

Blockchain and Cryptocurrency

Assignment 10: Crowdfunding DApp

The 10th assignment is to develop a simple DApp for crowdfunding, including Front-end, Back-end and Database. This assignment will help you understand the whole process of developing a DApp and connection with each part.

- This assignment includes creating a simple crowdfunding DApp.
 - You must submit a report containing the results (Web screen) obtained as you go through **<Step. 1>** and your code (files).
- ⇒ **Submit a report containing all results obtained from all steps and your codes.**

<TA>

Email: kkc90@postech.ac.kr

<Step. 0> Connect a private network

※ You can join other's private network by connecting a node. To connect a node, each node has the same network id and genesis file. By using attached genesis file, enode and IP address, join the private network which is already built before.

※ Refer to the previous assignment. (Assignment 5)

- IP: 141.223.82.142

- Port: 30303

- enode:

9157807b41da7be331120e8bd94afabae22d99b8c312c80ed1223fde71cbe33a304e8b0e3a9ed8f0a3551e4ec38ad6225ab5ecb7393e4e6765a53bb75de3ce9e

<Step. 1> Create a crowdfunding DApp

Crowdfunding DApp is a simple DApp that allows beneficiaries to raise funds to successfully carry out their own projects. This DApp has two main pages. One is for a beneficiary who wants to be funded through this DApp and the other makes it accessible to investors to fund the project.

This DApp roughly consists of three parts:

- Front-end: Web pages to an owner (beneficiary) and users (investors)
- Back-end: Smart contracts (Crowdfunding.sol, CustomToken.sol)
- Database: Ethereum Blockchain (A provided private network)

Requirements: Front-end

- Two main pages (for an owner and investors)

- [Owner's page]
 - This main page has to show the amount of currently funded ETH and goal to be funded
 - It has to show the address of the owner of this crowdfunding and the amount of ETH which the owner of this crowdfunding has
 - It has to show contract address of this crowdfunding and the amount of ETH and custom token(CTK) which this contract holds
 - It has to show the status of crowdfunding such as "End" or "Ongoing"
 - It has to have the link to go back the main page (All pages have it)
 - It has to have the button to function that the owner can withdraw funds when crowdfunding is successful
 - (Optional) Implement the button to check if the goal is currently reached or not
- [Investor's page]
 - This main page has to show the amount of currently funded ETH and goal to be funded
 - It has to show the number of the remaining tokens (Custom token to be given investors as a token of participation in this crowdfunding)
 - It has to have the link to go back the main page (All pages have it)
 - It has to have the button to link to a page that investors can join crowdfunding
 - In the page of "Join crowdfunding", to join crowdfunding, it needs an account address to fund, the amount of funds and passphrase to unlock the account (It also has to have to button to proceed in funding)
 - After funding is done, the detail page of the investor right before has to be shown, including investor's address, the amount of ETH which the investor funded and the amount of token which it currently holds
 - (Optional) Implement the button to withdraw ETH by the investor when this crowdfunding is failed (Time is over before funding reaches the goal) and investor overfunds this crowdfunding (the amount of goal to be funded exceeds)

Requirements: Back-end / Database

- Crowdfunding.sol
 - You can extend and improve the functionality of your smart contract used in assignment 7
 - You may use "event" in .sol to detect this event call on front-end using .watch()
 - A sample code for crowdfunding is provided in [References], please refer to this sample code
 - Implement withdrawExcess function for users when overfunding happens
- CustomToken.sol
 - Use a code of Custom token in [References]

- Database
 - Connect to the ethereum private network provided
- ※ This is good for quickly creating an application skeleton. Use the application generator tool.
 - **Express-generator:** <http://expressjs.com/en/starter/generator.html>
- ※ The quality of UI is not important. Please focus on implementing the functionality mentioned above.

