ASSIGNMENT – 8

Aim: Implementing your own Ethereum Setup

A smart contract is a computer program that automatically executes an agreement between two parties when certain conditions are met:

- How it works

Smart contracts are stored on a blockchain, where the terms are permanent and cannot be changed. When the conditions of the contract are met, the program automatically executes the actions required.

- Benefits

Smart contracts can reduce the need for intermediaries, arbitration costs, and fraud losses. They can also make transactions traceable, transparent, and irreversible.

- Uses

Smart contracts can be used for a variety of purposes, including:

  - Financial transactions: Smart contracts can be used to automate mortgage transactions, or to create decentralized finance (DeFi) apps that allow users to make loans and savings without a bank.
  - Supply chain: Smart contracts can be used to track the steps of a product through the supply chain, which can help eliminate errors, theft, and loss.
  - Property ownership: Smart contracts can be used to register property ownership more efficiently.

1. Introduction

This report presents the development and deployment of a **Fundraising Smart Contract** using Solidity on the Remix IDE. The contract is designed to facilitate fundraising campaigns by allowing users to contribute Ether and enabling the creator to withdraw funds once the target is reached.

2. Problem Statement

The primary goal of this project is to create a smart contract that:

- Allows a creator to initiate a fundraising campaign with a specified target amount.
- Enables users to contribute Ether to the campaign.
- Allows the creator to withdraw the funds only after reaching the target amount.

3. Smart Contract Code

Here is the complete code for the Fundraising smart contract:

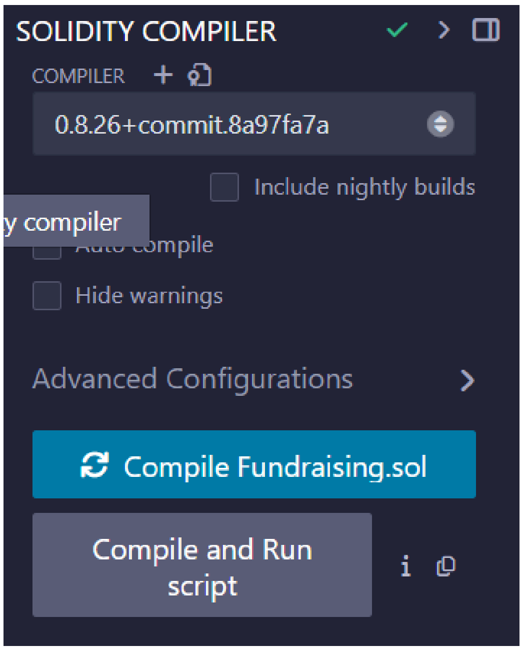
```
// SPDX-License-Identifier: MIT
```

```
pragma solidity ^0.8.0;

/// @title Fundraising Contract
/// @notice This contract allows users to contribute to a fundraising campaign
contract Fundraising {
    address public creator;
    uint public targetAmount;
    uint public totalContributions;
    bool public isTargetReached;
    mapping(address => uint) public contributions;

    event ContributionReceived(address indexed contributor, uint amount);
    event FundsWithdrawn(address indexed creator, uint amount);

    constructor(uint _targetAmount) {
        creator = msg.sender;
        targetAmount = _targetAmount;
        isTargetReached = false;
    }

    /// @notice Contribute to the fundraising campaign
    function contribute() external payable {
        require(msg.value > 0, "Contribution must be greater than 0");
        require(!isTargetReached, "Target has already been reached");

        contributions[msg.sender] += msg.value;
        totalContributions += msg.value;

        emit ContributionReceived(msg.sender, msg.value);

        // Check if the target has been reached
        if (totalContributions >= targetAmount) {
            isTargetReached = true;
        }
    }

    /// @notice Withdraw funds if the target is reached
    function withdraw() external {
        require(msg.sender == creator, "Only creator can withdraw");
        require(isTargetReached, "Target not reached yet");

        uint amount = totalContributions;
        totalContributions = 0; // Reset total contributions
        payable(creator).transfer(amount);

        emit FundsWithdrawn(creator, amount);
    }

    /// @notice Get the current status of the campaign
    function getStatus() external view returns (address, uint, uint, bool) {
        return (creator, targetAmount, totalContributions, isTargetReached);
    }
}
```

4. Deployment Instructions

To deploy the Fundraising contract in Remix IDE, follow these steps:

1. **Open Remix IDE**: Go to Remix IDE.
2. **Create a New File**: Name the file Fundraising.sol.
3. **Code**: Insert the code provided in Section 3.
4. **Compile the Contract**: Select the correct compiler version (0.8.0 or higher) and compile.


