**SOURCE CODE**

**MANAGE.PY**

#!/usr/bin/env python

import os

import sys

if \_name\_ == '\_main\_':

os.environ.setdefault('DJANGO\_SETTINGS\_MODULE', 'StudentCredential.settings')

try:

from django.core.management import execute\_from\_command\_line

except ImportError as exc:

raise ImportError(

"Couldn't import Django. Are you sure it's installed and "

"available on your PYTHONPATH environment variable? Did you "

"forget to activate a virtual environment?"

) from exc

execute\_from\_command\_line(sys.argv)

**SETTINGS.PY**

"""

Django settings for StudentCredential project.

Generated by 'django-admin startproject' using Django 2.1.7.

For more information on this file, see

https://docs.djangoproject.com/en/2.1/topics/settings/

For the full list of settings and their values, see

https://docs.djangoproject.com/en/2.1/ref/settings/

"""

import os

# Build paths inside the project like this: os.path.join(BASE\_DIR, ...)

BASE\_DIR = os.path.dirname(os.path.dirname(os.path.abspath(\_file\_)))

# Quick-start development settings - unsuitable for production

# See https://docs.djangoproject.com/en/2.1/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!

SECRET\_KEY = 'q0xb&&4w\*p+0d3jj26yt142$5w-\*ps1@3y6te!b^u-jhefwt1='

# SECURITY WARNING: don't run with debug turned on in production!

DEBUG = True

ALLOWED\_HOSTS = []

# Application definition

INSTALLED\_APPS = [

'django.contrib.admin',

'django.contrib.auth',

'django.contrib.contenttypes',

'django.contrib.sessions',

'django.contrib.messages',

'django.contrib.staticfiles',

'StudentCredentialApp'

]

MIDDLEWARE = [

'django.middleware.security.SecurityMiddleware',

'django.contrib.sessions.middleware.SessionMiddleware',

'django.middleware.common.CommonMiddleware',

'django.middleware.csrf.CsrfViewMiddleware',

'django.contrib.auth.middleware.AuthenticationMiddleware',

'django.contrib.messages.middleware.MessageMiddleware',

'django.middleware.clickjacking.XFrameOptionsMiddleware',

]

ROOT\_URLCONF = 'StudentCredential.urls'

TEMPLATES = [

{

'BACKEND': 'django.template.backends.django.DjangoTemplates',

'DIRS': [

os.path.join('E:/takeoff/oct23/CredentialVerification/StudentCredentialApp', 'templates'),

],

'APP\_DIRS': True,

'OPTIONS': {

'context\_processors': [

'django.template.context\_processors.debug',

'django.template.context\_processors.request',

'django.contrib.auth.context\_processors.auth',

'django.contrib.messages.context\_processors.messages',

],

},

},

]

WSGI\_APPLICATION = 'StudentCredential.wsgi.application'

# Database

# https://docs.djangoproject.com/en/2.1/ref/settings/#databases

DATABASES = {

'default': {

'ENGINE': 'django.db.backends.sqlite3',

'NAME': os.path.join(BASE\_DIR, 'db.sqlite3'),

}

}

# Password validation

# https://docs.djangoproject.com/en/2.1/ref/settings/#auth-password-validators

AUTH\_PASSWORD\_VALIDATORS = [

{

'NAME': 'django.contrib.auth.password\_validation.UserAttributeSimilarityValidator',

},

{

'NAME': 'django.contrib.auth.password\_validation.MinimumLengthValidator',

},

{

'NAME': 'django.contrib.auth.password\_validation.CommonPasswordValidator',

},

{

'NAME': 'django.contrib.auth.password\_validation.NumericPasswordValidator',

},

]

# Internationalization

# https://docs.djangoproject.com/en/2.1/topics/i18n/

LANGUAGE\_CODE = 'en-us'

TIME\_ZONE = 'UTC'

USE\_I18N = True

USE\_L10N = True

USE\_TZ = True

# Static files (CSS, JavaScript, Images)

# https://docs.djangoproject.com/en/2.1/howto/static-files/

STATIC\_URL = '/static/'

**VIEWS.PY**

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

import os

import json

from web3 import Web3, HTTPProvider

import hashlib

import os

from django.core.files.storage import FileSystemStorage

import pickle

global details, username, school\_name, company\_name

details=''

global contract

def readDetails(contract\_type):

global details

details = ""

print(contract\_type+"======================")

