# K. K. Wagh Institute of Engineering Education and Research, Nashik.

## Department of Computer Engineering

## Academic Year 2022-23

**Course Name:** Laboratory Practice – III **Course Code:** 410246

**Class: BE** **Div:** A

**Name of Students:** Devendra Bandu Solunke (37)

Gayatri Manikchand Deshmukh (38)

Sarthak Manikrao Wagh (39)

Aditya Sanjay Agrawal (40)

**Name of Faculty:** Prof. C. R. Patil

---

**Mini Project Report**

**Title of Mini-Project:** Develop a blockchain based application- Crowd Funding System using smart contract and deploy it using Ethereum.

## Objective:

1. Understand and explore the working of Blockchain technology and its applications.

2. Write Smart contract using solidity language and deploy it using ethereum.

## Introduction of Mini-Project:

Nowadays, Crowd funding platform in block-chain makes different possibilities for the start-ups by raising the funds to create their own digital currency and it is peer-to-peer fund raising model some of the famous crowd funding crypto currencies are coin space, swarm, judo-baby etc. Blockchain is a unique, independent and a transparent system which keep the transactions between parties transparent. Crowd funding is based on the trust between the investors and stakeholders. So, in this mini project we have write smart contract for crowd funding system using solidity language and successfully deploy it on Remix IDE with ethereum.

## Smart Contract:

Smart contracts are simply programs stored on a blockchain that run when predetermined conditions are met. They typically are used to automate the execution of an agreement so that all participants can be immediately certain of the outcome, without any intermediary's involvement or time loss. They can also automate a workflow, triggering the next action when conditions are met. To put it simply the code you write that works on Blockchain is called a smart contract. And to write those codes you have to use Solidity Programming language.

## Process of Build Smart Contract

1. Coding
2. Compilation – Used EVM
3. Deployed – to deploy you should have wallet with Public Private Key Pair & some Ethereum to spend as gas fee. (Remix provide 10 account with 100 Ethereum with each)

**Tool:** Remix IDE Provide all facilities like smart contract to compilation, deployment, testing interface, etc

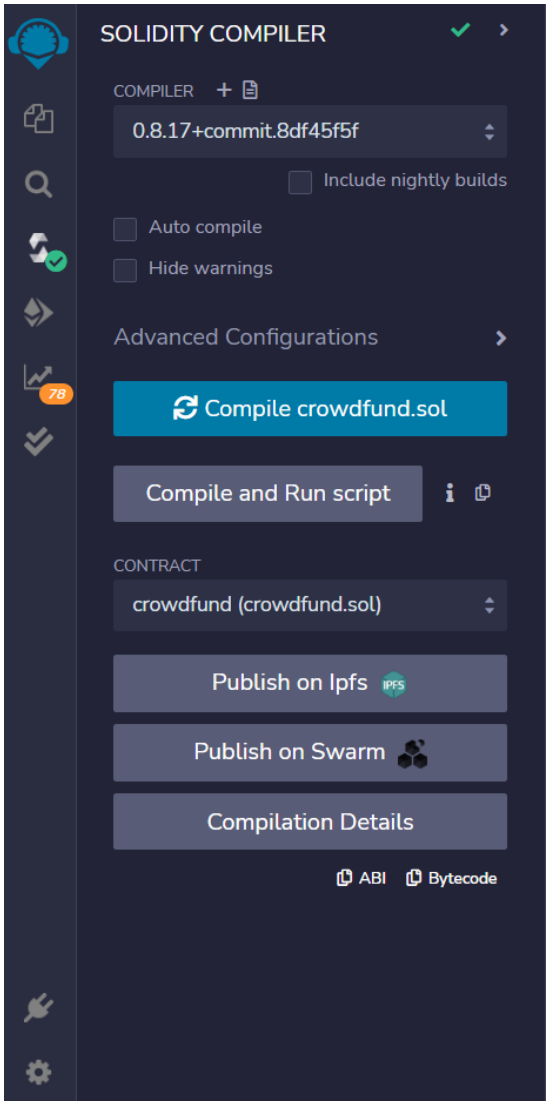
## Code:

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.13;

import "./IERC20.sol";

contract CrowdFund {
    event Launch(
        uint id,
        address indexed creator,
        uint goal,
        uint32 startAt,
        uint32 endAt
    );
    event Cancel(uint id);
    event Pledge(uint indexed id, address indexed caller, uint amount);
    event Unpledge(uint indexed id, address indexed caller, uint amount);
    event Claim(uint id);
    event Refund(uint id, address indexed caller, uint amount);

    struct Campaign {
        address creator;
        uint goal;
        uint pledged;
        uint32 startAt;
        uint32 endAt;
        bool claimed;
    }

