**SOURCE CODE**

APPLICATION

**Main app:**

**Views.py:**

from django.shortcuts import render,redirect

# Create your views here.

def main\_index(request):

return render(request,'main/index.html')

def about(request):

return render(request,'main/about.html')

**app.py:**

from django.apps import AppConfig

class MainappConfig(AppConfig):

default\_auto\_field = 'django.db.models.BigAutoField'

name = 'mainapp'

**Data user:**

**Views.py:**

from asyncio.windows\_events import NULL

from distutils.command.upload import upload

from xml.dom.minidom import Document

from pyexpat.errors import messages

from django.shortcuts import get\_object\_or\_404, render,redirect

import random

import requests

from django.contrib import messages

from userapp.models import UserModel,DownloadRequestModel,DownloadsModel

from ownerapp.models import UploadDocumentsModel,OwnerModel

from django.db.models import Q,F

from cryptography.fernet import Fernet

import base64

import csv

from FileSecure.settings import DEFAULT\_FROM\_EMAIL

from django.core.mail import EmailMultiAlternatives

import urllib.request

from django.core.files.storage import FileSystemStorage

from FileSecure import check\_internet

# Create your views here.

def user\_register(request):

if request.method =='POST' and request.FILES["profile"]:

print("post method")

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

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

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

profile = request.FILES['profile']

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

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

if UserModel.objects.filter(email=email,status="verified").exists():

print("email already exists")

messages.error(request,"already taken")

return redirect('user-register')

elif UserModel.objects.filter(email=email,status="pending").exists():

messages.warning(request,"Already Registered, Just verify your account!")

if pwd == cpwd:

otp=random.randint(2222,4444)

UserModel.objects.filter(email=email,status='pending').update(otp=otp)

if check\_internet.connect():

html\_content = "<br/> WellCome To Data User, <br/> <p> This Message Sent From FileSecure You Have Recieved a OTP <strong> " +str(otp)+ " </strong> on FileSecure Cloud Based File Services. </strong> <b> You Can Use online services 24/7 <strong> Thank You For Your Registration.</p>"

from\_mail = DEFAULT\_FROM\_EMAIL

to\_mail = [email]

# if send\_mail(subject,message,from\_mail,to\_mail):

msg = EmailMultiAlternatives("Account Registration Status", html\_content, from\_mail, to\_mail)

msg.attach\_alternative(html\_content, "text/html")

print(email)

if msg.send()

print("Sent")

print('your success')

# url = "https://www.fast2sms.com/dev/bulkV2"

# # create a dictionary

# my\_data = {

# # Your default Sender ID

**Models.py:**

from django.db import models

from ownerapp.models import OwnerModel, UploadDocumentsModel

# Create your models here.

class UserModel(models.Model):

user\_id = models.AutoField(primary\_key=True)

name = models.TextField(max\_length=200,default=True)

mobile = models.BigIntegerField(default=True)

email = models.EmailField(default=True)

profile = models.ImageField(upload\_to='user/',null=True)

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

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

reg\_date = models.DateField(auto\_now\_add=True)

otp = models.TextField(null=True)

status = models.CharField(default="pending",max\_length=20)

class Meta:

db\_table = 'user\_details'

class DownloadRequestModel(models.Model):

request\_id=models.AutoField(primary\_key=True)

owner\_id=models.IntegerField(null=True)

user\_id=models.IntegerField(null=True)

doc\_id=models.IntegerField(null=True)

document=models.FileField(upload\_to='files/')

doc\_name=models.CharField(max\_length=200)

description=models.CharField(max\_length=250, null=True)

doc\_size=models.BigIntegerField(null=True)

doc\_type=models.CharField(max\_length=300)

# search\_rank = models.CharField(default="0",null=True, max\_length=200)

# download\_rank = models.CharField(default="0",null=True, max\_length=250)

# otp=models.IntegerField(null=True)

file\_enc\_key = models.TextField(null=True)

secure\_key = models.CharField(max\_length=100,null=True)

request\_status=models.CharField(max\_length=50,default='pending',null="True")

download\_status = models.CharField(max\_length=200,default='pending',null=True)

requested\_date=models.DateField(auto\_now\_add=True, null=True)

class Meta:

db\_table = "user\_download\_requests"

class DownloadsModel(models.Model):

download\_id = models.AutoField(primary\_key=True)

user\_id = models.IntegerField(null=True)

doc\_id = models.IntegerField(null=True)

req\_id = models.IntegerField(null=True)

status = models.CharField(default="pending",null=True,max\_length=200)

