**SOURCE CODE**

from django.shortcuts import render

from django.template import RequestContext

from django.contrib import messages

from django.http import HttpResponse

from django.conf import settings

from datetime import date

import json

from web3 import Web3, HTTPProvider

global details

def readDetails():

global details

blockchain\_address = 'http://127.0.0.1:9545' #Blokchain connection IP

web3 = Web3(HTTPProvider(blockchain\_address))

web3.eth.defaultAccount = web3.eth.accounts[0]

compiled\_contract\_path = 'ForensicEvidenceContract.json' #forensic contract code

deployed\_contract\_address = '0xdbdeB1d0fe02FBE6a05f5C2194e187D3A92F2229' #hash address to access forensiccontract

with open(compiled\_contract\_path) as file:

contract\_json = json.load(file) # load contract info as JSON

contract\_abi = contract\_json['abi'] # fetch contract's abi - necessary to call its functions

file.close()

contract = web3.eth.contract(address=deployed\_contract\_address, abi=contract\_abi) #now calling contract to access data

details = contract.functions.getData().call()

if len(details) > 0:

if 'empty' in details: details = details[5:len(details)]

def saveDataBlockChain(currentData):

global details

global contract

blockchain\_address = 'http://127.0.0.1:9545'

web3 = Web3(HTTPProvider(blockchain\_address))

web3.eth.defaultAccount = web3.eth.accounts[0]

compiled\_contract\_path = 'ForensicEvidenceContract.json'

deployed\_contract\_address = '0xdbdeB1d0fe02FBE6a05f5C2194e187D3A92F2229'

with open(compiled\_contract\_path) as file:

contract\_json = json.load(file) # load contract info as JSON

contract\_abi = contract\_json['abi'] # fetch contract's abi - necessary to call its functions

file.close()

contract = web3.eth.contract(address=deployed\_contract\_address, abi=contract\_abi)

readDetails()

details+=currentData

msg = contract.functions.setEvidenceDetails(details).transact()

tx\_receipt = web3.eth.waitForTransactionReceipt(msg)

def index(request):

if request.method == 'GET':

return render(request, 'index.html', {})

def Admin(request):

if request.method == 'GET':

return render(request, 'Admin.html', {})

def AddEvidence(request):

if request.method == 'GET':

return render(request, 'AddEvidence.html', {})

def AdminLogin(request):

if request.method == 'POST':

username = request.POST.get('t1', False)

password = request.POST.get('t2', False)

if username == 'admin' and password == 'admin':

context= {'data':'welcome '+username}

return render(request, 'AdminScreen.html', context)

else:

context= {'data':'login failed'}

return render(request, 'Admin.html', context)

def ViewEvidence(request):

if request.method == 'GET':

global details

readDetails()

print("p det "+details)

arr = details.split("\n")

output = ''

font = "<font size=3 color=black>"

for i in range(len(arr)-1):

array = arr[i].split("$");

output+="<tr><td>"+font+array[0]+"</td>"

output+="<td>"+font+array[1]+"</td>"

output+="<td>"+font+array[2]+"</td>"

output+="<td>"+font+array[3]+"</td>"

output+="<td>"+font+array[4]+"</td>"

output+="<td>"+font+array[5]+"</td>"

output+="<td>"+font+array[6]+"</td>"

context= {'data':output}

return render(request, 'ViewEvidence.html', context)

def AddEvidenceAction(request):

if request.method == 'POST':

rid = request.POST.get('t1', False)

crime\_type = request.POST.get('t2', False)

desc = request.POST.get('t3', False)

evidence = request.POST.get('t4', False)

area = request.POST.get('t5', False)

witness = request.POST.get('t6', False)

today = date.today()

data = rid+"$"+crime\_type+"$"+desc+"$"+evidence+"$"+area+"$"+witness+"$"+str(today)+"\n"

saveDataBlockChain(data)

context= {'data':'Evidence details saved in Blockchain'}

return render(request, 'AddEvidence.html', context)