**SOURCE CODE**

url.py

fromdjango.contrib import admin

fromdjango.urls import path

fromdjango.conf import settings

fromdjango.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

fromadmins.views import adminlogincheck,adminactivateusers,activatewaitedusers

fromclouds.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'),

]

ifsettings.DEBUG:

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

Cloud Side views.py

fromdjango.shortcuts import render,HttpResponse

fromrest\_framework.views import APIView

fromrest\_framework.decorators import api\_view

fromrest\_framework import generics

fromusers.models import UserFileModel

importos

fromdjango.conf import settings

fromdjango.contrib import messages

fromrest\_framework import status

fromrest\_framework.response import Response

importos

fromdjango.http import HttpResponse, Http404

fromusers.models import UserAppCreatModel

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

defresturl(request,id):

role = request.session['role']

print('ROle is ',role)

ifrequest.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")

elifrequest.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)

returnHttpResponse('Am work fine')

defdownloadfile(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])

ifos.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',))

defdeletefile(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

.

.

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

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

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

returnself.description

classCloudUsersModel(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):

returnself.email

class Meta:

db\_table = "registrations"

classUserAppCreatModel(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):

returnself.appname

class Meta:

db\_table = "userapps"

classUserFileModel(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)

.

admin.py

fromdjango.contrib import admin

.

.

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

return render(request, 'userlogin.html')

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

defsnippet\_detail(request):

role = request.session['role']

try:

snippet = UserFileModel.objects.all()

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

exceptCloudUsersModel.DoesNotExist:

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

ifrequest.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})

elifrequest.method == 'PUT':

print('PUT Method Works Fine')

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

ifserializer.is\_valid():

serializer.save()

return Response(serializer.data)

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

.

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

<p>

<table border="2px">

<tr style="color: darkblue">. 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="btnbtn-primary btn-send-message btn-md" value="Login">

</div>

</form>

</div>

</div>

</div>

</div>

</li>

</ul>

</div>

</aside>

This is a Index page

{% endblock %}