `index.html` file for the crowdfunding application. The code handles campaign creation and donation logic using Solidity contracts and ethers.js.

```

<html>
<body>
<script>
104
105    async function createCampaign() {
106        try {
107            const title = document.getElementById("title").value;
108            const goalEth = document.getElementById("goal").value;
109            const goalWei = ethers.utils.parseEther(goalEth);
110            const duration = 86400 * 7; // 7 days in seconds
111
112            const tx = await crowdfundingContract.createCampaign(title, goalWei, duration);
113            console.log("Pending...");
114            await tx.wait();
115            alert("Campaign Created!");
116        } catch (err) {
117            console.error(err);
118            alert("Error: " + err.reason);
119        }
120    }
121
122    async function donate() {
123        try {
124            const id = document.getElementById("campaignId").value;
125            const amountEth = document.getElementById("amount").value;
126
127            const tx = await crowdfundingContract.contribute(id, {
128                value: ethers.utils.parseEther(amountEth)
129            });
130            console.log("Donating...");
131            await tx.wait();
132
133            alert("Donation Successful!");
134            const address = await signer.getAddress();
135            updateBalances(address, signer.provider); // Updating tokens after a donation
136        } catch (err) {
137            console.error(err);
138            alert("Error: " + err.reason);
139        }
140    }

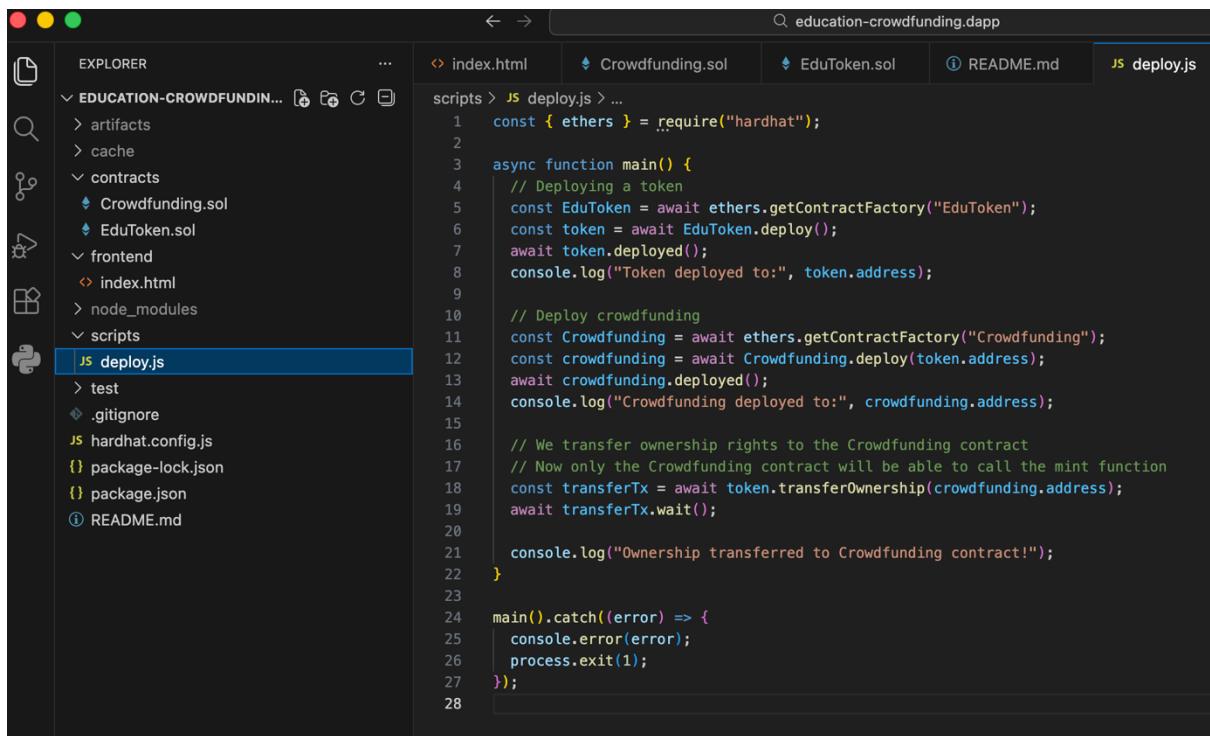
```