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 = 'Student.json' #student contract code

deployed\_contract\_address = '0xd374Cb05bd6187D6cF905D7bBD85f2b704fBDD29' #hash address to access student contract

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

if contract\_type == 'schoolcompany':

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

if contract\_type == 'enrollstudent':

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

if contract\_type == 'credential':

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

if contract\_type == 'accessrequest':

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

print(details)

def saveDataBlockChain(currentData, contract\_type):

global details

global contract

details = ""

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

web3 = Web3(HTTPProvider(blockchain\_address))

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

compiled\_contract\_path = 'Student.json' #student contract file

deployed\_contract\_address = '0xd374Cb05bd6187D6cF905D7bBD85f2b704fBDD29' #contract address

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(contract\_type)

if contract\_type == 'schoolcompany':

details+=currentData

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

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

if contract\_type == 'enrollstudent':

details+=currentData

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

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

if contract\_type == 'credential':

details+=currentData

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

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

if contract\_type == 'accessrequest':

details+=currentData

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

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

def saveDataBlockChain1(currentData, contract\_type):

global details

global contract

details = ""

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

web3 = Web3(HTTPProvider(blockchain\_address))

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

compiled\_contract\_path = 'Student.json' #student contract file

deployed\_contract\_address = '0xd374Cb05bd6187D6cF905D7bBD85f2b704fBDD29' #contract address

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)

if contract\_type == 'credential':

msg = contract.functions.setCredentialData(currentData).transact()

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

def StudentLoginAction(request):

if request.method == 'POST':

global details, username

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

readDetails('enrollstudent')

arr = details.split("\n")

status = "none"

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

array = arr[i].split("#")

if array[1] == username:

status = "success"

break

if status == "success":

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

return render(request, "StudentScreen.html", context)

else:

context= {'data':'Invalid username'}

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

def CompanyLoginAction(request):

if request.method == 'POST':

global details, username, school\_name, company\_name

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

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

readDetails('schoolcompany')

arr = details.split("\n")

status = "none"

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

array = arr[i].split("#")

if array[0] == 'company' and array[4] == username and array[5] == password:

status = "success"

company\_name = array[1]

break

if status == "success":

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

return render(request, "CompanyScreen.html", context)

else:

context= {'data':'Invalid username'}

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

def UniversityLoginAction(request):

if request.method == 'POST':

global details, username, school\_name, company\_name

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

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

readDetails('schoolcompany')

arr = details.split("\n")

status = "none"

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

array = arr[i].split("#")

print(array)

if array[0] == 'university' and array[4] == username and array[5] == password:

status = "success"

school\_name = array[1]

break

if status == "success":

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

return render(request, "UniversityScreen.html", context)

else:

context= {'data':'Invalid username'}

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

def UploadCertificate(request):

if request.method == 'GET':

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

def UpdateCertificate(request):

if request.method == 'GET':

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

def index(request):

if request.method == 'GET':

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

def EnrollStudent(request):

if request.method == 'GET':

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

def UniversityLogin(request):

if request.method == 'GET':

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

def CompanyLogin(request):

if request.method == 'GET':

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

def StudentLogin(request):

if request.method == 'GET':

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

def UniversitySignup(request):

if request.method == 'GET':

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

def CompanySignup(request):

if request.method == 'GET':

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

def UniversitySignupAction(request):

if request.method == 'POST':

global details

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

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

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

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

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

readDetails('schoolcompany')

arr = details.split("\n")

status = "none"

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

array = arr[i].split("#")

if array[4] == user:

status = user+" Username already exists"

break

if status == "none":

data = "university#"+school+"#"+address+"#"+contact+"#"+user+"#"+password+"\n"

saveDataBlockChain(data,"schoolcompany")

context = {"data":"University signup task completed"}

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

else:

context = {"data":status}

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

def CompanySignupAction(request):

if request.method == 'POST':

global details

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

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

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

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

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

readDetails('schoolcompany')

arr = details.split("\n")

status = "none"

