CS 491: Charity Tracker

Chad Saltzman

21 November 2021

Contents

[Abstract 2](#_Toc88404251)

[Introduction 2](#_Toc88404252)

[Description 2](#_Toc88404253)

[Python Code Details 3](#_Toc88404254)

[Libraries 3](#_Toc88404255)

[Variables 3](#_Toc88404256)

[Operations 4](#_Toc88404257)

[Complete a donation 5](#_Toc88404258)

[View the status 6](#_Toc88404259)

[View the total amount donated 6](#_Toc88404260)

[Update Status 6](#_Toc88404261)

[Verify Charity Address 7](#_Toc88404262)

[Switch Charities 7](#_Toc88404263)

[Exit 7](#_Toc88404264)

[Solidity Code Details 7](#_Toc88404265)

[References 8](#_Toc88404266)

# Abstract

Charity fraud is a common issue where organizations masquerade themselves as a valid charity, but instead take and use the charity donations for their self-interest. In these situations, it is very difficult for charities to prove their authenticity to new donators. Currently the only way to check the authenticity of a charity is by verifying if the charity is registered with the government and has no fraud claims with the Federal Trade Commission. Applying blockchain technology to charities would resolve this issue by allowing charities to publicly display the supply chain process of all donations proving their legitimateness. Potential Donators can observe past donations and even new donations identifying what the contribution was used for or if it is still pending an application. The introduction of blockchain would create more transparency and encourage new donators to trust and commit to contributing to the proposed cause.

# Introduction

The Charity Tracker Application is designed to protect benefactors from contributing money to non-ethical charities and increase the reputation of charities. This is done through full transparency of the charity where most of the information related to the charity is publicly accessible allowing people to perform their own investigations on the validity of the charity. The blockchain technology allows for this transparency to occur when every donation is processed as a transaction and is verified and added to the blockchain. With the blockchain being publicly viewable, contributors will be allowed to track what their donation has been used for and/or verify the integrity of the charity. Smart Contracts will be used to integrate with the blockchain to update statuses and retrieve information about the charity and donations.

# Description

The Charity Tracker application was designed using the programming languages python and solidity. Python was used for the core application and Solidity was implemented to incorporate smart contracts on the blockchain. To run this application, one must first deploy the smart contract. The parameters to deploy the smart contract is going to be an array containing the names of all the charities, and then another array holding the addresses of those charities. Once deployed, the contract address and the abi of the Ethereum Virtual Machine must be copied and inserted into python file to link the smart contract to the source code. Once these steps have been completed the application can be ran by executing the command python3 CharityTracker.py

# Python Code Details

## Libraries

As seen in Figure 1, the libraries used in this application are json, web3, sys, and getpass. The json library is used to parse the abi and convert it into a python dictionary. The web3 library is used to interact with both the Ganache network and the smart contracts. Sys is implemented to end the program when the user has finished accessing the application. Getpass is used as a secure way to retrieve the private key from the user for their Ethereum account.

import json

from web3 import Web3

import sys

from getpass import getpass

Figure 1: This figure displays the libraries that were imported.

## Variables

Figure 2 shows the static variables that are used multiple times throughout the application. These variables are ganache\_url, address, web3, contract, blockchainOptions and charityList. The ganache\_url variable is used by the Web3 library to create a web3 object linked to the ganache Ethereum account which contains the many accounts used. The web3 object allows for interaction between the ganache network and the smart contracts deployed. The contract variable holds the address to the deployed smart contract. BlockchainOptions is a list containing the different choices that the user can select including “Complete a donation” and “View the status of a donation”. The charityList variable holds the details for each charity including the charity names and their respective addresses. In the future I would like to make a smart contract to retrieve the charity details so that it is not explicitly available in the source code.