# download\_rank = models.IntegerField(null=True)

downloaded\_date=models.DateField(auto\_now\_add=True, null=True)

class Meta:

db\_table = "document\_downloads"

**app.py**

from django.apps import AppConfig

class UserappConfig(AppConfig):

default\_auto\_field = 'django.db.models.BigAutoField'

name = 'userapp'

**Migration.py**

from django.db import migrations

class Migration(migrations.Migration):

dependencies = [

('userapp', '0002\_downloadsmodel\_req\_id'),]

operations = [

migrations.RemoveField(

model\_name='downloadsmodel',

name='download\_rank',

),

]

from django.db import migrations, models

class Migration(migrations.Migration):

dependencies = [

('userapp', '0001\_initial'),

]

operations = [

migrations.AddField(

model\_name='downloadsmodel',

name='req\_id',

field=models.IntegerField(null=True),

),

]

**\_\_init\_\_.py:**

from django.db import migrations, models

class Migration(migrations.Migration):

initial = True

dependencies = [

]

operations = [

migrations.CreateModel(

name='DownloadRequestModel',

fields=[

('request\_id', models.AutoField(primary\_key=True, serialize=False)),

('owner\_id', models.IntegerField(null=True)),

('user\_id', models.IntegerField(null=True)),

('doc\_id', models.IntegerField(null=True)),

('document', models.FileField(upload\_to='files/')),

('doc\_name', models.CharField(max\_length=200)),

('description', models.CharField(max\_length=250, null=True)),

('doc\_size', models.BigIntegerField(null=True)),

('doc\_type', models.CharField(max\_length=300)),

('file\_enc\_key', models.TextField(null=True)),

('secure\_key', models.CharField(max\_length=100, null=True)),

('request\_status', models.CharField(default='pending', max\_length=50, null='True')),

('download\_status', models.CharField(default='pending', max\_length=200, null=True)),

('requested\_date', models.DateField(auto\_now\_add=True, null=True)),

],

options={

'db\_table': 'user\_download\_requests',

},

),

migrations.CreateModel(

name='DownloadsModel',

fields=[

('download\_id', models.AutoField(primary\_key=True, serialize=False)),

('user\_id', models.IntegerField(null=True)),

('doc\_id', models.IntegerField(null=True)),

('status', models.CharField(default='pending', max\_length=200, null=True)),

('download\_rank', models.IntegerField(null=True)),

('downloaded\_date', models.DateField(auto\_now\_add=True, null=True)),

],

options={

'db\_table': 'document\_downloads',

},

),

migrations.CreateModel(

name='UserModel',

fields=[

('user\_id', models.AutoField(primary\_key=True, serialize=False)),

('name', models.TextField(default=True, max\_length=200)),

('mobile', models.BigIntegerField(default=True)),

('email', models.EmailField(default=True, max\_length=254)),

('profile', models.ImageField(null=True, upload\_to='user/')),

('pwd', models.CharField(max\_length=20, null=True)),

('cpwd', models.CharField(max\_length=20, null=True)),

('reg\_date', models.DateField(auto\_now\_add=True)),

('otp', models.TextField(null=True)),

('status', models.CharField(default='pending', max\_length=20)),

],

options={

'db\_table': 'user\_details',

},

),

]

Data Owner app:

Views.py:

import imp

from operator import indexOf

import os

from tabnanny import filename\_only

from xml.dom.minidom import Document

from django.conf import settings

from django.shortcuts import render,redirect,get\_object\_or\_404

from ownerapp.models import \*

from userapp.models import \*

from django.contrib import messages

import random

import requests

import string

import random

from userapp import practice as p

from cryptography.fernet import Fernet

import pathlib

from django.db.models import Q

from FileSecure.settings import DEFAULT\_FROM\_EMAIL

from django.core.mail import EmailMultiAlternatives

from FileSecure import check\_internet

from FileSecure.check\_internet import \*

from django.core.files.storage import FileSystemStorage

# Create your views here.

def owner\_register(request):

if request.method =='POST' and request.FILES["profile"]:

print("post method")

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

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

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

profile = request.FILES['profile']

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

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

if OwnerModel.objects.filter(email=email).filter(verification="verified").exists():

print("email already exists")

messages.error(request,"already taken")

return redirect('owner-register')

elif OwnerModel.objects.filter(email=email).filter(verification = "pending").exists():

# messages.error(request,"Already Registered, Just verify your account!")

if pwd == cpwd:

otp=random.randint(2222,4444)