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

array = arr[i].split("#")

if array[4] == user:

status = user+" Username already exists"

break

if status == "none":

data = "company#"+company+"#"+address+"#"+contact+"#"+user+"#"+password+"\n"

saveDataBlockChain(data,"schoolcompany")

context = {"data":"Company signup task completed"}

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

else:

context = {"data":status}

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

def ViewStudents(request):

if request.method == 'GET':

global details, username, school\_name

output = '<table border=1 align=center width=100%>'

font = '<font size="" color="white">'

arr = ['University Name','Student ID','Student Name','University Details','Course Name','Joining Date']

output += "<tr>"

for i in range(len(arr)):

output += "<th>"+font+arr[i]+"</th>"

readDetails('enrollstudent')

arr = details.split("\n")

status = "none"

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

print(arr[i])

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>"

context= {'data':output}

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

def EnrollStudentAction(request):

if request.method == 'POST':

global details, username, school\_name

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

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

school\_details = request.POST.get('t3', False)

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

joining\_date = request.POST.get('t5', False)

readDetails('enrollstudent')

arr = details.split("\n")

status = "none"

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

array = arr[i].split("#")

if array[0] == sid:

status = sid+" Student ID already exists"

break

if status == "none":

details = ""

data = school\_name+"#"+sid+"#"+sname+"#"+school\_details+"#"+course+"#"+joining\_date+"\n"

saveDataBlockChain(data,"enrollstudent")

context = {"data":"New Student Enrollment Task Completed"}

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

else:

context = {"data":status}

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

def UploadCertificateAction(request):

if request.method == 'POST':

global details, username, school\_name

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

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

issue\_date = request.POST.get('t3', False)

filename = request.FILES['t4'].name

myfile = request.FILES['t4'].read()

'''

myfile = pickle.dumps(myfile)

hashcode = api.add\_pyobj(myfile)

#readDetails('credential')

'''

readDetails('credential')

credential = details.split("\n")

certificate\_id = len(credential)

#if certificate\_id > 1:

#certificate\_id = certificate\_id + 1

result = hashlib.sha256(myfile)

hashcode = result.hexdigest()

with open('StudentCredentialApp/static/certificates/'+str(certificate\_id)+".png", "wb") as file:

file.write(myfile)

file.close()

data = school\_name+"#"+str(certificate\_id)+"#"+sid+"#"+certificate+"#"+issue\_date+"#"+filename+"#"+hashcode+"\n"

saveDataBlockChain(data,"credential")

context = {"data":"Certificate saved with hashcode saving in Blockchain: "+hashcode}

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

def UpdateCertificateAction(request):

if request.method == 'POST':

global details, username, school\_name

cid = request.POST.get('certificate', False)

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

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

issue\_date = request.POST.get('t3', False)

filename = request.FILES['t4'].name

myfile = request.FILES['t4'].read()

result = hashlib.sha256(myfile)

hashcode = result.hexdigest()

if os.path.exists('StudentCredentialApp/static/certificates/'+cid+".png"):

os.remove('StudentCredentialApp/static/certificates/'+cid+".png")

with open('StudentCredentialApp/static/certificates/'+cid+".png", "wb") as file:

file.write(myfile)

file.close()

'''

myfile = pickle.dumps(myfile)

hashcode = api.add\_pyobj(myfile)

'''

readDetails('credential')

data = ""

credential = details.split("\n")

flag = False

output = "Given Certificate or Student Id doesn't match"

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

array = credential[i].split("#")

if array[1] != cid:

data += credential[i]+"\n"

if array[1] == cid:

flag = True

if flag == True:

data += school\_name+"#"+cid+"#"+sid+"#"+certificate+"#"+issue\_date+"#"+filename+"#"+hashcode+"\n"

saveDataBlockChain1(data,"credential")

output = "Modified certificate saved with hashcode saving in Blockchain: "+hashcode

context = {"data":output}

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

def SendAccessRequest(request):

if request.method == 'GET':

global details, username, school\_name

output = '<table border=1 align=center width=100%>'

font = '<font size="" color="white">'

arr = ['School Name','Student ID','Student Name','School Details','Course Name','Joining Date','Send Access Request']

output += "<tr>"

for i in range(len(arr)):

output += "<th>"+font+arr[i]+"</th>"

readDetails('enrollstudent')

arr = details.split("\n")

status = "none"