Preparations:

- Make three accounts: Owner, Investor 1, Investor 2
- [Using owner's account] Create Crowdfunding and CustomToken / Deploy them to the blockchain => each contract address will be generated
- [Using owner's account] Transfer custom tokens from CustomToken to Crowdfunding contract
- When creating each contract, parameters are needed:
 - [CustomToken] Token name = "Anything you want", Token symbol="CTK", Decimals = 0, Initial supply = 1000
 - [Crowdfunding] Funding goal = 1000, Duration(Minutes) = 10, Cost of each token = 1 (means 1:1), address of token contract for rewards = "CustomToken's address"

- ※ Lists to be included in the report: two scenarios

[First: Success]

- 1) Show two main pages for an owner and investors
- 2) Show the page for investors to join crowdfunding
- 3) Show the page for checking the investor 1's funding status after investor 1 successfully finishes funding 500 ETH
- 4) Show the page for checking the investor 2's funding status after investor 2 successfully finishes funding 700 ETH
- 5) Show the main page of the owner after investor 1 and 2 fund this crowdfunding
- 6) Show the main page of investors after executing checkGoalReached function
- 7) Show the main page of the owner after executing withdraw function by the owner

[Second: Fail]

- 1) Show two main pages for an owner and investors (The same as a case of success)
- 2) Show the page for investors to join crowdfunding (The same as a case of success)
- 3) Show the page for checking the investor 1's funding status after investor 1 successfully finishes funding 500 ETH

- 4) Show the page for checking the investor 2's funding status after investor 2 successfully finishes funding 200 ETH
- 5) Show the main page of the owner after investor 1 and 2 fund this crowdfunding
- 6) Show the main page of investors after executing checkGoalReached function
- 7) Show the main page of investors after executing withdraw function by each investor
- 8) Show the main page of the owner


[References]

1. Example) A scenario of success: Crowdfunding for a filmmaker

1) [Initial] A main page for a beneficiary

[\[Go to the main page for crowdfunding\]](#)

Current status : ##### Crowdfunding is still under way. #####



The amount of currently funded ETH : 0(Current) / 1000(Total) ETH

The amount of ETH which FilmMaker holds : 2776.080509074 ETH

[Withdraw the collection from crowdfunding](#)

@ CrowdFunding Contract's Account :
0xfa376bDccD736c89124cdEC211c0140b0c4eCF9E : 0 (ETH), 0 (CTK)


@ Film-maker's Account :
0xb7efc43d2c91fa9402072cfb0fc56bf042f77d43 : 2776.080509074 (ETH), 0 (CTK)

- If you execute transfer function of CustomToken to transfer 1000 token to Crowdfunding contract address, the amount of CTK will increase to 1000

2) [Initial] The other main page for investors

[\[Go to the main page for crowdfunding\]](#)

Current status : ##### Crowdfunding is still under way. #####



The amount of currently funded ETH : 0(Current) / 1000(Total) ETH


The amount of currently remaining Token : 1000 CTK

[Join crowdfunding](#)

- 3) [Connected to investor's main page] The page for investors to join this crowdfunding

[\[Go to the main page for crowdfunding\]](#)

Current status : ##### Crowdfunding is still under way. #####



[Join crowdfunding]

Account to fund

The amount of funding


Passphase

[Funding!](#)

- 4) A page for investors to check their status after investor 1 finished funding

[\[Go to the main page for crowdfunding\]](#)

Current status : ##### Crowdfunding is still under way. #####



The amount of currently funded ETH : 500(Current) / 1000(Total) ETH

The amount of currently remaining Token : 500 CTK

[Join crowdfunding](#)

@ User's Details @

User Account : 0x73c0c9765d7b221cbd6d9dc88185008f79ceba68


The amount of ETH which user funds: 500 ETH

The amount of Token which user currently holds: 500 CTK

- 5) A page for investors to check their status after investor 2 finished funding

[\[Go to the main page for crowdfunding\]](#)

Current status : ##### Crowdfunding is still under way. #####



The amount of currently funded ETH : 1000(Current) / 1000(Total) ETH

The amount of currently remaining Token : 0 CTK

[Join crowdfunding](#)

@ User's Details @

User Account : 0x94d3b81dE11E1843A3F6a79b95DDB7037C0b544F


The amount of ETH which user funds: 500 ETH

The amount of Token which user currently holds: 500 CTK

- 6) A main page for a beneficiary to check crowdfunding's status after the goal was reached

[\[Go to the main page for crowdfunding\]](#)

Current status : ##### Crowdfunding is still under way. #####



The amount of currently funded ETH : $1000(\text{Current}) / 1000(\text{Total})$ ETH

The amount of ETH which FilmMaker holds : 2776.079867731 ETH

Withdraw the collection from crowdfunding


@ CrowdFunding Contract's Account :
 0xE2b69cFA9A3034E64580043fd9c7708EeCfa45a3 : 1000 (ETH), 0 (CTK)

