**SAMPLE CODE**

url.py

from django.contrib import admin

from django.urls import path

from django.conf import settings

from django.conf.urls.static import static

from users import views

from users.views import index,userlogin,adminlogin,cloudlogin,userregister,storeregistration,logout,userlogincheck,usercreateapp,appcreaterequest,useruploadfile,snippet\_detail

from admins.views import adminlogincheck,adminactivateusers,activatewaitedusers

from clouds.views import activateuserapp,cloudlogincheck,clouduserappactivations

from .views import resturl,downloadfile,deletefile,uploadfile

urlpatterns = [

path('admin/', admin.site.urls),

path('',index,name='index'),

path(r'accounts', views.AccountAPIView.as\_view(), name='account-list'),

path(r'contacts', views.ContactAPIView.as\_view(), name='contact-list'),

path(r'activities', views.ActivityAPIView.as\_view(), name='activity-list'),

path(r'activitystatuses', views.ActivityStatusAPIView.as\_view(), name='activity-status-list'),

path(r'contactsources', views.ContactSourceAPIView.as\_view(), name='contact-source-list'),

path(r'contactstatuses', views.ContactStatusAPIView.as\_view(), name='contact-status-list'),

path(r'logout',logout,name='logout'),

path(r'adminlogincheck',adminlogincheck,name='adminlogincheck'),

path(r'adminactivateusers',adminactivateusers,name='adminactivateusers'),

path(r'activatewaitedusers/<id>/$',activatewaitedusers,name='activatewaitedusers'),

path(r'userlogin',userlogin,name='userlogin'),

path(r'adminlogin', adminlogin, name='adminlogin'),

path(r'cloudlogin', cloudlogin, name='cloudlogin'),

path(r'userregister', userregister, name='userregister'),

path(r'storeregistration',storeregistration,name='storeregistration'),

path(r'userlogincheck', userlogincheck, name='userlogincheck'),

path(r'usercreateapp',usercreateapp,name='usercreateapp'),

path(r'appcreaterequest',appcreaterequest,name='appcreaterequest'),

path(r'useruploadfile/<appname>/$',useruploadfile,name='useruploadfile'),

path(r'^snippet\_detail/$',snippet\_detail,name='snippet\_detail'),

path(r'resturl/<id>',resturl,name='resturl'),

path(r'downloadfile/<id>',downloadfile,name='downloadfile'),

path(r'deletefile/<id>',deletefile,name='deletefile'),

path(r'uploadfile',uploadfile,name='uploadfile'),

path(r'activateuserapp',activateuserapp,name='activateuserapp'),

path(r'cloudlogincheck',cloudlogincheck,name='cloudlogincheck'),

path(r'clouduserappactivations/<appname>/$',clouduserappactivations, name='clouduserappactivations'),

]

if settings.DEBUG:

urlpatterns += static(settings.MEDIA\_URL,document\_root=settings.MEDIA\_ROOT)

Cloud Side views.py

from django.shortcuts import render,HttpResponse

from rest\_framework.views import APIView

from rest\_framework.decorators import api\_view

from rest\_framework import generics

from users.models import UserFileModel

import os

from django.conf import settings

from django.contrib import messages

from rest\_framework import status

from rest\_framework.response import Response

import os

from django.http import HttpResponse, Http404

from users.models import UserAppCreatModel

@api\_view(['GET', 'PUT', 'DELETE','POST'])

def resturl(request,id):

role = request.session['role']

print('ROle is ',role)

if request.method == 'GET':

if role=='user':

dict = {}

data = UserFileModel.objects.get(id=id)

filepath = data.userfile

file = str(filepath).split("/")

rd = open(os.path.join(settings.MEDIA\_ROOT+'/media/', file[1]),'r',encoding='UTF-8',errors='ignore')

filedata = rd.read()

dict.update({'id':id,'filename':file[1],'seckey':data.secretkey,'fdata':filedata})

return render(request,'users/editfilesdata.html',dict)

elif role=='admin':

print('Admin resturl works fine')