OwnerModel.objects.filter(email=email,status='pending').update(otp=otp)

if connect():

html\_content = "<br/> WellCome To Data Owner, <br/> <p> This Message Sent From FileSecure You Have Recieved a OTP <strong> " +str(otp)+ " </strong> on FileSecure Cloud Based File Services. </strong> <b> You Can Use online services 24/7 <strong> Thank You For Your Registration.</p>"

from\_mail = DEFAULT\_FROM\_EMAIL

to\_mail = [email]

# if send\_mail(subject,message,from\_mail,to\_mail):

msg = EmailMultiAlternatives("Account Registration Status", html\_content, from\_mail, to\_mail)

msg.attach\_alternative(html\_content, "text/html")

print(email,"email ")

return redirect('owner-otp-verify')

else:

messages.warning(request,"password does't match")

return redirect('owner-register')

else:

otp = random.randint(1111,9999)

print(otp)

if pwd == cpwd:

owner\_register = OwnerModel.objects.create(name=name,email=email,mobile=mobile,pwd=pwd,cpwd=cpwd,otp=otp,profile=profile)

owner = OwnerModel.objects.get(email=email)

otp\_data = owner.owner\_id

print('session',otp\_data)

request.session['demo'] = otp\_data

print(otp\_data)

if connect():

html\_content = "<br/> WellCome To Data Owner, <br/> <p> This Message Sent From FileSecure You Have Recieved a OTP <strong> " +str(otp)+ " </strong> on FileSecure Cloud Based File Services. </strong> <b> You Can Use online services 24/7 <strong> Thank You For Your Registration.</p>"

from\_mail = DEFAULT\_FROM\_EMAIL

to\_mail = [owner\_register.email]

# if send\_mail(subject,message,from\_mail,to\_mail):

msg = EmailMultiAlternatives("Account Registered Status", html\_content, from\_mail, to\_mail)

msg.attach\_alternative(html\_content, "text/html")

print(email)

if msg.send():

print("Sent")

print('your success')

return redirect('owner-otp-verify')

else:

messages.warning(request,"password does't match")

return redirect('owner-register')

# messages.info(request,"successfully registered")

return render(request,'owner/owner-register.html')

def owner\_otp\_verify(request):

owner\_id = request.session['demo']

print("owner\_id",owner\_id)

data = OwnerModel.objects.get(owner\_id=owner\_id)

if request.method == 'POST':

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

print(otp)

try:

check = OwnerModel.objects.get(otp=otp)

print(check)

o\_id = request.session["owner\_id"]=check.owner\_id

otp = check.otp

if otp == otp:

print("c")

OwnerModel.objects.filter(owner\_id=o\_id).update(verification="verified")

print("record")

messages.info(request,"otp-verify")

return redirect('owner-login')

else:

print("d")

messages.error(request,"wrong-otp")

return redirect('owner-otp-verify')

except:

pass

return render(request,'owner/owner-otp-verify.html',{'d':data})

def owner\_login(request):

if request.method == 'POST':

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

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

try:

check = OwnerModel.objects.get(email=email,pwd=pwd)

print("check")

id = request.session["owner\_id"]=check.owner\_id

status = check.status

if status == "Accepted" and check.verification == "verified" :

messages.info(request,"log in successfull!")

print("message....")

return redirect('owner-home')

elif status == "Rejected" or status=="pending":

messages.warning(request,"admin not accepted")

else:

messages.error(request,"something went wrong!")

return redirect('owner-login')

except:

print('invalid.......')

messages.error(request,"Invalid Credentials")

return redirect("owner-login")

return render(request,'owner/owner-login.html')

def owner\_home(request):

owner\_id = request.session["owner\_id"]

document = UploadDocumentsModel.objects.filter(owner\_id=owner\_id,status="Accepted")

if request.method == "POST" and 'btn1' in request.POST or 'btn2' in request.POST:

search = request.POST.get("search")

document = UploadDocumentsModel.objects.filter(owner\_id=owner\_id,status="Accepted").filter(Q(doc\_name\_\_icontains=search)|Q(doc\_type\_\_icontains=search)|Q(random\_keys\_\_icontains=search))

return render(request,'owner/owner-home.html',{'docs':document})

def owner\_upload\_docs(request):

owner\_id = request.session['owner\_id']

supported\_files = "html,java,py,txt,css,js"

# print("support",supported\_files)

if request.method =='POST' and request.FILES['file']:

print("post method")

# key = str(Fernet.generate\_key(), 'utf-8')

# crypter = Fernet(key)

# document = request.FILES['file'].encode()

# latt = str(crypter.encrypt(document), 'utf-8')

document=request.FILES['file']

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

doc\_name = document.name

doc\_type = document.content\_type

file\_extension = doc\_name

fs = FileSystemStorage()

name = fs.save('files/' + doc\_name,document)

print("111111")

print(name)

a = file\_extension.index(".")

