

## DA51 Lab Session 8 : Dapp Crowdfunding

Question 2:

Smart Contract:

```
Crowdfunding.sol x
1 // SPDX-License-Identifier: UNLICENSED
2 pragma solidity >=0.4.22 <0.9.0;
3
4 contract Crowdfunding {
5     struct Campaign {
6         address payable creator;
7         uint256 goal;
8         uint256 deadline;
9         uint256 balance;
10        bool closed;
11    }
12
13    mapping(uint256 => Campaign) public campaigns;
14    uint256 public campaignId;
15
16    function startCampaign(uint256 _goal, uint256 _deadline) public returns (uint256) {
17        require(_goal > 0, "Goal must be greater than 0");
18        require(_deadline > block.timestamp, "Deadline must be in the future");
19
20        campaigns[campaignId] = Campaign(creator: {
21            creator: msg.sender,
22            goal: _goal,
23            deadline: _deadline,
24            balance: 0,
25            closed: false
26        });
27
28        campaignId++;
29        return campaignId - 1;
30    }
31
32    function contribute(uint256 _campaignId) public payable returns (uint256) {
33        Campaign storage campaign = campaigns[_campaignId];
34        require(msg.value > 0, "Contribution must be greater than 0");
35        require(!campaign.closed, "Campaign is closed");
36        require(block.timestamp < campaign.deadline, "Deadline has passed");
37
38        campaign.balance += msg.value;
39        return campaign.balance;
40    }
41
42    function checkGoalReached(uint256 _campaignId) public returns (bool) {
43        Campaign storage campaign = campaigns[_campaignId];
44        require(!campaign.closed, "Campaign is closed");
45        //require(block.timestamp >= campaign.deadline, "Deadline has not passed");
46
47        if (campaign.balance >= campaign.goal) {
48            return true;
49        } else {
50            return false;
51        }
52    }
53
54    function closeCampaign(uint256 _campaignId) public {
55        Campaign storage campaign = campaigns[_campaignId];
56        require(!campaign.closed, "Campaign is already closed");
57
58        campaign.closed = true;
59    }
60
61    function withdraw(uint256 _campaignId) public {
62        Campaign storage campaign = campaigns[_campaignId];
63        require(campaign.closed, "Campaign is not closed");
64        require(msg.sender == campaign.creator, "Only the creator can withdraw funds");
65
66        campaign.creator.transfer(value: campaign.balance);
67    }
68 }
```

### Question 3 :

#### Compile and migration :

```
Terminal Local x + v
PS E:\Documents\UTBM\Cours\DA51\Lab session 8> truffle deploy

Compiling your contracts...
=====
> Compiling .\contracts\Crowdfunding.sol
> Compilation warnings encountered:

    /E:/Documents/UTBM/Cours/DA51/Lab session 8/contracts/Crowdfunding.sol:42:5: Warning: Function
ew
    function checkGoalReached(uint256 _campaignId) public returns (bool) {
      ^ (Relevant source part starts here and spans across multiple lines).

> Artifacts written to E:\Documents\UTBM\Cours\DA51\Lab session 8\build\contracts
> Compiled successfully using:
   - solc: 0.5.16+commit.9c3226ce.Emscripten.clang

Starting migrations...
=====
> Network name:    'development'
> Network id:     5777
> Block gas limit: 0x6691b7

1_initial_migration.js
=====

Replacing 'Migrations'
-----
> transaction hash: 0xf3e107d212865a101b692cb5cd0114d4eb5a6795b4df5eb1c5511f167ae75a5
> Blocks: 0        Seconds: 0
> contract address: 0xc19Fc2390824A9df11c29f44FF57fFe4d619CacC
> block number:    1
> block timestamp: 1732039851
> account:         0x88581997729445DF98a0398fF0b6503ed4eb5a43
> balance:         99.99613514
> gas used:        193243
> gas price:       20 gwei
> value sent:      0 ETH
> total cost:      0.00386486 ETH

> Saving migration to chain.
> Saving artifacts
-----
> Total cost:      0.00386486 ETH

2_deploy_contracts.js
=====

Replacing 'Crowdfunding'
-----
> transaction hash: 0xc3d914d0445081d47078e7dcd58b3b72fe4b3b30bd5c61e55ff507b38e8274a9
> Blocks: 0        Seconds: 0
> contract address: 0xF89A2aCa7A0C4f9CAfb776BF3d8C07cB17D5336a
> block number:    3
> block timestamp: 1732039851
> account:         0x88581997729445DF98a0398fF0b6503ed4eb5a43
> balance:         99.98299894
> gas used:        611072
> gas price:       20 gwei
> value sent:      0 ETH
> total cost:      0.01222144 ETH

> Saving migration to chain.
> Saving artifacts
-----
> Total cost:      0.01222144 ETH

Summary
=====
> Total deployments: 2
Using network 'development'.
```