dict = {}

data = UserFileModel.objects.get(id=id)

filepath = data.userfile

file = str(filepath).split("/")

rd = open(os.path.join(settings.MEDIA\_ROOT + '/media/', file[1]), 'r', encoding='UTF-8', errors='ignore')

filedata = rd.read()

dict.update({'id': id, 'filename': file[1], 'seckey': data.secretkey, 'fdata': filedata})

return render(request, 'admin/admineditfilesdata.html', dict)

elif role=='cloud':

return Response(status=status.HTTP\_405\_METHOD\_NOT\_ALLOWED)

else:

print("Invalid URL")

elif request.method =='POST':

fileid = request.POST.get('fileid')

filename = request.POST.get('filename')

filedata = request.POST.get('filedata')

with open(settings.MEDIA\_ROOT+'/media' +'/'+filename, 'w+', encoding='UTF-8') as f:

f.write(filedata)

return Response(status=status.HTTP\_200\_OK)

print('POST Request Executed')

print('User ID ',role,'File ID ',id)

return HttpResponse('Am work fine')

def downloadfile(request,id):

data = UserFileModel.objects.get(id=id)

filepath = data.userfile

# x1 = os.path.join(settings.MEDIA\_ROOT+"//"+filepath)

# print('X1 path = ',filepath)

fppath = str(filepath).split("/")

file\_path = os.path.join(settings.MEDIA\_ROOT+'/media/', fppath[1])

if os.path.exists(file\_path):

with open(file\_path, 'rb') as fh:

response = HttpResponse(fh.read(), content\_type="application/vnd.ms-excel")

response['Content-Disposition'] = 'inline; filename=' + os.path.basename(file\_path)

return response

raise Http404

@api\_view(('GET',))

def deletefile(request,id):

role = request.session['role']

if role == 'user':

data = UserFileModel.objects.get(id=id)

data.delete()

##filepath = data.userfile

#fpath = filepath #settings.MEDIA\_ROOT+'/'+filepath

#print('Removing FIle path is ',fpath)

#os.remove(fpath)

return Response(status=status.HTTP\_200\_OK)

elif role =='admin':

return Response(status = status.HTTP\_405\_METHOD\_NOT\_ALLOWED)

elif role =='cloud':

data = UserFileModel.objects.get(id=id)

data.delete()

return Response(status = status.HTTP\_200\_OK)

@api\_view(('GET',))

def uploadfile(request):

role = request.session['role']

if role =='user':

usremail = request.session['email']

dict = UserAppCreatModel.objects.filter(email=usremail)

return render(request,'users/uploadfile.html',{'objects':dict})

elif role =='admin':

return Response(status = status.HTTP\_405\_METHOD\_NOT\_ALLOWED)

elif role =='cloud':

return Response(status = status.HTTP\_405\_METHOD\_NOT\_ALLOWED)

Cloud Side models.py

from django.shortcuts import render,HttpResponse

from django.contrib import messages

from users.models import UserAppCreatModel

import string

import random

# Create your views here.

def cloudlogincheck(request):

if request.method == "POST":

usid = request.POST.get('name')

pswd = request.POST.get('password')

print("User ID is = ", usid)

if usid == 'cloud' and pswd == 'cloud':

request.session['role'] = 'cloud'

return render(request, 'clouds/cloudhome.html')

else:

messages.success(request, 'Invalid Login Details')

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

def activateuserapp(request):

dict = UserAppCreatModel.objects.all()

return render(request,'clouds/userappactivation.html',{'objects':dict})

def clouduserappactivations(request,appname):

accessKey = genAccessToken(10)

secretKey = genSecretKey(32)

print('App Name = ', appname,' Access Key ',accessKey,' Secret Key ',secretKey)

UserAppCreatModel.objects.filter(appname=appname).update(accesskey=accessKey,secretkey=secretKey)

dict = UserAppCreatModel.objects.all()

return render(request, 'clouds/userappactivation.html', {'objects': dict})

def genAccessToken(stringLength=10):

letters = string.ascii\_lowercase

return ''.join(random.choice(letters) for i in range(stringLength))

def genSecretKey(stringLength=32):

