// SPDX-License-Identifier: MIT

pragma solidity ^0.8.0;

contract Crowdfunding {

// Variables

address public manager; // Project creator

uint public goal; // Funding goal (in wei)

uint public deadline; // Time limit for funding (block timestamp)

uint public totalFunds; // Total funds raised

bool public goalReached; // Flag to indicate if the goal is reached

bool public fundsWithdrawn; // Flag to prevent multiple withdrawals

mapping(address => uint) public contributions; // Track contributions by each backer

// Events

event ContributionReceived(address contributor, uint amount);

event GoalReached(uint totalAmount);

event FundsWithdrawn(address recipient, uint amount);

event RefundIssued(address contributor, uint amount);

// Modifier to restrict access to the manager

modifier onlyManager() {

require(msg.sender == manager, "Only the project manager can call this function");

}

// Modifier to ensure the campaign is still active

modifier campaignActive() {

require(block.timestamp < deadline, "The crowdfunding campaign has ended");

}

// Constructor to initialize the crowdfunding campaign

constructor(uint \_goal, uint \_duration) {

manager = msg.sender;

goal = \_goal;

deadline = block.timestamp + \_duration;

}

// Function to contribute funds to the campaign (payable)

function contribute() public payable campaignActive {

require(msg.value > 0, "Contribution must be greater than zero");

// Track contributions

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

totalFunds += msg.value;

// Emit event for contribution

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

// Check if the funding goal is reached

if (totalFunds >= goal) {

goalReached = true;

emit GoalReached(totalFunds);

}

}

// Function to allow the project manager to withdraw funds if the goal is reached

function withdrawFunds() public onlyManager {

require(goalReached, "Funding goal not yet reached");

require(!fundsWithdrawn, "Funds have already been withdrawn");

// Transfer the contract balance to the manager

payable(manager).transfer(address(this).balance);

fundsWithdrawn = true;

// Emit event for withdrawal

emit FundsWithdrawn(manager, address(this).balance);

}

// Function to allow contributors to get a refund if the goal is not met by the deadline

function requestRefund() public {

require(block.timestamp >= deadline, "Crowdfunding campaign is still active");

require(!goalReached, "Goal has been reached, no refunds available");

require(contributions[msg.sender] > 0, "You have no contributions");

// Refund the contributor

uint refundAmount = contributions[msg.sender];

contributions[msg.sender] = 0;

payable(msg.sender).transfer(refundAmount);

// Emit event for refund

emit RefundIssued(msg.sender, refundAmount);

}

// Function to get the contract's current balance

function getBalance() public view returns (uint) {

return address(this).balance;

}

// Function to check the time remaining in the campaign

function timeLeft() public view returns (uint) {

if (block.timestamp >= deadline) {

return 0;

} else {

return deadline - block.timestamp;

}

}

}

A screenshot of a chat

Description automatically generated