Screenshot of a code editor showing the `index.html` file for the crowdfunding application. This version includes a script to load a list of campaigns from the blockchain.

```

<html>
<body>
<script>
143
144    //Update list
145    async function loadCampaigns() {
146        const listDiv = document.getElementById("campaignList");
147        listDiv.innerHTML = "Loading...";
148        try {
149
150            // but Solidity does not give the array length automatically via call
151            let html = "";
152            for (let i = 0; i < 5; i++) {
153                try {
154                    const c = await crowdfundingContract.campaigns(i);
155                    html += `<div><strong>ID ${i}</strong> | Goal: ${ethers.utils.formatEther(c.goal)} ETH | Raised: ${c.raised}</div>`;
156                } catch (e) { break; } // If there is no campaign with this ID, we exit the loop
157            }
158            listDiv.innerHTML = html || "No campaigns found.";
159        } catch (err) {
160            listDiv.innerHTML = "Error loading campaigns.";
161        }
162    }
163    </script>
164
165
166 </body>
</html>

```



```
index.html Crowdning.sol EduToken.sol README.md JS deploy.js
scripts > JS deploy.js > ...
1 const { ethers } = require("hardhat");
2
3 async function main() {
4     // Deploying a token
5     const EduToken = await ethers.getContractFactory("EduToken");
6     const token = await EduToken.deploy();
7     await token.deployed();
8     console.log("Token deployed to:", token.address);
9
10    // Deploy crowdfunding
11    const Crowdning = await ethers.getContractFactory("Crowdning");
12    const crowdfunding = await Crowdning.deploy(token.address);
13    await crowdfunding.deployed();
14    console.log("Crowdning deployed to:", crowdfunding.address);
15
16    // We transfer ownership rights to the Crowdning contract
17    // Now only the Crowdning contract will be able to call the mint function
18    const transferTx = await token.transferOwnership(crowdning.address);
19    await transferTx.wait();
20
21    console.log("Ownership transferred to Crowdning contract!");
22 }
23
24 main().catch((error) => {
25     console.error(error);
26     process.exit(1);
27 });
28 }
```

Deployment and Execution Instructions

Smart Contract Deployment

1. Install dependencies:
2. `npm install`
3. Compile smart contracts:
4. `npx hardhat compile`
5. Deploy contracts to a test network:

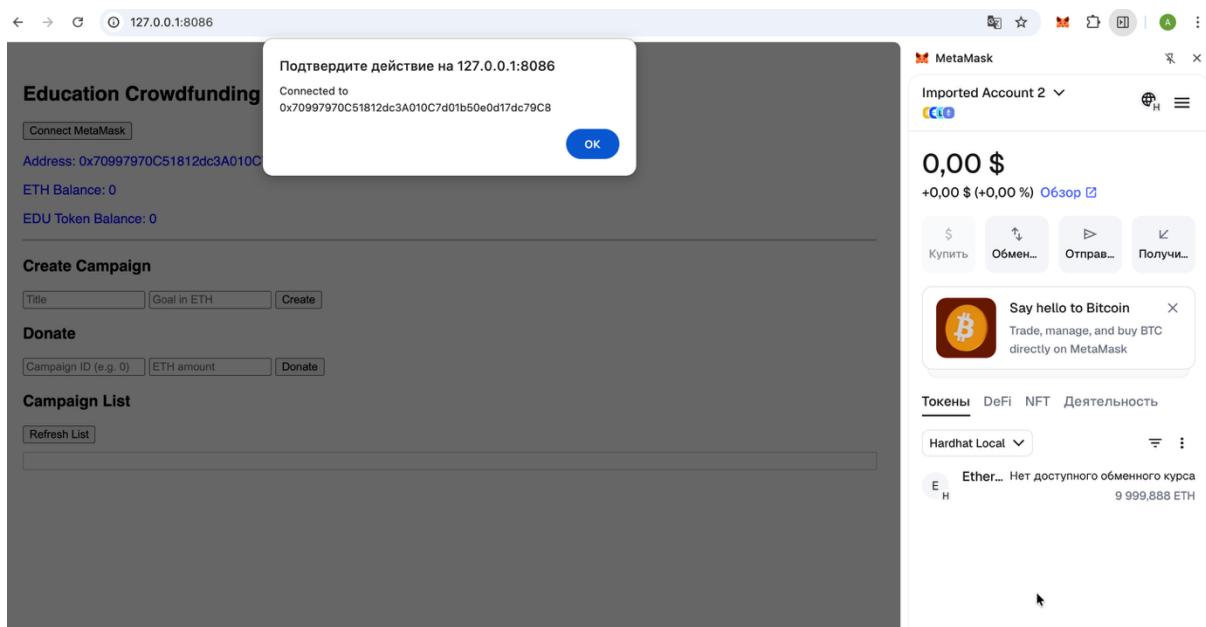
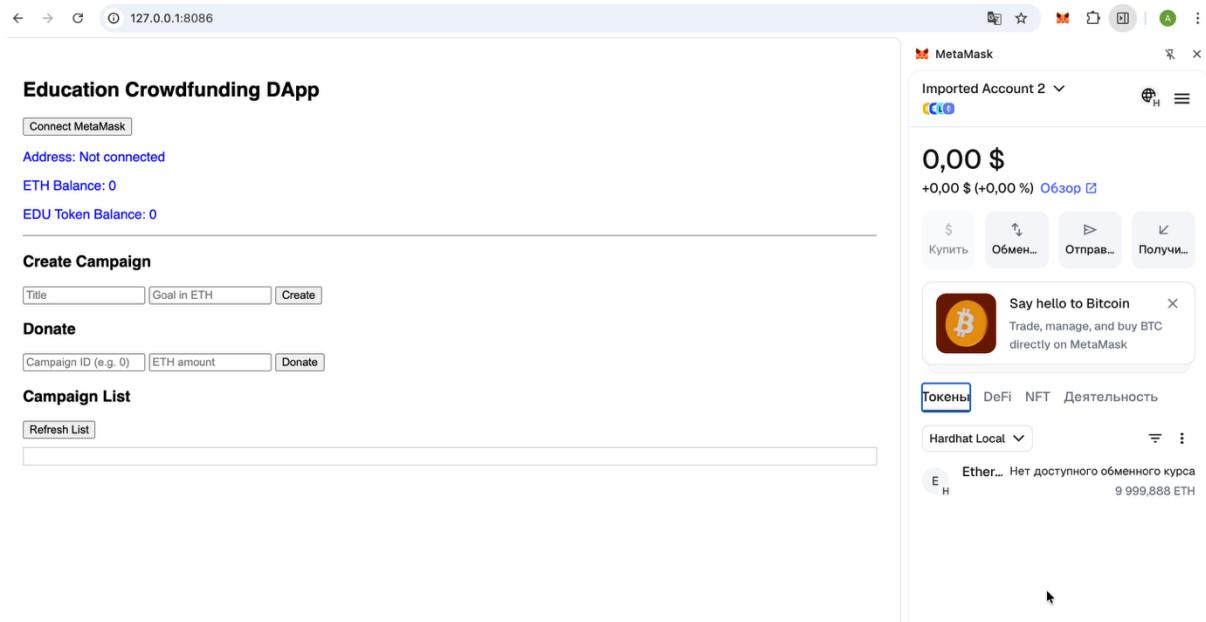
```
npx hardhat run scripts/deploy.js --network sepolia
```