"""Generate a random string of letters and digits """

lettersAndDigits = string.ascii\_letters + string.digits

return ''.join(random.choice(lettersAndDigits) for i in range(stringLength))

User side Views.py

from django.db import models

from django.contrib.auth.models import User

import os

INDCHOICES = (

('FINANCE', 'FINANCE'),

('HEALTHCARE', 'HEALTHCARE'),

('INSURANCE', 'INSURANCE'),

('LEGAL', 'LEGAL'),

('MANUFACTURING', 'MANUFACTURING'),

('PUBLISHING', 'PUBLISHING'),

('REAL ESTATE', 'REAL ESTATE'),

('SOFTWARE', 'SOFTWARE'),

)

class Account(models.Model):

name = models.CharField("Name of Account", "name", max\_length=64)

email = models.EmailField(blank = True, null = True)

phone = models.CharField(max\_length=20, blank = True, null = True)

industry = models.CharField("Industry Type", max\_length=255, choices=INDCHOICES, blank=True, null=True)

website = models.URLField("Website", blank=True, null=True)

description = models.TextField(blank=True, null=True)

createdBy = models.ForeignKey(User, related\_name='account\_created\_by', on\_delete=models.CASCADE)

createdAt = models.DateTimeField("Created At", auto\_now\_add=True)

isActive = models.BooleanField(default=False)

def \_\_str\_\_(self):

return self.name

class ContactSource(models.Model):

status = models.CharField("Contact Source", max\_length=20)

def \_\_str\_\_(self):

return self.status

class ContactStatus(models.Model):

status = models.CharField("Contact Status", max\_length=20)

def \_\_str\_\_(self):

return self.status

class Contact(models.Model):

first\_name = models.CharField("First name", max\_length=255, blank = True, null = True)

last\_name = models.CharField("Last name", max\_length=255, blank = True, null = True)

account = models.ForeignKey(Account, related\_name='lead\_account\_contacts', on\_delete=models.CASCADE, blank=True, null=True)

email = models.EmailField()

phone = models.CharField(max\_length=20, blank = True, null = True)

address = models.TextField(blank=True, null=True)

description = models.TextField(blank=True, null=True)

createdBy = models.ForeignKey(User, related\_name='contact\_created\_by', on\_delete=models.CASCADE)

createdAt = models.DateTimeField("Created At", auto\_now\_add=True)

isActive = models.BooleanField(default=False)

def \_\_str\_\_(self):

return self.first\_name

class ActivityStatus(models.Model):

status = models.CharField("Activity Status", max\_length=20)

def \_\_str\_\_(self):

return self.status

class Activity(models.Model):

description = models.TextField(blank=True, null=True)

createdAt = models.DateTimeField("Created At", auto\_now\_add=True)

contact = models.ForeignKey(Contact, on\_delete=models.CASCADE, blank=True, null=True)

def \_\_str\_\_(self):

return self.description

class CloudUsersModel(models.Model):

id = models.AutoField(primary\_key=True)

name = models.CharField(max\_length=200)

email = models.CharField(max\_length=100,unique=True)

password = models.CharField(max\_length=100)

mobile = models.CharField(max\_length=100)

address = models.TextField(max\_length=100)

city = models.CharField(max\_length=100)

state = models.CharField(max\_length=100)

status = models.CharField(max\_length=100,default='waiting')

def \_\_str\_\_(self):

return self.email

class Meta:

db\_table = "registrations"

class UserAppCreatModel(models.Model):

id = models.AutoField(primary\_key=True)

name = models.CharField(max\_length=200)

email = models.CharField(max\_length=200)

appname = models.CharField(max\_length=200,unique=True)

accesskey = models.CharField(max\_length=200,default='waiting')

secretkey = models.CharField(max\_length=200,default='waiting')

def \_\_str\_\_(self):

return self.appname

class Meta:

db\_table = "userapps"

class UserFileModel(models.Model):

id = models.AutoField(primary\_key=True)

