SOLIDITY:

pragma solidity ^0.5.1;

contract MintYourCoins {

address public minter;

mapping(address => uint) public balance;

event Send(address indexed from, address indexed to, uint amount);

constructor() public {

minter = msg.sender;

}

function mint(address receiver, uint amount) public {

require(msg.sender == minter, "Only minter can mint");

require(amount < 1e60, "Amount exceeds limit");

balance[receiver] += amount;

}

function send(address receiver, uint amount) public {

require(amount <= balance[msg.sender], "Insufficient Balance");

balance[msg.sender] -= amount;

balance[receiver] += amount;

emit Send(msg.sender, receiver, amount);

}

function getBalance(address \_account) external view returns (uint) {

return balance[\_account];

}

}

from tkinter import \*

from web3 import Web3

import json

# Replace these values with your actual contract address and ABI

contract\_address = "0x7430D84933f8f48e176E0CF2c54D1e8F32F9CC7d"

contract\_abi = json.load(open('abi.json'))

# Web3 connection

w3 = Web3(Web3.HTTPProvider('http://127.0.0.1:7545')) # Replace with your Ganache RPC server link

contract = w3.eth.contract(address=contract\_address, abi=contract\_abi)

# Tkinter app

root = Tk()

root.title("Smart Contract Interaction")

# Functions

import time

def mint\_coins():

receiver = receiver\_entry.get()

amount = int(amount\_entry.get())

sender\_address = w3.eth.accounts[0] # Assuming you have unlocked your account

w3.eth.default\_account = sender\_address

tx\_hash = contract.functions.mint(receiver, amount).transact({'from': sender\_address})

# Wait for the transaction to be mined

receipt = w3.eth.wait\_for\_transaction\_receipt(tx\_hash)

update\_balance()

# Wait for the transaction to be mined

while True:

receipt = w3.eth.get\_transaction\_receipt(tx\_hash)

if receipt is not None:

break

time.sleep(1) # Adjust the delay as needed

update\_balance()

def send\_coins():

receiver = receiver\_entry.get()

amount = int(amount\_entry.get())

sender\_address = w3.eth.accounts[0] # Assuming you have unlocked your account

w3.eth.default\_account = sender\_address

tx\_hash = contract.functions.send(receiver, amount).transact({'from': sender\_address})

# Wait for the transaction to be mined

receipt = w3.eth.wait\_for\_transaction\_receipt(tx\_hash)

update\_balance()

# Wait for the transaction to be mined

while True:

receipt = w3.eth.get\_transaction\_receipt(tx\_hash)

if receipt is not None:

break

time.sleep(1) # Adjust the delay as needed

update\_balance()

def update\_balance():

account = w3.eth.accounts[0] # Assuming you're using the first account

balance = contract.functions.getBalance(account).call()

balance\_label.config(text=f"Your Balance: {balance} coins")

# GUI Elements

receiver\_label = Label(root, text="Receiver Address:")

receiver\_label.pack()

receiver\_entry = Entry(root)

receiver\_entry.pack()

amount\_label = Label(root, text="Amount:")

amount\_label.pack()

amount\_entry = Entry(root)

amount\_entry.pack()

mint\_button = Button(root, text="Mint Coins", command=mint\_coins)

mint\_button.pack()

send\_button = Button(root, text="Send Coins", command=send\_coins)

send\_button.pack()

balance\_label = Label(root, text="")

balance\_label.pack()

update\_balance()

root.mainloop()

[

{

"constant": false,

"inputs": [

{

"internalType": "address",

"name": "receiver",

"type": "address"

},

{

"internalType": "uint256",

"name": "amount",

"type": "uint256"

}

],

"name": "mint",

"outputs": [],

"payable": false,

"stateMutability": "nonpayable",

"type": "function"

},

{

"constant": false,

"inputs": [

{

"internalType": "address",

"name": "receiver",

"type": "address"

},

{

"internalType": "uint256",

"name": "amount",

"type": "uint256"

}

],

"name": "send",

"outputs": [],

"payable": false,

"stateMutability": "nonpayable",

"type": "function"

},

{

"inputs": [],

"payable": false,

"stateMutability": "nonpayable",

"type": "constructor"

},

{

"anonymous": false,

"inputs": [

{

"indexed": true,

"internalType": "address",

"name": "from",

"type": "address"

},

{

"indexed": true,

"internalType": "address",

"name": "to",

"type": "address"

},

{

"indexed": false,

"internalType": "uint256",

"name": "amount",

"type": "uint256"

}

],

"name": "Send",

"type": "event"

},

{

"constant": true,

"inputs": [

{

"internalType": "address",

"name": "",

"type": "address"

}

],

"name": "balance",

"outputs": [

{

"internalType": "uint256",

"name": "",

"type": "uint256"

}

],

"payable": false,

"stateMutability": "view",

"type": "function"

},

{

"constant": true,

"inputs": [

{

"internalType": "address",

"name": "\_account",

"type": "address"

}

],

"name": "getBalance",

"outputs": [

{

"internalType": "uint256",

"name": "",

"type": "uint256"

}

],

"payable": false,

"stateMutability": "view",

"type": "function"

},

{

"constant": true,

"inputs": [],

"name": "minter",

"outputs": [

{

"internalType": "address",

"name": "",

"type": "address"

}

],

"payable": false,

"stateMutability": "view",

"type": "function"

}

]