SOLIDITY COMPILER

COMPILER

0.8.26+commit.8a97fa7a

Include nightly builds

Auto compile

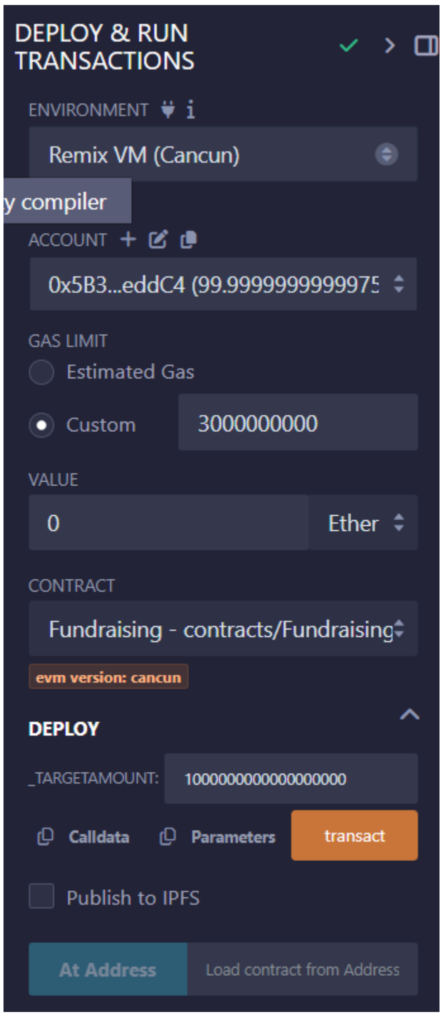
Hide warnings

Advanced Configurations

Compile Fundraising.sol

Compile and Run script

5. **Deploy the Contract**:
   - **Environment**: Set to **Remix VM (Cancun)**.
   - **Value**: Set to 0 Ether.
   - **_TARGETAMOUNT**: Input 1000000000000000000 (for 1 Ether).
   - Click **transact** to deploy.


DEPLOY & RUN TRANSACTIONS

ENVIRONMENT: Remix VM (Cancun)

ACCOUNT: 0x5B3...eddC4 (99.99999999999975

GAS LIMIT: Estimated Gas / Custom 3000000000

VALUE: 0 Ether

CONTRACT: Fundraising - contracts/Fundraising

evm version: cancun

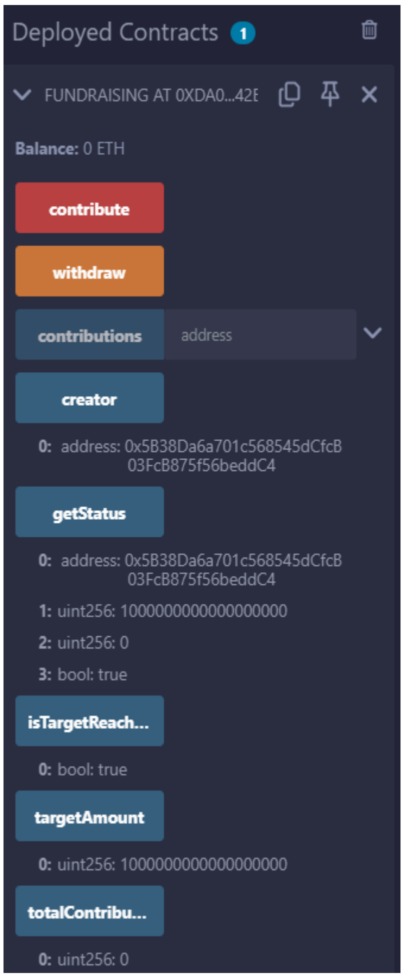
DEPLOY

_TARGETAMOUNT: 1000000000000000000

Calldata Parameters transact

Publish to IPFS

At Address Load contract from Address


Deployed Contracts 1

FUNDRAISING AT 0XDA0...42E

Balance: 0 ETH

contribute

withdraw

contributions address

creator

0: address: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4

getStatus

0: address: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4
1: uint256: 1000000000000000000
2: uint256: 0
3: bool: true

isTargetReach...

0: bool: true

targetAmount

0: uint256: 1000000000000000000

totalContribu...

0: uint256: 0

5. Interaction with the Contract

Once deployed, you can interact with the contract as follows:

Contributing to the Fundraising Campaign

- Set the **Value** to the amount you want to contribute (e.g., 0.5 for 0.5 Ether).
- Click the **contribute** button.

Checking Campaign Status

- Call the getStatus function to view:
  - Creator's address
  - Target amount
  - Total contributions
  - Target status (reached or not)

Withdrawing Funds

- Once the target amount is reached, the creator can click the withdraw function to transfer the funds.

6. Conclusion

The Fundraising Smart Contract demonstrates the practical use of blockchain technology in managing fundraising campaigns securely and transparently. Users can contribute funds, while the creator maintains control over the withdrawal process, ensuring that funds are only accessed once the fundraising goal is met.

7. References

- Solidity Documentation: Solidity Docs
- Remix IDE: Remix IDE