name = models.CharField(max\_length=200)

email = models.CharField(max\_length=200)

appname = models.CharField(max\_length=200)

accesskey = models.CharField(max\_length=200)

secretkey = models.CharField(max\_length=200)

filename = models.CharField(max\_length=200)

userfile = models.FileField(upload\_to='media/')

def \_\_str\_\_(self):

return os.path.basename(self.userfile.name)

class Meta:

db\_table = "userfiles"

def delete(self, \*args, \*\*kwargs):

self.userfile.delete()

super().delete(\*args, \*\*kwargs)

forms.py

from django import forms

from .models import CloudUsersModel,UserFileModel

class CloudUserFrom(forms.ModelForm):

name = forms.CharField(widget=forms.TextInput(attrs={'size':50,'class': 'special'}), required=True,max\_length=100 )

email = forms.CharField(widget=forms.TextInput(attrs={'size':50}), required=True, max\_length=100)

password = forms.CharField(widget=forms.PasswordInput(attrs={'size':50}), required=True,max\_length=100)

mobile = forms.CharField(widget=forms.TextInput(attrs={'size':50}), required=True,max\_length=100)

address = forms.CharField(widget=forms.Textarea(attrs={'rows': 3, 'cols': 52}), required=True,max\_length=250)

city = forms.CharField(widget=forms.TextInput(attrs={'size':50}), required=True,max\_length=100)

state = forms.CharField(widget=forms.TextInput(attrs={'size':50}), required=True,max\_length=100)

status = forms.CharField(widget=forms.HiddenInput(), initial='waiting', max\_length=100)

class Meta():

model = CloudUsersModel

fields=['name','email','password','mobile','address','city','state','status']

class UserFileForm(forms.ModelForm):

name = forms.CharField(max\_length=100 )

#email = forms.CharField(max\_length=200)

#appname = forms.CharField(max\_length=200)

#accesskey = forms.CharField(max\_length=200)

#secretkey = forms.CharField(max\_length=200)

#filename = forms.CharField(max\_length=200)

#userfile = forms.FileField()

class Meta():

model = UserFileModel

fields = '\_\_all\_\_'

admin.py

from django.contrib import admin

from users.models import Contact,Account,ContactSource,ContactStatus,ActivityStatus,Activity,CloudUsersModel,UserFileModel

# Register your models here.

admin.site.register(Contact)

admin.site.register(Account)

admin.site.register(ContactSource)

admin.site.register(ContactStatus)

admin.site.register(ActivityStatus)

admin.site.register(Activity)

admin.site.register(CloudUsersModel)

admin.site.register(UserFileModel)

user side views.py

from django.shortcuts import render,HttpResponseRedirect,HttpResponse

from rest\_framework import generics

import json

from .models import Account, Activity, ActivityStatus, Contact, ContactSource, ContactStatus

from .serializers import AccountSerializer, ActivitySerializer, ActivityStatusSerializer, ContactSerializer, ContactSourceSerializer, ContactStatusSerializer,UserFileModelSerializer

from rest\_framework.decorators import api\_view

# Create your views here.

from rest\_framework import generics

from django.http import JsonResponse

from .models import Account, Activity, ActivityStatus, Contact, ContactSource, ContactStatus

from .serializers import AccountSerializer, ActivitySerializer, ActivityStatusSerializer, ContactSerializer, ContactSourceSerializer, ContactStatusSerializer

from rest\_framework.renderers import TemplateHTMLRenderer

from rest\_framework.response import Response

from rest\_framework.views import APIView

from rest\_framework import serializers

from .forms import CloudUserFrom,UserFileForm

from .models import CloudUsersModel

from django.contrib import messages

from rest\_framework import status

from .models import CloudUsersModel,UserAppCreatModel,UserFileModel

#@api\_view(['GET', 'POST'])

class AccountAPIView(generics.ListCreateAPIView):

queryset = Account.objects.all()

serializer\_class = AccountSerializer

renderer\_classes = [TemplateHTMLRenderer]

template\_name = 'profile\_list.html'

def get(self, request):