b = len(file\_extension)

x = file\_extension[a+1:b]

print(x)

if x in supported\_files:

print("File is supporteddd")

doc\_size = document.size

owner = OwnerModel.objects.get(owner\_id = owner\_id)

files = UploadDocumentsModel.objects.create(doc\_name=doc\_name,doc\_size=doc\_size,doc\_type=doc\_type,description=description,document=name,owner=owner)

files.save()

messages.info(request,"document uploaded")

doc\_id = request.session["doc\_id"] = files.doc\_id

print(doc\_id)

return redirect('owner-encrypt-file')

else:

messages.error(request,"The " + x + " file format is not supported by the cloud")

return redirect("upload-docs")

return render(request,'owner/owner-upload-docs.html')

def owner\_encrypt\_file(request):

owner\_id = request.session['owner\_id']

data = ''

document = request.session["doc\_id"]

file = UploadDocumentsModel.objects.get(doc\_id=document)

filename = str(file.doc\_name)

# f = filename.replace(' ','\_')

path = 'media/files/' + filename

if request.method=="GET":

print(path)

try:

f = open(path,'r')

print('f')

data = f.read()

f.close()

print(data)

except Exception as e:

print(e)

elif request.method=="POST" and 'encrypt' in request.POST:

print("keywords")

data1 = request.POST.get("description")

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

print(keywords)

# #File Encryption

# file\_enc\_key = Fernet.generate\_key()

# print(file\_enc\_key,"keykeykey")

# print("aaaa")

# print(file\_enc\_key,"dddd")

# print("aaaa")

# data\_encode = data1.encode()

# fernet = Fernet(file\_enc\_key)

# enc\_data\_1 = fernet.encrypt(data\_encode).decode()

# print(enc\_data\_1)

# #File Encryption

file\_enc\_key = Fernet.generate\_key()

#Read File

print(path)

file = open(path,'rb')

data = file.read()

print(data)

fernet = Fernet(file\_enc\_key)

data = fernet.encrypt(data)

print(data)

UploadDocumentsModel.objects.filter(owner\_id=owner\_id,doc\_name=filename).update(file\_enc\_key=file\_enc\_key.decode(),file\_data=data.decode(),random\_keys=keywords)

with open(path,'wb') as f:

f.write(data)

messages.warning(request,"encrypted file uploaded")

return redirect("owner-home")

return render(request,"owner/owner-encrypt-file.html",{'file': data,'filename': filename})

def owner\_download\_requests(request):

owner\_id = request.session["owner\_id"]

# docs = UploadDocumentsModel.objects.filter(status="Accepted")

user\_data = UserModel.objects.all()

document = DownloadRequestModel.objects.filter(owner\_id=owner\_id).all()

if request.method == "POST" and 'btn1' in request.POST:

search = request.POST.get("search")

document = DownloadRequestModel.objects.filter(Q(doc\_name\_\_icontains=search)|Q(doc\_type\_\_icontains=search))

return render(request,'owner/owner-download-requests.html',{'req':document,'user':user\_data})

# key or captcha generator, we can mention any size in size argument

def generate\_key(size=10, chars=string.ascii\_uppercase + string.ascii\_lowercase + string.digits + '!@#$%^&\*()\_+?~'):

return ''.join(random.choice(chars) for \_ in range(size))

def key\_generator(request,id):

key = generate\_key()

print(key)

accept = get\_object\_or\_404(DownloadRequestModel,request\_id =id)

accept.request\_status = "request\_accepted"

accept.secure\_key=key

accept.save(update\_fields=['request\_status','secure\_key'])

accept.save()

# DownloadRequestModel.objects.filter(request\_id=id).update(secure\_key=key,request\_status="accepted")

return redirect('owner-download-requests')

def reject\_request(request,id):

reject = get\_object\_or\_404(DownloadRequestModel,request\_id =id)

reject.request\_status = "Rejected"

reject.save(update\_fields=['request\_status'])

reject.save()

return redirect('owner-download-requests')

def owner\_topk\_downloads(request):

owner\_id = request.session["owner\_id"]

# owner = UploadDocumentsModel.objects.get(owner\_id = owner\_id)

# doc\_id = owner.doc\_id

Document = UploadDocumentsModel.objects.filter(owner\_id=owner\_id).order\_by("-download\_rank")