    IERC20 public immutable token;
    uint public count;
    mapping(uint => Campaign) public campaigns;
    mapping(uint => mapping(address => uint)) public pledgedAmount;

    constructor(address _token) {
        token = IERC20(_token)
```

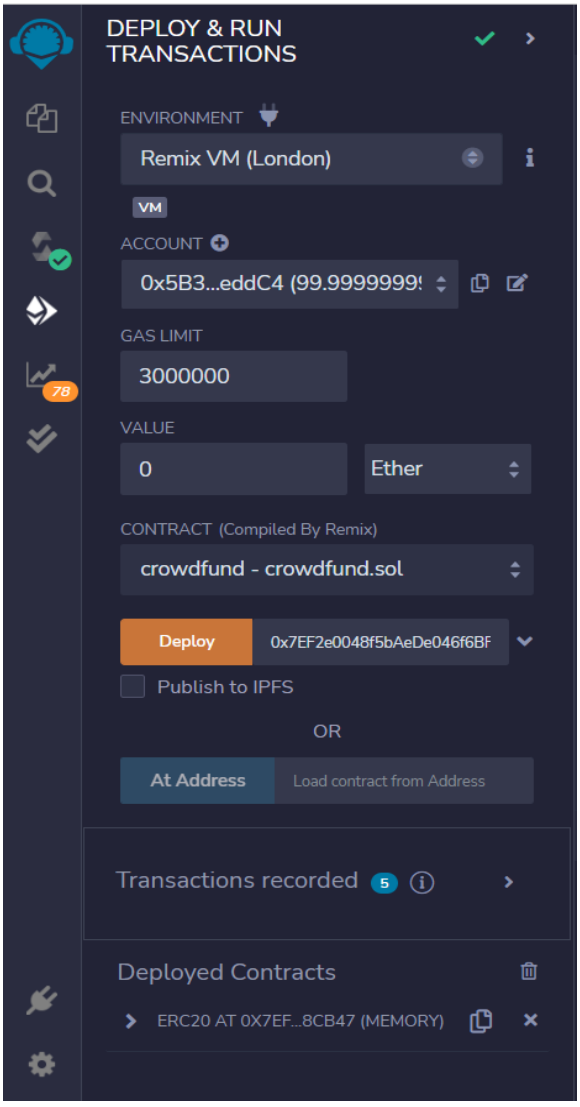
```
}

function launch(
    uint _goal,
    uint32 _startAt,
    uint32 _endAt
) external {
    require(_startAt >= block.timestamp, "start at < now");
    require(_endAt >= _startAt, "end at < start at");
    require(_endAt <= block.timestamp + 90 days, "end at > max duration");

    count += 1;
    campaigns[count] = Campaign({
        creator: msg.sender,
        goal: _goal,
        pledged: 0,
        startAt: _startAt,
        endAt: _endAt,
        claimed: false
    });

    emit Launch(count, msg.sender, _goal, _startAt, _endAt);
}

function cancel(uint _id) external {
    Campaign memory campaign = campaigns[_id];
    require(msg.sender == campaign.creator , "not creator");
    require(block.timestamp < campaign.startAt, "started");

    delete campaigns[_id];
    emit Cancel(_id);
}

function pledge(uint _id, uint _amount) external {
    Campaign storage campaign = campaigns[_id];
    require(block.timestamp >= campaign.startAt, "not started");
    require(block.timestamp <= campaign.endAt, "ended");

    campaign.pledged += _amount;
    pledgedAmount[_id][msg.sender] += _amount;
    token.transferFrom(msg.sender, address(this), _amount);

    emit Pledge(_id, msg.sender, _amount);
}

function unpledge(uint _id, uint _amount) external {
    Campaign storage campaign = campaigns[_id];
    require(block.timestamp <= campaign.endAt, "ended");
```