queryset = Account.objects.all()

return Response({'profiles': queryset})

def post(self, request, pk):

print('AM going to Execute atleast once in my life')

profile = get\_object\_or\_404(Account, pk=pk)

serializer = AccountSerializer(profile, data=request.data)

if not serializer.is\_valid():

return Response({'serializer': serializer, 'profile': Account})

serializer.save()

return redirect('account-list')

class ActivityAPIView(generics.ListCreateAPIView):

queryset = Activity.objects.all()

serializer\_class = ActivitySerializer

class ActivityStatusAPIView(generics.ListCreateAPIView):

queryset = ActivityStatus.objects.all()

serializer\_class = ActivitySerializer

class ContactAPIView(generics.ListCreateAPIView):

queryset = Contact.objects.all()

serializer\_class = ContactSerializer

class ContactStatusAPIView(generics.ListCreateAPIView):

queryset = ContactStatus.objects.all()

serializer\_class = ContactSerializer

class ContactSourceAPIView(generics.ListCreateAPIView):

queryset = ContactSource.objects.all()

serializer\_class = ContactSourceSerializer

def index(request):

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

def userlogin(request):

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

def adminlogin(request):

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

def cloudlogin(request):

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

def userregister(request):

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

@api\_view(['GET', 'POST'])

def storeregistration(request):

if request.method == 'POST':

form = CloudUserFrom(request.POST)

if form.is\_valid():

try:

rslt = form.save()

print("Form Result Status ", rslt)

messages.success(request, 'You have been successfully registered')

except:

messages.success(request, 'Email Already Registerd')

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

else:

print("Invalid form")

else:

form = CloudUserFrom()

return render(request, 'userregister.html', {'form': form})

def userlogincheck(request):

if request.method == "POST":

email = request.POST.get('cf-email')

pswd = request.POST.get('cf-password')

print("Email = ", email)

try:

check = CloudUsersModel.objects.get(email=email, password=pswd)

request.session['id'] = check.id

request.session['loggeduser'] = check.name

request.session['email'] = check.email

request.session['role']='user'

status = check.status

if status == "activated":

print("User id At", check.id, status)

return render(request, 'users/userpage.html', {})

else:

messages.success(request, 'Your Account Not at activated')

return render(request, 'userlogin.html')

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

except:

pass

messages.success(request, 'Invalid Email id and password')

return render(request, 'userlogin.html')

@api\_view(['GET', 'PUT', 'DELETE','POST'])

def snippet\_detail(request):

role = request.session['role']

try:

snippet = UserFileModel.objects.all()

#print('Type is ',snippet.id)

except CloudUsersModel.DoesNotExist:

return Response(status=status.HTTP\_404\_NOT\_FOUND)

if request.method == 'GET':

if role == 'user':

print('Get Method Works Fine')

usremail = request.session['email']

queryset = UserFileModel.objects.filter(email=usremail)

serializer\_class = UserFileModelSerializer

print('Return Type is ',serializer\_class)

return render(request,'users/uploadedfiles.html',{'objects':queryset})

elif role=='admin':

queryset = UserFileModel.objects.all()

serializer\_class = UserFileModelSerializer

return render(request, 'admin/adminuploadedfiles.html', {'objects': queryset})

elif role =='cloud':

queryset = UserFileModel.objects.all()

serializer\_class = UserFileModelSerializer

return render(request, 'clouds/clouduploadedfiles.html', {'objects': queryset})

elif request.method == 'PUT':

print('PUT Method Works Fine')

serializer = UserFileModelSerializer(snippet, data=request.data)

if serializer.is\_valid():

serializer.save()

return Response(serializer.data)

return Response(serializer.errors, status=status.HTTP\_400\_BAD\_REQUEST)

elif request.method == 'DELETE':

print('DELETE Method Works Fine')