@ Film-maker's Account :
 0xb7efc43d2c91fa9402072cfc0fc56bf042f77d43 : 2776.079867731 (ETH), 0 (CTK)

7) A main page for investors after executing checkGoalReached function

[\[Go to the main page for crowdfunding\]](#)

Current status : ##### Crowdfunding already ended. #####



The amount of currently funded ETH : $1000(\text{Current}) / 1000(\text{Total})$ ETH

The amount of currently remaining Token : 0 CTK

Join crowdfunding

8) A main page for a beneficiary after executing withdraw function by the beneficiary

[\[Go to the main page for crowdfunding\]](#)

Current status : ##### Crowdfunding already ended. #####



The amount of currently funded ETH : 1000(Current) / 1000(Total) ETH

The amount of ETH which FilmMaker holds : 2776.079769023 ETH

Withdraw the collection from crowdfunding

@ CrowdFunding Contract's Account :

0xE2b68cFA9A3034E64580043fd9c7708EeCfa45a3 : 1000 (ETH), 0 (CTK)

@ Film-maker's Account :

0xb7efc43d2c91fa9402072cfb0fc56bf042f77d43 : 2776.079769023 (ETH), 0 (CTK)

2. Sample code of smart contract: CustomToken

```
pragma solidity ^0.4.18;

contract CustomToken {
    string public name;
    string public symbol;
    uint8 public decimals;

    mapping (address => uint256) public balanceOf;

    event Transfer(address _from, address _to, uint _value);

    constructor(string _tokenName,string _tokenSymbol,uint8 _decimalUnits, uint256
_initialSupply) public {
        name = _tokenName;
        symbol = _tokenSymbol;
        decimals = _decimalUnits;
        balanceOf[msg.sender] = _initialSupply;
```

```

    }

    function transfer(address _to, uint256 _value) public {
        require(balanceOf[msg.sender] >=_value);
        require(balanceOf[_to] + _value >= balanceOf[_to] );
        balanceOf[msg.sender] -= _value;
        balanceOf[_to] += _value;
        emit Transfer(msg.sender,_to,_value);
    }
}

```

3. Sample code of smart contract: Crowdfunding

```

pragma solidity ^0.4.18;

interface token {
    function transfer(address receiver, uint amount) external;
}

contract CrowdFunding {

    address public owner;
    uint public goalAmount;
    uint public totalAmount;
    uint public deadline;
    uint public price;
    token public tokenReward;
    mapping(address => uint256) public balanceOf;
    bool public goalReached;
    bool public ended;
}

```

```
event GoalReached(address ownerAddress, uint amountRaisedValue);
event FundTransfer(address backer, uint amount, bool isContribution);
```

```
modifier onlyOwner() {
    require(msg.sender == owner);
    _;
}
```

```
modifier afterDeadline() {
    require (now >= deadline);
    _;
}
```

```
constructor(
    uint _goalAmount,
    uint _durationMinutes,
    uint _costOfToken,
    address _tokenAddress
) public {
    owner = msg.sender;
    goalAmount = _goalAmount * 1 ether;
    deadline = now + _durationMinutes * 1 minutes;
    price = _costOfToken * 1 ether;
    tokenReward = token(_tokenAddress);
    totalAmount = 0;
    goalReached = false;
    ended = false;
}
```

```
function () payable external {
    require(!ended);
    uint amount = msg.value;
    balanceOf[msg.sender] += amount;
    totalAmount += amount;
    tokenReward.transfer(msg.sender, amount / price);
    emit FundTransfer(msg.sender, amount, true);
}
```

```
function checkGoalReached () external afterDeadline {
```

```

        require(!ended);
        if (totalAmount >= goalAmount){
            goalReached = true;
            emit GoalReached(owner, totalAmount);
        }
        ended = true;
    }

    function withdraw() external afterDeadline {
        if (!goalReached) {
            uint amount = balanceOf[msg.sender];
            balanceOf[msg.sender] = 0;
            if (amount > 0) {
                if (msg.sender.send(amount)) {
                    emit FundTransfer(msg.sender, amount, false);
                } else {
                    balanceOf[msg.sender] = amount;
                }
            }
        }
        if (goalReached && owner == msg.sender) {
            if (owner.send(totalAmount)) {
                emit FundTransfer(owner, totalAmount, false);
            } else {
                goalReached = false;
            }
        }
    }

    function kill() public onlyOwner {
        selfdestruct(owner);
    }
}

```