## Question 4:

### Testing:

```
TestCrowdfunding.js x
1  const Crowdfunding = artifacts.require("Crowdfunding");
2
3  contract("Crowdfunding", (accounts) => {
4      let crowdfundingInstance;
5
6      const [creator, contributor] = accounts;
7
8      beforeEach(async () => {
9          crowdfundingInstance = await Crowdfunding.new();
10     });
11
12     it("should start a campaign", async () => {
13         const goal = web3.utils.toWei("10", "ether");
14         const deadline = Math.floor(Date.now() / 1000) + 3600; // 1 hour from now
15
16         await crowdfundingInstance.startCampaign(goal, deadline, { from: creator });
17
18         const campaign = await crowdfundingInstance.campaigns(0);
19         assert.equal(campaign.creator, creator, "Campaign creator is incorrect");
20         assert.equal(campaign.goal.toString(), goal, "Campaign goal is incorrect");
21         assert.equal(campaign.deadline.toNumber(), deadline, "Campaign deadline is incorrect");
22         assert.equal(campaign.balance.toString(), "0", "Campaign balance should start at 0");
23         assert.equal(campaign.closed, false, "Campaign should not be closed initially");
24     });
25
26     it("should allow contributions", async () => {
27         const goal = web3.utils.toWei("10", "ether");
28         const deadline = Math.floor(Date.now() / 1000) + 3600;
29
30         await crowdfundingInstance.startCampaign(goal, deadline, { from: creator });
31
32         const contributionAmount = web3.utils.toWei("1", "ether");
33         await crowdfundingInstance.contribute(0, { from: contributor, value: contributionAmount });
34
35         const campaign = await crowdfundingInstance.campaigns(0);
36         assert.equal(campaign.balance.toString(), contributionAmount, "Campaign balance should reflect the contribution");
37     });
38
39     it("should close the campaign and allow withdrawal when the goal is reached", async () => {
40         const goal = web3.utils.toWei("2", "ether");
41         const deadline = Math.floor(Date.now() / 1000) + 3600;
42
43         await crowdfundingInstance.startCampaign(goal, deadline, { from: creator });
44
45         await crowdfundingInstance.contribute(0, { from: contributor, value: web3.utils.toWei("2", "ether") });
46
47         await crowdfundingInstance.checkGoalReached(0, { from: creator });
48
49         const campaign = await crowdfundingInstance.campaigns(0);
50         assert.equal(campaign.closed, true, "Campaign should be closed after goal is reached");
51
52         const initialBalance = web3.utils.toBN(await web3.eth.getBalance(creator));
53         await crowdfundingInstance.withdraw(0, { from: creator });
54
55         const finalBalance = web3.utils.toBN(await web3.eth.getBalance(creator));
56         assert(finalBalance.gt(initialBalance), "Creator should have withdrawn funds");
57     });
58
59     it("should fail to contribute after the campaign deadline", async () => {
60         const goal = web3.utils.toWei("10", "ether");
61         const deadline = Math.floor(Date.now() / 1000) + 1; // 1 second from now
62
63         await crowdfundingInstance.startCampaign(goal, deadline, { from: creator });
64
65         await new Promise((executor, resolve) => setTimeout(resolve, 2000));
66
67         try {
68             await crowdfundingInstance.contribute(0, { from: contributor, value: web3.utils.toWei("1", "ether") });
69             assert.fail("Contribution should fail after the deadline");
70         } catch (error) {
71             assert(error.message.includes("Deadline has passed"), "Expected 'Deadline has passed' error");
72         }
73     });
74 });
```