```
        campaign.pledged -= _amount;
        pledgedAmount[_id][msg.sender] -= _amount;
        token.transfer(msg.sender, _amount);

        emit Unpledge(_id, msg.sender, _amount);
    }

    function claim(uint _id) external {
        Campaign storage campaign = campaigns[_id];
        require(campaign.creator == msg.sender, "not creator");
        require(block.timestamp > campaign.endAt, "not ended");
        require(campaign.pledged >= campaign.goal, "pledged < goal");
        require(!campaign.claimed, "claimed");

        campaign.claimed = true;
        token.transfer(campaign.creator, campaign.pledged);

        emit Claim(_id);
    }

    function refund(uint _id) external {
        Campaign memory campaign = campaigns[_id];
        require(block.timestamp > campaign.endAt, "not ended");
        require(campaign.pledged < campaign.goal, "pledged >= goal");

        uint bal = pledgedAmount[_id][msg.sender];
        pledgedAmount[_id][msg.sender] = 0;
        token.transfer(msg.sender, bal);

        emit Refund(_id, msg.sender, bal);
    }
}
```

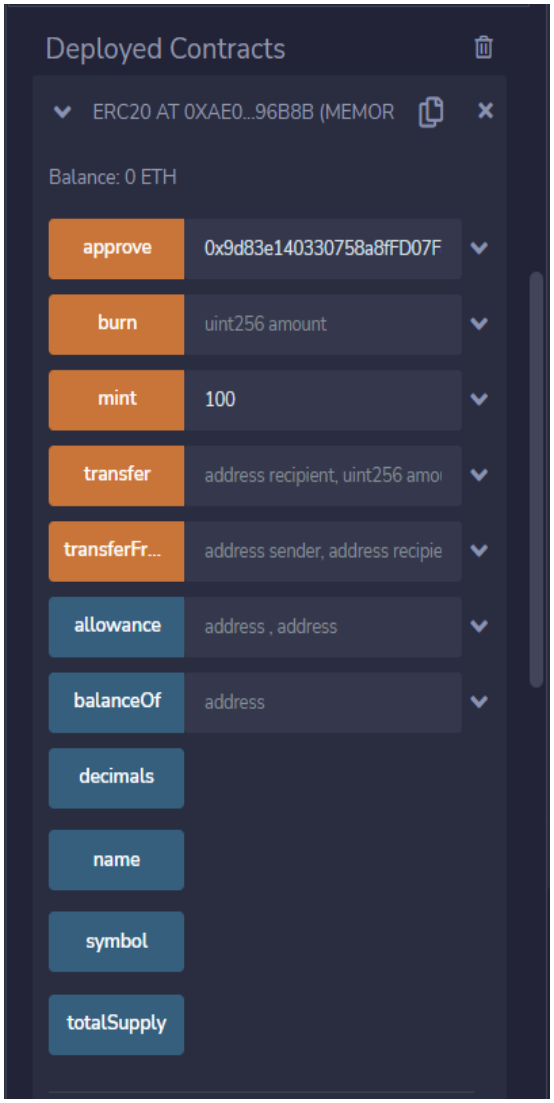
## Output:



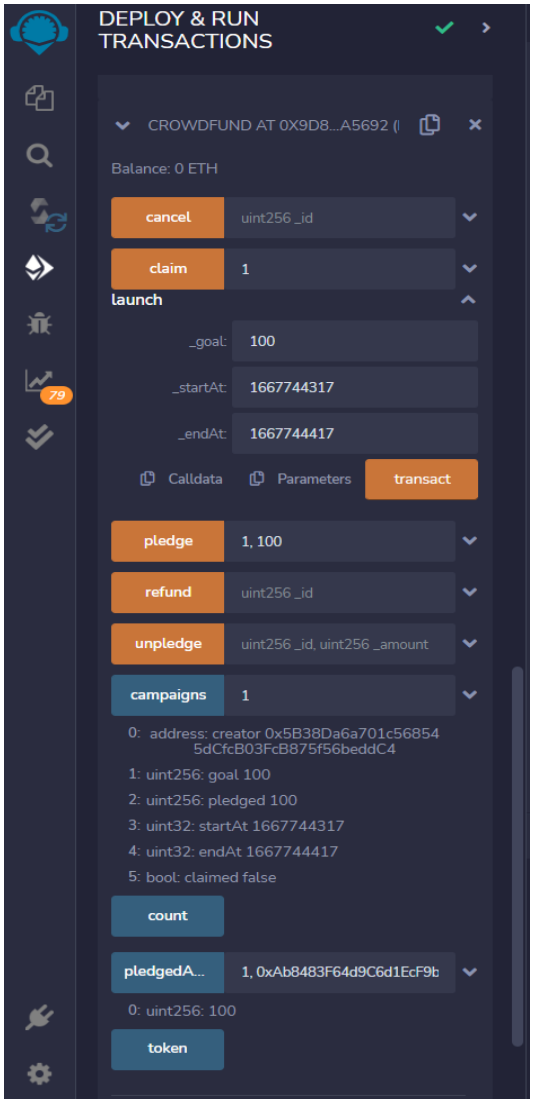

1) Successful Compilation of Smart contract



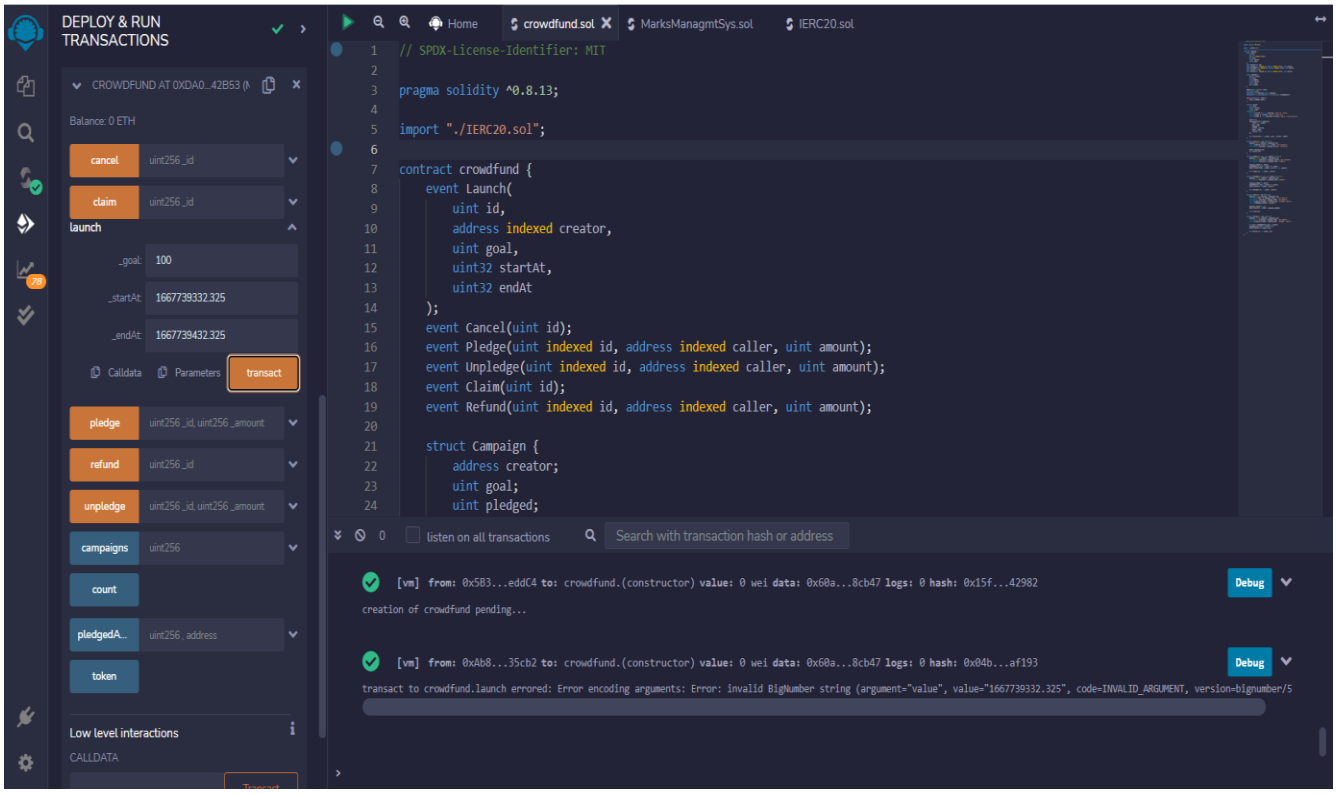

Deploy the smart contract




3) Adding the mint and approving the user




4) Launching the goal, Pledging the transactions, campaigns and claim the successful funding




5) Successful Crowd Funding

## Conclusion:

Our mini-project is on “Develop a blockchain based application- Crowd Funding System using smart contract and deploy it using Ethereum.” From this mini-project, we understood and explored the working of Blockchain technology and its applications like Movie Rating System Smart Contract. We also learned about smart contract based blockchain concept and implement it using solidity language and finally deploy the smart contract on Ethereum successfully for Movie Rating System.