//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;

}

}