### Question 5 :

```
PS E:\Documents\UTBM\Cours\DA51\Lab session 8> truffle test
Using network 'development'.

Compiling your contracts...
=====
> Everything is up to date, there is nothing to compile.

Contract: Crowdfunding
  ✓ should start a campaign
  ✓ should allow contributions
  ✓ should close the campaign and allow withdrawal when the goal is reached (59ms)
  ✓ should fail to contribute after the campaign deadline (2046ms)

4 passing (2s)
```

### Question 6-7 :

Ganache

ACCOUNTS

BLOCKS

TRANSACTIONS

CONTRACTS

EVENTS

LOGS

SEARCH FOR BLOCK NUMBERS OR TX HASHES

CURRENT BLOCK  
55

GAS PRICE  
20000000000

GAS LIMIT  
6721975

HARDFORK  
MERGE

NETWORK ID  
5777

RPC SERVER  
HTTP://127.0.0.1:7545

MINING STATUS  
AUTOMINING

WORKSPACE  
QUICKSTART

SAVE

SWITCH

Lab session 8E:\Documents\UTBM\Cours\DA51\Lab session 8

NAME	ADDRESS	TX COUNT	
Crowdfunding	0xF89A2aCa7A0C4f9CAfb776BF3dBC07cB17D5336a	0	DEPLOYED
Migrations	0xc19Fc2390824A9df11c29f44FF57fFe4d619CAcC	1	DEPLOYED

Question 8-9-10 :

Index.html :

```
<> index.html x
1  <!DOCTYPE html>
2  <html lang="en">
3  >  <head>
43  </head>
44  <body>
45  <h1>Crowdfunding DApp</h1>
46
47  <!-- Start Campaign Section -->
48  <h2>Start a Campaign</h2>
49  <label for="goal">Goal (in Wei):</label>
50  <input type="number" id="goal" placeholder="Enter goal in wei">
51  <label for="deadline">Deadline (UNIX Timestamp):</label>
52  <input type="number" id="deadline" placeholder="Enter deadline timestamp">
53  <button class="btn btn-primary btn-block btn-start-campaign">Start Campaign</button>
54  <p id="startCampaignResult"></p>
55
56  <!-- Contribute Section -->
57  <h2>Contribute to a Campaign</h2>
58  <label for="campaignId">Campaign ID:</label>
59  <input type="number" id="campaignId" placeholder="Enter campaign ID">
60  <label for="amount">Amount (in Wei):</label>
61  <input type="number" id="amount" placeholder="Enter amount to contribute">
62  <button class="btn btn-primary btn-block btn-contribute">Contribute</button>
63  <p id="contributeResult"></p>
64
65  <!-- Check Goal Reached Section -->
66  <h2>Check if Campaign Goal Reached</h2>
67  <label for="campaignIdCheck">Campaign ID:</label>
68  <input type="number" id="campaignIdCheck" placeholder="Enter campaign ID">
69  <button class="btn btn-primary btn-block btn-check-goal">Check Goal</button>
70  <p id="checkGoalResult"></p>
71
72  <!-- Close the campaign -->
73  <h2>Close a Campaign</h2>
74  <label for="campaignIdClose">Campaign ID:</label>
75  <input type="number" id="campaignIdClose" placeholder="Enter campaign ID">
76  <button class="btn btn-primary btn-block btn-close">Close</button>
77  <p id="closeResult"></p>
78
79  <!-- Withdraw Section -->
80  <h2>Withdraw Funds</h2>
81  <label for="campaignIdWithdraw">Campaign ID:</label>
82  <input type="number" id="campaignIdWithdraw" placeholder="Enter campaign ID">
83  <button class="btn btn-primary btn-block btn-withdraw">Withdraw</button>
84  <p id="withdrawResult"></p>
85
86  <!-- jQuery (necessary for Bootstrap's JavaScript plugins) -->
87  <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>
88  <!-- Include all compiled plugins (below), or include individual files as needed -->
89  <script src="js/bootstrap.min.js"></script>
90  <script src="js/web3.min.js"></script>
91  <script src="js/truffle-contract.js"></script>
92  <script src="js/app.js"></script>
93  </body>
94  </html>
```

App.js :

