Final Project

Students:

Bekesh Dastan,

Shyngys Abdimomyn,

Issabek Suleimenov

Documentation

Part 1. Overview of the application architecture.

1.1. Overview

Crowdfunding platform – which is created to help young start-ups to launch their products. Our application is decentralized and build on ethereum. It solves the problem of transparency and intermediates. In our application will not take a place the intermediates as in centralized systems. And investors can take the take the erc20 tokens from investments, for example you invest 1 ethereum, you will get 100 tokens.

1.2. Smart Contracts

Smart Contract is the heart of the application. Which contracts we have?

erc20.sol – implementation of general erc20 token with all necessary methods.

CroudfundingContract – implementation of the main contract of our app with main functions as create_campaign, donate and etc.

1.3. Blockchain

Blockchain ensures decentralization of platform which created in ethereum due to this, our app don't have intermediates. To make connection between blockchain and front-end we use ether.js

1.4. Frontend

Basic front-end with js, css and external css libraries. We strive to make our site beautiful, that why design of our site is pretty good.

1.5. Wallet

We use metamask as our and clients' wallet. Because it's too comfortable us and our clients☺

Part 2. Explanation of design and implementation decisions.

2.1. Why ether.js instead of web3.js?

In opinions of our team, ether.js is preferable because it's much more readable a light. Our codes are less complex and easier to edit or maintain in the future. Furthermore, ether.js is more modern, and more people know this language, and in our university we learned this library more.

2.2. No database and backend

We don't have databases because it will make our application centralized. And in front of security is worsen then decentralized app. If one server is fall, others will not.

2.3. Security

All our contracts work through require, which ensure the security of money and data. For example, only owner of company can get money from fund.

Part 3. Description of smart contract logic.

3.1. erc20 contract

This common contract in major part of that kind application. It improves interconnectivity with other application.

There is basic list of functions which I include:

balanceOf – to get balance

Transfer - to transfer tokens

Approve – to approve using owner money

Allowance - to see how much money you can use

TransferFrom – to transfer other guys' money.

Mint – to add some money

Burn – to decrease the balance

3.2. Croudfunding contract

This contract is designed to implement all main functions of our app. This is custom contract to perform company creation, donation and etc.

There is basic list of functions which I include:

Create_campaign – to create company

Donate - to donate to the young company

Finalize - to finalize the project, the company will get money

refund - when goal isn't reached, investors will get money back.

getters – to get some data such as count of companies, list of companies and user contributions

Part 4. Explanation of frontend-to-blockchain interaction.

The interaction takes place in three stages:

1. Connection:

window.ethereum.request requests access to the wallet.

const accounts = await window.ethereum.request({ method: 'eth_accounts' });

Here we get all account from the browser

2. Provider and Signer:

The Provider is used to read data such as balances and list of campaigns.

provider = new ethers.BrowserProvider(window.ethereum); - TO GET PROVIDER

Signer is used to record data (campaign creation, donation), as these actions require signing and paying for gas.

signer = await provider.getSigner(); - TO GET SIGNER

3. Data Formatting:

Ethers functions are used.formatEther to show the balance to the user and ethers.parseEther to send data from input fields to the blockchain. We make it because computers mostly work with Wei

Part 5. Deployment and execution instructions.

1. Write to the terminal

npx hardhat node.

2. Open contracts in remix which located in the browser.

3. Check that your wallet is connected to your local network.

4. Compile erc20.sol

5. Deploy it injected provider with all parameters.

6. Copy the link to erc20.sol.

7. Compile CroudfundingContract.sol

8. Deploy it in injected provider with the link to erc20.sol

9. Copy and paste abi and links to app.js

Part 6. Description of the process for obtaining test ETH

We got all our test ETH through hardhat. Hardhat installation steps

1. Initialize npm project

npm init -y

2. Install hardhat

Npm install –save-dev hardhat

3. Create hardhat project

Npx hardhat init

npm install –save-dev hardhat@^2.22.0

npm install @nomicfoundation/hardhat-toolbox

4. Start local blockchain

npx hardhat node

Hardhat test ethereums are very useful. We can test our contracts and make some transactions in our local blockchain without deployment to the network. Also it easy to initialize this code to our app.

The last and the main step:

Stay positive and smile a lot☺

Additional Work.

https://youtu.be/aJWJCUX2iig

And GithubLink

https://github.com/BekeshDastan/Crowdfunting-Platform