#snippet.delete()

return Response(status=status.HTTP\_204\_NO\_CONTENT)

elif request.method=="POST":

print('POST Method Works Fine')

form = UserFileForm(request.POST, request.FILES)

if form.is\_valid():

form.save()

else:

print('Invalid Form')

return Response(status=status.HTTP\_201\_CREATED)

def logout(request):

request.session.modified = True

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

def usercreateapp(request):

usremail = request.session['email']

dict = UserAppCreatModel.objects.filter(email=usremail)

return render(request,'users/userappcreations.html',{'objects':dict})

def appcreaterequest(request):

if request.method=='POST':

usrname = request.POST.get('usrname')

usremail = request.POST.get('usremail')

appname = request.POST.get('appname')

accesskey = request.POST.get('accesskey')

secretkey = request.POST.get('secretkey')

try:

UserAppCreatModel.objects.create(name=usrname,email=usremail,appname=appname)

messages.success(request, 'Your App creation Request is Under Process')

except:

messages.success(request, 'App Name Already exist')

pass

print(usrname,usremail,appname,accesskey,secretkey)

dict = UserAppCreatModel.objects.filter(email=usremail)

return render(request,'users/userappcreations.html',{'objects':dict})

def useruploadfile(request,appname):

check = UserAppCreatModel.objects.get(appname=appname)

acckey = check.accesskey

secretkey = check.secretkey

dict = {'appname':appname,'acckey':acckey,'secretkey':secretkey}

return render(request,'users/uploaddatatocloud.html',dict)

admin.py

from django.shortcuts import render,HttpResponse

from users.models import CloudUsersModel

from django.contrib import messages

# Create your views here.

def adminlogincheck(request):

if request.method == "POST":

usid = request.POST.get('name')

pswd = request.POST.get('password')

print("User ID is = ", usid)

if usid == 'admin' and pswd == 'admin':

request.session['role'] = 'admin'

return render(request, 'admin/adminhome.html')

else:

messages.success(request, 'Invalid Login Details')

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

def adminactivateusers(request):

dict = CloudUsersModel.objects.all()

return render(request,'admin/activateusers.html',{'objects':dict})

def activatewaitedusers(request,id):

if request.method == 'GET':

#uid = request.GET.get('uid')

status = 'activated'

print("PID = ", id,status)

CloudUsersModel.objects.filter(id=id).update(status=status)

dict = CloudUsersModel.objects.all()

return render(request, 'admin/activateusers.html', {'objects': dict})

clouduploadfiles.html

{% extends 'clouds/cloudbase.html'%}

{% load static %}

{% block contents%}

<div id="qbootstrap-testimonial" class="qbootstrap-bg-section">

<div class="container">

<div class="row animate-box">

<span><h1>Cloud Server View Uploaded Files of Users</h1></span>

<p>

<table border="2px">

<tr style="color: darkblue">

<th>S.No</th>

<th>Name</th>

<th>Email</th>

<th>App Name</th>

<th>File Name</th>

<th>Server path</th>

<th>Edit</th>

<th>Delete</th>

<th>Download</th>

<th>Upload New</th>

</tr>

{% for i in objects %}

<tr style="color: RED">

<td>{{forloop.counter}}</td>

<td>{{i.name}}</td>

<td>{{i.email}}</td>

<td>{{i.appname}}</td>

<td>{{i.filename}}</td>

<td>{{i.userfile}}</td>

<td><a class="btn btn-primary btn-md" href="{%url 'resturl' i.id%}">Edit</a> </td>

<td><a class="btn btn-danger btn-md" href="{%url 'deletefile' i.id%}">Delete</a></td>

<td><a class="btn btn-success btn-md" href="{% url 'downloadfile' i.id%}">Download</a> </td>

<td><a class="btn btn-warning btn-md" href="{% url 'uploadfile'%}">Upload</a></td>

</tr>

{% endfor %}

</table>

</p>

</div>

</div>

</div>

{% endblock %}

Cloudbase.html

{% extends 'clouds/cloudbase.html' %}

{% load static %}

{% block contents %}

<div id="qbootstrap-counter" class="qbootstrap-counters" style="background-image: url({% static 'images/img\_bg\_2.jpg'%});" data-stellar-background-ratio="0.5">