Frontend Execution

1. Navigate to the frontend directory.
2. Install dependencies:
3. `npm install`
4. Start the development server:
5. `npm run dev`
6. Open the application in a browser and connect MetaMask.

```
hardhat.config.js > ...
1  require("@nomiclabs/hardhat-ethers");
2
3  module.exports = {
4    solidity: "0.8.20",
5    networks: {
6      localhost: {
7        url: "http://127.0.0.1:8545"
8      }
9    }
10 };
11
12 
```

```
package.json > ...
1  {
2    "name": "education-crowdfunding-dapp",
3    "version": "1.0.0",
4    "license": "MIT",
5    "devDependencies": {
6      "@nomiclabs/hardhat-ethers": "^2.2.3",
7      "ethers": "^5.8.0",
8      "hardhat": "^2.22.0"
9    },
10   "dependencies": {
11     "@openzeppelin/contracts": "^5.4.0"
12   }
13 }
14 
```



Process for Obtaining Test ETH

To obtain test ETH:

1. Install and set up MetaMask.
2. Switch the network to an Ethereum test network (e.g., Sepolia).
3. Visit a testnet faucet.
4. Enter the wallet address and request test ETH.
5. Once confirmed, the test ETH will appear in the wallet balance.

127.0.0.1:8086

Education Crowdfunding DApp

Address: 0x70997970C51812dc3A010C7d01b50e0d17dc79C8
ETH Balance: 9999.888245220074405551
EDU Token Balance: 11.0

Create Campaign

Title Goal in ETH Create

Donate

Campaign ID (e.g. 0) ETH amount Donate

Campaign List

Refresh List

MetaMask

Imported Account 2

0,00 \$ +0,00 \$ (+0,00 %) Обзор

Купить Обмен... Отправ... Получи...

Say hello to Bitcoin Trade, manage, and buy BTC directly on MetaMask

Токены DeFi NFT Деятельность

Hardhat Local

Ether... Нет доступного обменного курса 9 999,888 ETH

MetaMask

Imported Account 2 Imported accounts

Запрос транзакции

Сеть Hardhat Local
Запрос от ② 127.0.0.1:8086
Взаимодействие с ② 0x0B306...97016
Метод ② Create Campaign

Комиссия сети ② 0.0001 Ⓜ ETH 0,26 \$
Скорость
Максимальная комиссия ② 0.0001 0,26 \$

Одноразовый код ② 18

Данные
Функция createCampaign
Param #1 ② Test
Param #2 ② 10000...00000
Param #3 ② 604800

Отмена Подтвердить

MetaMask

Imported Account 2 Imported accounts

0x0B306...97016

Метод ② Create Campaign

Комиссия сети ② 0.0001 Ⓜ ETH 0,26 \$
Скорость
Максимальная комиссия ② 0.0001 0,26 \$

Одноразовый код ② 18

Данные
Функция createCampaign
Param #1 ② Test
Param #2 ② 10000...00000
Param #3 ② 604800

Отмена Подтвердить

← → ⌛ 127.0.0.1:8086

Education Crowdfunding

Address: 0x70997970C51812dc3A010C7d01b50e0d17dc79C8
ETH Balance: 9999.888245220074405551
EDU Token Balance: 11.0

Create Campaign

Test 1

Donate

Campaign ID (e.g. 0) ETH amount

Campaign List

Refresh List

Подтвердите действие на 127.0.0.1:8086
Campaign Created!
OK

MetaMask

Imported Account 2

0,00 \$ +0,00 \$ (+0,00 %) Обзор

Купить Обмен... Отправ... Получи...

Say hello to Bitcoin Trade, manage, and buy BTC directly on MetaMask

Токены DeFi NFT Деятельность

Hardhat Local

Feb 8, 2026

Событие	Статус	Валюта	Значение
Create Campaign	Подтверждено	-0 ETH	-0,00 \$
Contribute	Подтверждено	-0,01 ETH	-20,96 \$
Create Campaign		-0 ETH	

← → ⌛ 127.0.0.1:8086

Education Crowdfunding DApp

Address: 0x70997970C51812dc3A010C7d01b50e0d17dc79C8
ETH Balance: 9999.888245220074405551
EDU Token Balance: 11.0

Create Campaign

Test 1

Donate

Campaign ID (e.g. 0) ETH amount

Campaign List

Refresh List

MetaMask

Imported Account 2

0,00 \$ +0,00 \$ (+0,00 %) Обзор

Купить Обмен... Отправ... Получи...