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><a href=\'SendRequest?t1='+array[1]+'\'><font size=3 color=white>Click Here</font></a></td></tr>'

context= {'data':output}

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

def SendRequest(request):

if request.method == 'GET':

global details, username, company\_name

sid = request.GET.get('t1', False)

data = sid+"#"+company\_name+"#Pending\n"

saveDataBlockChain(data,"accessrequest")

context = {"data":"Access Request sent to student: "+sid}

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

def checkAccess(credential\_arr, sid):

status = False

for i in range(len(credential\_arr)-1):

arr = credential\_arr[i].split("#")

if arr[0] == sid and arr[1] == company\_name and arr[2] == "Accepted":

status = True

break

return status

def AccessCertificate(request):

if request.method == 'GET':

global details, username, company\_name

output = '<table border=1 align=center width=100%>'

font = '<font size="" color="white">'

arr = ['University Name','Certificate ID', 'Student ID','Certificate Details','Issue Date','File Name','Hashcode','Certificate']

output += "<tr>"

for i in range(len(arr)):

output += "<th>"+font+arr[i]+"</th>"

readDetails('accessrequest')

credential\_arr = details.split("\n")

readDetails('credential')

access\_certificate = details.split("\n")

for i in range(len(access\_certificate)-1):

array = access\_certificate[i].split("#")

status = checkAccess(credential\_arr, array[2])

if status == True:

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>"

'''

content = api.get\_pyobj(array[5])

content = pickle.loads(content)

'''

output+='<td><img src=static/certificates/'+array[1]+'.png width=400 height=400></img></td>'

context= {'data':output}

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

def ViewDetails(request):

if request.method == 'GET':

global details, username

output = '<table border=1 align=center width=100%>'

font = '<font size="" color="white">'

arr = ['University Name','Student ID','Student Name','School Details','Course Name','Joining Date']

output += "<tr>"

for i in range(len(arr)):

output += "<th>"+font+arr[i]+"</th>"

readDetails('enrollstudent')

arr = details.split("\n")

status = "none"

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

array = arr[i].split("#")

if array[1] == username:

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>"

context= {'data':output}

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

def AccessOwnCertificate(request):

if request.method == 'GET':

global details, username

output = '<table border=1 align=center width=100%>'

font = '<font size="" color="white">'

arr = ['University Name','Certificate ID', 'Student ID','Certificate Details','Issue Date','File Name','Hashcode','Certificate']

output += "<tr>"

for i in range(len(arr)):

output += "<th>"+font+arr[i]+"</th>"

readDetails('credential')

access\_certificate = details.split("\n")

for i in range(len(access\_certificate)-1):

array = access\_certificate[i].split("#")

if array[2] == username:

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>"

'''

content = api.get\_pyobj(array[5])

content = pickle.loads(content)

if os.path.exists('StudentCredentialApp/static/certificates/'+array[4]):

os.remove('StudentCredentialApp/static/certificates/'+array[4])

with open('StudentCredentialApp/static/certificates/'+array[4], "wb") as file:

file.write(content)

file.close()

'''

output+='<td><img src=static/certificates/'+array[1]+'.png width=400 height=400></img></td>'

context= {'data':output}

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

def GrantAccess(request):

if request.method == 'GET':

global details, username

output = '<table border=1 align=center width=100%>'

font = '<font size="" color="white">'

arr = ['Student ID','Company Name','Access Status','Grant Access']

output += "<tr>"

for i in range(len(arr)):

output += "<th>"+font+arr[i]+"</th>"

readDetails('accessrequest')

arr = details.split("\n")

status = "none"

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

array = arr[i].split("#")

if array[0] == username:

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

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

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

output+='<td><a href=\'GrantAccessAction?t1='+array[0]+'&t2='+array[1]+'\'><font size=3 color=white>Click Here</font></a></td></tr>'

context= {'data':output}

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

def GrantAccessAction(request):

if request.method == 'GET':

global details, username, company\_name

sid = request.GET.get('t1', False)

company = request.GET.get('t2', False)

data = ""

readDetails('accessrequest')

arr = details.split("\n")

status = "none"

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

array = arr[i].split("#")

if array[0] != sid and array[1] != company:

data+=array[0]+"#"+array[1]+"#"+array[2]+"\n"

data+=sid+"#"+company+"#Accepted\n"

saveDataBlockChain(data,"accessrequest")

context = {"data":"Access Request granted to company: "+company}

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