Charity Application Smart Contract Tech Specs
Author: Olugbenga Ayoola
Email: ayoolaolugbenga38@yahoo.com

Introduction

The purpose of creating the Charity Application smart Contract is to ensure people can donate to several charities without the fear of the money being mis-used or getting to the wrong hands. Funding projects are created, donations are made, targets are reached and a lot more. Projects can be created to raise more funds in addition to yours and can also be done single handedly.

Features

The Following functionalities are needed in the smart contract:

1. Ability to define the owner of the smart contract
2. Ability to transfer the ownership of the smart contract
3. Ability for the owner to adjust the minimum pledge amount
4. Ability to register charity organizations.
5. Ability to create donation projects.
6. Ability to donate funds to an existing project.
7. Ability to cancel an existing project (only by the creator).
8. Ability to retrieve all donations to a particular project.
9. Ability to retrieve all projects created by a user

Requirements

The contract should :

1. Have the ability to map a charity organisation by unique integer id.
2. Have the ability to map a funding project by unique integer id.
3. Have the ability to map a donation by unique integer id.
4. Have the ability to tag a project successful or unsuccessful.
5. Have the ability to tag a project active or inactive..

Specifications

The following specifications are desired in the smart contract

Library Functions

The contract uses additions and subtractions in various functions, thus, it’s also a requirement to import and use the **SafeMath** library of OpenZeppelin.
https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/math/SafeMath.sol

Structs

The following structs are required:

Name	**Structure**
Charity	The charity struct handles the registration of a charity organisation.

Name	Type
name	*string* Contains the name of the organisation
acc	*address* Contains the wallet address for the organisation.
aboutUs	*string* Contains additional information about the Organisation.
id	*uint* Contains the unique integer id for the charity.

Name	**Structure**
Project	The project struct is used for the creation of a new funding project, it can be used to tag the project active or inactive, successful or unsuccessful and holds other information about the project.

Name	Type
id	*uint* Holds the project unique integer id
charityId	*uint* Holds the integer id of the charity the project is made for.
creator	address To store the address of the person creating the project
timeStarted	*uint* To store the time the project was created.
timeCompleted	*uint* To store the time the project ended.
maxNoOfDays	*uint* It holds the maximum number of days the project can be allowed to run.
initialFund	*uint* To store the amount the project's owner seeded the project with.
projectBalance	*uint* To store the project's current balance.
target	*uint* To store the amount the projectBalance must reach for the project to be completed.
active	*bool* Indicates if the project is still active or not.
successful	*bool* Indicates if the campaign was successful or not.
description	*string* Describes the reason for the project and the purpose it is meant to serve.

Donation	It handles the donation of funds to an existing project.

Name	Type
id	*uint* Holds the donation’s unique integer id
ProjectId	*uint* It holds the integer id for the project the donation is being made to.
amount	*uint* It contains the amount being donated to the project.
donator	*address* To store the address making the donation
timeSent	*uint* To store the time the donation was made

Variables

The following variables are recommended to be used:

Name	**Type**
charities	*array* This array holds all the registered charities and the individual charities can be accessed using their unique integer id.
projects	*array* The array holds all the funding projects created.
donations	*array* *It holds all the donations made to all the projects.*
projectsStartedByUser	*mapping* A mapping of address to an integer array. The array holds the unique integer id of all the projects created by the address.
donationsByUser	*mapping* A mapping of address to an integer array. The array holds the unique integer id of all the donations made by the address.
DonationsToAProject	*mapping* A mapping of an integer to integer array. It maps the unique integer id of a project to an array containing the unique integer id of the donations made to the project.

UserDonationsToAProject	*mapping* A mapping of an address to integer to integer array. It maps the address of a user to a project id to an integer array containing all the donation ids made by the user to that particular project.
projectsIdDonatedToByUser	*mapping* A mapping of an address to an integer array. It maps the address of a user to all the project ids donated to by the user.
totalEthRaised	*integer* It contains all the total funds raised in the contract
minimumPledgeAmount	*integer* *The minimum amount that can be donated. It can only be changed by the owner.*
noOfSuccessfulProjects	*integer* It contains the number of successful projects.
noOfCancelledProjects	*integer* It contains the number of cancelled projects.
owner	*address* It stores the address that deployed the contract.

Events

The following events are recommended to be included in the smart contract:

Events	**Purpose**
newProject	When a new project is created, it should emit the project id,the time created,the creator and the charity it is meant for.
newCharity	When a new charity is registered, it should emit the charity id,the address and other details.
donationAdded	When a donation is made to a project it should emit the donation id,project id, the amount,the donor and the time.
ProjectSuccessful	When funds have been successfully transfered,it should emit the project id,the amount transferred,the time and the charity id.
ProjectCancelled	When a project is cancelled, should emit the project id, the time and the charity Id.

Modifiers

The following modifiers is recommended to ensure proper restrictions on the smart contract functions:

Modifiers	**Purpose**
onlyOwner()	To check if the caller of the function is the current owner.
onlyProjectCreator()	To check if the address calling the function is the creator of the project.

Functions

The following functions should be present in the smart contract, The student should decide on the visibility of the functions and variables along with view and pure.

For Reference: https://solidity.readthedocs.io/en/v0.4.24/contracts.html

Constructor

Functions	**Purpose**
constructor	- Set the owner to the address deploying the contract

Other Functions

Functions	**Purpose**
registerCharity	- Passes the name, address and description of the new charity - Creates a struct Charity using the arguments to the function. - Pushes the new struct to the array charities - The push operation returns the length of the array from which one is subtracted to give the unique integer id for the charity. - Emit using the proper event.

createProject	- Passes the charity id,the integer percentToBeAdded, maxNoOfDays and the description. - Checks if the msg.value is greater than the minimum fund required - Creates a struct Project using the arguments to the function. - Pushes the new struct to the array projects. - The push operation returns the length of the array from which one is subtracted to give the unique integer id for the project. - The project id is mapped to the user's address using projectsStartedByUser. - If the integer _percentToBeAdded is 0,the fund is transferred to the charity without others contributing. - Emit using the proper event.
ContributeToProject	- Passes the project id of the project the donation is meant for - Creates a struct Donation using the arguments to the function. - Pushes the new struct to the array donations. - The push operation returns the length of the array from which one is subtracted to give the unique integer id for the donation. - The donation id is added to the user's list of donations. - The donation id is added to the list of donations to the project - The donation id is added to the user's list of donations to the project. - The project id is added to the list of projects donated to by user - If the total funds required for the project is completed or the maximum number of days have been reached then the funds are transferred and the project tagged successful. - Emit using the proper event.
CancelProject	- Passes the project id of the project. - Checks if the msg.sender is the creator of the project. - Checks if the project is active. - The project creator and all donors are refunded. - Tags the project inactive and unsuccessful. - Finally emit the event
adjustMinimumPledgeAmount	- Passes the new minimum amount - Checks if the msg.sender is the owner of the contract. - If so set the minimumPledgeAmount to the amount

	given in the argument.
generalInfo	- Returns the number of active projects,total number of projects,donations and charities.
returnRemainingFundsNeeded	-Passes the project id -returns the total eth remaining for the project to be completed.
returnCharityDetails	- It takes the charityId as an argument and returns all the details about the charity.
returnDonationDetails	- It takes the donationId as an argument and returns all the details about the donation.
returnProjectDetails	- It takes the projectId as an argument and returns all the details about the project.