**Code Eater**

//SPDX-License-Identifier: UNLICENSED

pragma solidity >=0.5.0 < 0.9.0;

contract CrowdFunding{

mapping(address=>uint) public contributors; //contributors[msg.sender]=100

address public manager;

uint public minimumContribution;

uint public deadline;

uint public target;

uint public raisedAmount;

uint public noOfContributors;

struct Request{

string description;

address payable recipient;

uint value;

bool completed;

uint noOfVoters;

mapping(address=>bool) voters;

}

mapping(uint=>Request) public requests;

uint public numRequests;

constructor(uint \_target,uint \_deadline){

target=\_target;

deadline=block.timestamp+\_deadline; //10sec + 3600sec (60\*60)

minimumContribution=100 wei;

manager=msg.sender;

}

function sendEth() public payable{

require(block.timestamp < deadline,"Deadline has passed");

require(msg.value >=minimumContribution,"Minimum Contribution is not met");

if(contributors[msg.sender]==0){

noOfContributors++;

}

contributors[msg.sender]+=msg.value;

raisedAmount+=msg.value;

}

function getContractBalance() public view returns(uint){

return address(this).balance;

}

function refund() public{

require(block.timestamp>deadline && raisedAmount<target,"You are not eligible for refund");

require(contributors[msg.sender]>0);

address payable user=payable(msg.sender);

user.transfer(contributors[msg.sender]);

contributors[msg.sender]=0;

}

modifier onlyManger(){

require(msg.sender==manager,"Only manager can calll this function");

\_;

}

function createRequests(string memory \_description,address payable \_recipient,uint \_value) public onlyManger{

Request storage newRequest = requests[numRequests];

numRequests++;

newRequest.description=\_description;

newRequest.recipient=\_recipient;

newRequest.value=\_value;

newRequest.completed=false;

newRequest.noOfVoters=0;

}

function voteRequest(uint \_requestNo) public{

require(contributors[msg.sender]>0,"YOu must be contributor");

Request storage thisRequest=requests[\_requestNo];

require(thisRequest.voters[msg.sender]==false,"You have already voted");

thisRequest.voters[msg.sender]=true;

thisRequest.noOfVoters++;

}

function makePayment(uint \_requestNo) public onlyManger{

require(raisedAmount>=target);

Request storage thisRequest=requests[\_requestNo];

require(thisRequest.completed==false,"The request has been completed");

require(thisRequest.noOfVoters > noOfContributors/2,"Majority does not support");

thisRequest.recipient.transfer(thisRequest.value);

thisRequest.completed=true;

}

}