- Function to dialog with smart contract
  - o The first screen is about starting a campaign, contributing to and check if the crowdfunding reaches the goal

```
startCampaignJs: function (event) :void {
    event.preventDefault();
    var doc;
    const goal = $('#goal').val();
    if (!goal) {
        alert('Please enter a goal');
        return;
    }
    const deadline = $('#deadline').val();
    if (!deadline) {
        alert('Please enter a deadline');
        return;
    }
    web3.eth.getAccounts(function (error, accounts) :void {
        if (error) {
            console.log(error);
        }
        var account = accounts[0];
        App.contracts.Crowdfunding.deployed().then(function (instance) {
            doc = instance;
            return doc.startCampaign(goal, deadline, {from: account});
        }).then(function (result) :void {
            $('#startCampaignResult').text('Campaign started successfully' + result);
        }).catch(function (err) :void {
            console.log(err.message);
        });
    });
},

contributeJs: function (event) :void {
    event.preventDefault();
    var doc;
    const campaignId = $('#campaignId').val();
    const amount = $('#amount').val();
    if (!amount) {
        alert('Please enter an amount');
        return;
    }
    web3.eth.getAccounts(function (error, accounts) :void {
        if (error) {
            console.log(error);
        }
        var account = accounts[0];
        App.contracts.Crowdfunding.deployed().then(function (instance) {
            doc = instance;
            return doc.contribute(campaignId, {from: account, value: amount});
        }).then(function (result) :void {
            $('#contributeResult').text('Contribution successful, balance:' + result);
        }).catch(function (err) :void {
            console.log(err.message);
        });
    });
},

checkGoalJs: function (event) :void {
    event.preventDefault();
    const campaignId = $('#campaignIdCheck').val();
    web3.eth.getAccounts(function (error, accounts) :void {
        if (error) {
            console.log(error);
        }
        var account = accounts[0];
        App.contracts.Crowdfunding.deployed().then(function (instance) {
            return instance.checkGoalReached(campaignId, {from: account});
        }).then(function (result) :void {
            if (result) {
                $('#checkGoalResult').text('Goal reached');
            } else {
                $('#checkGoalResult').text('Goal not reached');
            }
        }).catch(function (err) :void {
            console.log(err.message);
        });
    });
},
```

- The second screen is about closing and withdrawing a campaign

```
closeCampaignJs: function (event) :void {
  event.preventDefault();
  const campaignId = $('#campaignIdClose').val();
  web3.eth.getAccounts(function (error, accounts) :void {
    if (error) {
      console.log(error);
    }
    var account = accounts[0];
    App.contracts.Crowdfunding.deployed().then(function (instance) {
      return instance.close(campaignId, {from: account});
    }).then(function (result) :void {
      $('#closeResult').text(value: 'Campaign closed');
    }).catch(function (err) :void {
      console.log(err.message);
    });
  });
},

withdrawJs: function (event) :void {
  event.preventDefault();
  const campaignId = $('#campaignIdClose').val();
  web3.eth.getAccounts(function (error, accounts) :void {
    if (error) {
      console.log(error);
    }
    var account = accounts[0];
    App.contracts.Crowdfunding.deployed().then(function (instance) {
      return instance.withdraw(campaignId, {from: account});
    }).then(function (result) :void {
      $('#withdrawResult').text(value: 'Withdrawal successful');
    }).catch(function (err) :void {
      console.log(err.message);
    });
  });
},
}
```

Final Result:

# Crowdfunding DApp

## Start a Campaign

Goal (in Wei):  Deadline (UNIX Timestamp):

Start Campaign

## Contribute to a Campaign

Campaign ID:  Amount (in Wei):

Contribute

## Check if Campaign Goal Reached

Campaign ID:

Check Goal

## Close a Campaign

Campaign ID:

Close

## Withdraw Funds

Campaign ID:

Withdraw



Function tests:

# Crowdfunding DApp

## Start a Campaign

Goal (in Wei):

45

Deadline (UNIX Timestamp):

1111111111

Start Campaign

Campaign started successfully0xf426a3defc3b0d57e0f3f9701a83f4f845ff975c3a10e767c13905d3105c56e6

## Contribute to a Campaign

Campaign ID:

04263e30570397018348

Amount (in Wei):

10

Contribute

Contribution successful, balance:[object Object]

## Check if Campaign Goal Reached

Campaign ID:

04263e30570397018348

Check Goal

Goal reached

## Close a Campaign

Campaign ID:

04263e30570397018348

Close

## Withdraw Funds

Campaign ID:

04263e30570397018348

Withdraw

Withdrawal successful