Say hello to Bitcoin Trade, manage, and buy BTC directly on MetaMask

Токены DeFi NFT Деятельность

Hardhat Local

Feb 8, 2026

Событие	Статус	Валюта	Значение
Create Campaign	Подтверждено	-0 ETH	-0,00 \$
Contribute	Подтверждено	-0,01 ETH	-20,96 \$
Create Campaign		-0 ETH	

Left Screenshot (Request Step):

Imported Account 2
Imported accounts

Запрос транзакции

Сеть Hardhat Local
Запрос от 127.0.0.1:8086
Взаимодействие с 0x0B306...97016
Метод Contribute
Сумма 0.01 ETH
Комиссия сети 0.0001 ETH
Скорость 0.12 \$
Максимальная комиссия 0.0001 0.12 \$
Одноразовый код 19
Сумма 0.01 ETH
Комиссия сети 0.0001 ETH
Скорость 0.12 \$
Максимальная комиссия 0.0001 0.12 \$
Одноразовый код 19

Отмена Подтвердить

Right Screenshot (Confirmation Step):

Imported Account 2
Imported accounts

0x0B306...97016
Метод Contribute
Сумма 0.01 ETH
Комиссия сети 0.0001 ETH
Скорость 0.12 \$
Максимальная комиссия 0.0001 0.12 \$
Одноразовый код 19
Данные
Функция contribute
Param #1 0
Сумма 0.01 ETH
Комиссия сети 0.0001 ETH
Скорость 0.12 \$
Максимальная комиссия 0.0001 0.12 \$
Одноразовый код 19

Отмена Подтвердить

Education Crowdfunding

Подтвердите действие на 127.0.0.1:8086
Donation Successful!

OK

Address: 0x70997970C51812dc3A010C7001b5660d178c79C8
ETH Balance: 9999.888245220074405551
EDU Token Balance: 11.0

Create Campaign
Test 1 Create
Donate 0 0.01 Donate
Campaign List Refresh List

MetaMask History:

Imported Account 2
0,00 \$ +0,00 \$ (+0,00 %) Обзор

OK

Токены DeFi NFT Деятельность

Hardhat Local

Feb 8, 2026

Действие	Сумма
Contribute	-0.01 ETH
Create Campaign	-0 ETH
Contribute	-0.01 ETH

Say hello to Bitcoin
Trade, manage, and buy BTC directly on MetaMask

The screenshot shows the MetaMask browser extension interface. At the top, there are several icons: a lock, a star, a fox logo, a folder, a square with a circle, a green circle with 'A', and a three-dot menu. The title bar says "MetaMask". Below the title bar, it displays "Imported Account 2" with a dropdown arrow and a small icon. To the right are a refresh button and a close button.

The main area shows a balance of **0,00 \$** with a note "+0,00 \$ (+0,00 %) [Обзор](#)". Below the balance are four buttons: "Купить" (Buy), "Обмен..." (Swap), "Отправ..." (Send), and "Получи..." (Receive).

A promotional box for Bitcoin integration is visible, titled "Say hello to Bitcoin" with the subtext "Trade, manage, and buy BTC directly on MetaMask".

Below the promotional box, there are tabs: Токены (Tokens), DeFi, NFT, and Деятельность (Activity). The "Hardhat Local" network is selected.

The activity section shows two transactions from "Feb 8, 2026":

	Contribute	-0.01 ETH
	Подтверждено	-20,96 \$

	Create Campaign	-0 ETH
	Подтверждено	-0,00 \$

Education Crowdfunding DApp

[Connect MetaMask](#)

Address: 0x70997970C51812dc3A010C7d01b50e0d17dc79C8

ETH Balance: 9999.878063777825101951

EDU Token Balance: 12.0

Create Campaign

Donate

Campaign List

[Refresh List](#)

No campaigns found.