ganache\_url = "http://127.0.0.1:7545"

    web3 = Web3(Web3.HTTPProvider(ganache\_url))

    abi = json.loads('[{"inputs":[{"internalType":"string[]","name":"CharityName","type":"string[]"},{"internalType":"address[]","name":"CharityAdd","type":"address[]"}],"stateMutability":"payable","type":"constructor"},{"inputs":[{"internalType":"uint256","name":"num","type":"uint256"},{"internalType":"string","name":"trx","type":"string"},{"internalType":"string","name":"CharityName","type":"string"}],"name":"contributeDonation","outputs":[],"stateMutability":"nonpayable","type":"function"},{"inputs":[{"internalType":"address","name":"UserAddress","type":"address"}],"name":"getBalance","outputs":[{"internalType":"uint256","name":"","type":"uint256"}],"stateMutability":"view","type":"function"},{"inputs":[{"internalType":"string","name":"searchCharityName","type":"string"}],"name":"getCharityAddress","outputs":[{"internalType":"address","name":"tempAdd","type":"address"}],"stateMutability":"view","type":"function"},{"inputs":[{"internalType":"string","name":"trx","type":"string"}],"name":"getStatus","outputs":[{"internalType":"string","name":"status","type":"string"}],"stateMutability":"view","type":"function"},{"inputs":[{"internalType":"string","name":"searchCharityName","type":"string"}],"name":"getTotalDonated","outputs":[{"internalType":"uint256","name":"runningTotal","type":"uint256"}],"stateMutability":"view","type":"function"},{"inputs":[{"internalType":"string","name":"newStatus","type":"string"},{"internalType":"string","name":"trx","type":"string"}],"name":"updateStatus","outputs":[],"stateMutability":"nonpayable","type":"function"}]')

    address = web3.toChecksumAddress("0xe0d7A1Ae367010F6Ea2512300f2196c70cB64A69")  # Deployed Contract's address

    contract = web3.eth.contract(address=address, abi=abi)  # Creates an object for the contract.

    charityList = {

        "Feeding America": "0x4c2b2972B316d77Ad6C2be9AAa6AbCd845FFB38a",

        "American Red Cross": "0x5D42f081Af27233Ab7ad8fE7039c6B145440d055",

        "St. Jude Children's Research Hospital": "0xDB57c099b403C62388Dfd6a5E5e02fCE83a656E3"

    }

    blockchainOptions = ["Complete a donation", "View the status of a donation", "View the total amount donated", "Update Status", "Verify Charity Address", "Choose a different charity", "Exit"]

Figure 2: This figure displays the core variables used throughout the application.

## Operations

The first choice the user is given is what charity they would like to select. Once selecting the charity, the user can always go back and switch between the provided charities, but after selecting one, all future operations will be directly correlated to the charity selected. An example of the charity selection screen is provided in Figure 3.

Choose a charity from the list below:

1. Feeding America

2. American Red Cross

3. St. Jude Children's Research Hospital

Figure 3: This figure displays the options of charity provided to the user.

After selecting a charity, the user will be prompted with a list of operations as detailed in Figure 4. For each operation the user will be providing additional information to fullfill the operation detailed such as providing donation amount, charity name, or the transaction address that they are looking for a status on. After the user has completed any operations that they would like, they are able to exit the application by inputting the value 7 into the terminal which will close the application.

Welcome to the Feeding America Charity

choose from the following operations:

1. Complete a donation

2. View the status of a donation

3. View the total amount donated

4. Update Status

5. Verify Charity Address

6. Choose a different charity

7. Exit

Figure 4: This figure displays the different operations that the user can choose to interact with a charity

### Complete a donation

The contribute a donation operation gathers the donation amount, user’s Ethereum wallet address, and their private key to process the transaction. Once the information is gathered a transaction is executed from the user’s account to the charity account for the amount specified. The “tx” variable specifies the transaction data and the “signed\_tx” variable is the execution of the transaction on the blockchain with the transaction and user’s signature. Once the transaction is completed, the user will be given a transaction hash which can be used to check the status of their donation. Finally, a call transaction to the smart contract must be executed to transfer the donation data to the contract to account for the statuses of the donation and the total donation of the charity selected as shown in Figure 5.

donationValue = int(input("Enter in the value of your donation (Ether):"))

            account = input("Enter in your Ethereum Address: ")

            priv\_key = getpass("enter in your private key: ")

            nonce = web3.eth.getTransactionCount(account)

            tx = {          #Need to verify the donater has enough ether to donate.