<div class="overlay"></div>

<div class="container">

<div class="row">

<div class="col-md-10 col-md-offset-1">

<div class="row">

<div class="col-md-3 col-sm-6 text-center animate-box">

<span class="icon"><i class="icon-group-outline"></i></span>

<span class="qbootstrap-counter js-counter" data-from="0" data-to="65535" data-speed="5000" data-refresh-interval="50"></span>

<span class="qbootstrap-counter-label">Satisfied Customer</span>

</div>

<div class="col-md-3 col-sm-6 text-center animate-box">

<span class="icon"><i class="icon-home-outline"></i></span>

<span class="qbootstrap-counter js-counter" data-from="0" data-to="378" data-speed="5000" data-refresh-interval="50"></span>

<span class="qbootstrap-counter-label">Cloud Hosts</span>

</div>

<div class="col-md-3 col-sm-6 text-center animate-box">

<span class="icon"><i class="icon-user-add-outline"></i></span>

<span class="qbootstrap-counter js-counter" data-from="0" data-to="400" data-speed="5000" data-refresh-interval="50"></span>

<span class="qbootstrap-counter-label">Qualified Professionals</span>

</div>

<div class="col-md-3 col-sm-6 text-center animate-box">

<span class="icon"><i class="icon-point-of-interest-outline"></i></span>

<span class="qbootstrap-counter js-counter" data-from="0" data-to="30" data-speed="5000" data-refresh-interval="50"></span>

<span class="qbootstrap-counter-label">Nodes</span>

</div>

</div>

</div>

</div>

</div>

</div>

{% endblock %}

Useruploadfile.html

{% extends 'users/userbase.html'%}

{% load static %}

{% block contents%}

<div id="qbootstrap-testimonial" class="qbootstrap-bg-section">

<div class="container">

<div class="row animate-box">

<p>

<h1>Created Apps and upload file</h1>

<table border="2px">

<tr style="color: darkblue">

<th>S.No</th>

<th>Name</th>

<th>Email</th>

<th>App Name</th>

<th>Access Key</th>

<th>Token Key</th>

<th>Upload Data</th>

</tr>

{% for i in objects %}

<tr style="color: RED">

<td>{{forloop.counter}}</td>

<td>{{i.name}}</td>

<td>{{i.email}}</td>

<td>{{i.appname}}</td>

<td>{{i.accesskey}}</td>

<td>{{i.secretkey}}</td>

{% if i.secretkey != 'waiting' %}

<td><a class="btn-link" href="{% url 'useruploadfile' i.appname %}" style="color:GREEN">Upload Files</a></td>

{% else %}

<td style="color:Yellow"> Key Not Generated</td>

{% endif %}

</tr>

{% endfor %}

</table>

</p>

</div>

</div>

</div>

{% endblock %}

Cloudlogin.html

{% extends 'base.html' %}

{% load static %}

{% block contents %}

<aside id="qbootstrap-hero">

<div class="flexslider">

<ul class="slides">

<li style="background-image: url({%static 'images/cldadmin.webp'%});">

<div class="overlay"></div>

<div class="container">

<div class="row">

<div class="col-md-8 col-md-offset-2 text-center slider-text">

<div class="slider-text-inner">

<h2>Cloud Server Login here</h2>

<form action="{%url 'cloudlogincheck'%}" method="POST" class="contact-form">

{% csrf\_token %}

<div class="form-group">

<label for="name" class="sr-only">Cloud Name</label>

<input type="text" name="name" style="background-color : #d1d1d1;" class="form-control" id="name" placeholder="Enter Login Name">

</div>

<div class="form-group">

<label for="email" class="sr-only">Email</label>

<input type="password" name="password" style="background-color : #d1d1d1;" class="form-control" id="email" placeholder="Enter Password ">

</div>

<div class="form-group">

<input type="submit" id="btn-submit" class="btn btn-primary btn-send-message btn-md" value="Login">

</div>

</form>

</div>

</div>

</div>

</div>

</li>

</ul>

</div>

</aside>

This is a Index page

{% endblock %}