                'nonce': nonce,

                'to': charityList[list(charityList)[charityValue - 1]],

                'value': web3.toWei(donationValue,'ether'),

                'gas': 2000000,

                'data': b"",

                'gasPrice': web3.toWei('50', 'gwei')

            }

            signed\_tx = web3.eth.account.signTransaction(tx, priv\_key)

            tx\_hash = web3.toHex(web3.eth.sendRawTransaction(signed\_tx.rawTransaction))

            print(f"You can track the status of this donation with the following hash code: {tx\_hash}")

            print(web3.toWei(donationValue,'ether'))

            contract.functions.contributeDonation(web3.toWei(donationValue,'ether'), tx\_hash,list(charityList)[charityValue - 1]).transact()

Figure 5: This figure displays the source code that executes completing a transaction and logging the transaction data into the smart contract.

### View the status

The view status operation allows the user to input the transaction hash returned from their donation and retrieve the status of that donation. The status right after the donation will be set to “Initial Donation” and from there can be updated by the charity administrators. A call to the smart contract is performed to retrieve this information as shown in figure 6.

tx = str(input("What is the transaction hash of your donation: "))

            status = contract.functions.getStatus(tx).call()

            print(f"the status of the transaction {tx} is: {status}")

Figure 6: this figure displays the source code that retrieves the status of a donation

### View the total amount donated

The view total amount donated operation allows the user to see the total contributions to the charity selected. This is retrieved through a call to the smart contract which stores this information as shown in Figure 7.

balance = contract.functions.getTotalDonated(list(charityList)[charityValue -1]).call()

            print("\nTotal donated = " + str(web3.fromWei(balance, "ether")) + " ether")

Figure 7: this figure displays the source code that retrieves the total amount donated to the selected charity.

### Update Status

The update status operation allows the status of the donation to be updated. Currently anyone can update the status of a donation which is a current flaw in the project. In the future I would like to implement an administrator role with executive privileges and only users with the administrator role can make this change. The change is completed through a transaction with the smart contract as shown in Figure 8.

tx = input("Enter in the hash associated with the donation you would like to update: ")

            status = input("What is the updated status for this transaction: ")

            tx\_hash = contract.functions.updateStatus(status,tx).transact()

            print(f"Status has been updated. The transaction hash for this is: {tx\_hash}")

Figure 8: This figure displays the source code that updates the status of a donation.

### Verify Charity Address

The verify charity address operation returns to the terminal the Ethereum address associated with the charity so that the user can verify the validity of this address and look at the chain of transactions performed to and from this account. Therefore, if the status of their donation shows as delivered, then they should be able to see the transaction leaving the charity’s account and it going to a new account. Figure 9 details the process of retrieving the address of the charity from the smart contract.

address = contract.functions.getCharityAddress(list(charityList)[charityValue - 1]).call()

            print(f"The ethereum address of {list(charityList)[charityValue - 1]} is: {address}")

Figure 9: This figure displays the source code that retrieves the address of the selected charity.

### Switch Charities

The switch charities operations return the user back to the main menu where they get to select from the list of predefined charities. Once the user selects one of the charities, they are returned to the operation menu where they can interact with that charity.

### Exit

The exit operation terminates the application.

# Solidity Code Details

# References

1. <https://web3py.readthedocs.io/en/stable/quickstart.html>
2. <https://docs.soliditylang.org/en/v0.8.10/>
3. <https://www.youtube.com/watch?v=p5W67zTQFRM&t=627s&ab_channel=DappUniversity>