# downloads = DownloadsModel.objects.all()

# request\_id = downloads.req\_id

# downloads = DownloadsModel.objects.filter(Q(doc\_name\_\_icontains=search)|Q(doc\_type\_\_icontains=search))

# user = UserModel.objects.all()

# ddd = DownloadRequestModel.objects.filter(owner\_id = owner\_id)

return render(request,'owner/owner-topk-downloads.html',{'downloads':Document,})

def owner\_profile(request):

owner = request.session["owner\_id"]

data = OwnerModel.objects.get(owner\_id=owner)

# owner\_data = OwnerModel.objects.filter(owner\_id=owner).exclude(status="paid").exclude(status='Delivered').count()

obj = get\_object\_or\_404(OwnerModel,owner\_id=owner)

if request.method == "POST" and request.FILES["profile"]:

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

mobile = request.POST.get('phone')

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

# address = request.POST.get('address')

profile = request.FILES['profile']

# pincode = request.POST.get('pincode')

obj.name = name

obj.mobile = mobile

obj.profile = profile

obj.email = email

obj.save(update\_fields=['name','profile','mobile','email'])

obj.save()

messages.info(request,"profile Updated successfully..")

if obj.save():

messages.info(request,"profile Updated successfully..")

else:

messages.error(request,"something went wrong!")

return redirect("owner-profile")

return render(request,'owner/owner-profile.html',{'owner':data})

def owner\_about(request):

return render(request,'owner/owner-about.html')

Models.py:

from pyexpat import model

from django.db import models

# Create your models here.

class OwnerModel(models.Model):

owner\_id = models.AutoField(primary\_key=True)

name = models.TextField(max\_length=200,default=True)

mobile = models.BigIntegerField(default=True)

email = models.EmailField(default=True)

profile = models.ImageField(upload\_to='owner/',null=True)

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

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

reg\_date = models.DateField(auto\_now\_add=True)

otp = models.TextField(null=True)

status = models.CharField(default="pending",max\_length=20)

verification = models.CharField(default="pending",max\_length=20)

class Meta:

db\_table = 'owner\_details'

class UploadDocumentsModel(models.Model):

doc\_id = models.AutoField(primary\_key=True)

owner = models.ForeignKey(OwnerModel,on\_delete=models.CASCADE,null=True)

document = models.ImageField(upload\_to='files/',null=True)

doc\_type = models.TextField(null=True)

doc\_size = models.BigIntegerField(null=True)

description = models.TextField(null=True)

doc\_name = models.TextField(null=True)

enc\_doc\_name = models.TextField(null=True)

random\_keys = models.TextField(null=True)

file\_enc\_key = models.TextField(null=True)

file\_data = models.TextField(null=True)

download\_rank = models.IntegerField(null=True)

upload\_date = models.DateField(auto\_now\_add=True)

status = models.TextField(default="pending")

class Meta:

db\_table='upload\_documents'

**Apps.py :**

from django.apps import AppConfig

class OwnerappConfig(AppConfig):

default\_auto\_field = 'django.db.models.BigAutoField'

name = 'ownerapp'

**Migration.py**

Generated by Django 4.0.4 on 2022-06-25 10:45

from django.db import migrations, models

class Migration(migrations.Migration):

dependencies = [

('ownerapp', '0001\_initial'),

]

operations = [

migrations.AddField(

model\_name='uploaddocumentsmodel',

name='download\_rank',

field=models.IntegerField(null=True),

),

]

**Initial.py:**

from django.db import migrations, models

import django.db.models.deletion

class Migration(migrations.Migration):

initial = True

dependencies = [

]

operations = [

migrations.CreateModel(

name='OwnerModel',

fields=[

('owner\_id', models.AutoField(primary\_key=True, serialize=False)),

('name', models.TextField(default=True, max\_length=200)),

('mobile', models.BigIntegerField(default=True)),

('email', models.EmailField(default=True, max\_length=254)),

('profile', models.ImageField(null=True, upload\_to='owner/')),

('pwd', models.CharField(max\_length=20, null=True)),

('cpwd', models.CharField(max\_length=20, null=True)),

('reg\_date', models.DateField(auto\_now\_add=True)),

('otp', models.TextField(null=True)),

('status', models.CharField(default='pending', max\_length=20)),

('verification', models.CharField(default='pending', max\_length=20)),

],

options={

'db\_table': 'owner\_details',

},

),

migrations.CreateModel(

name='UploadDocumentsModel',

fields=[

('doc\_id', models.AutoField(primary\_key=True, serialize=False)),

('document', models.ImageField(null=True, upload\_to='files/')),

('doc\_type', models.TextField(null=True)),

('doc\_size', models.BigIntegerField(null=True)),

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

('doc\_name', models.TextField(null=True)),

('enc\_doc\_name', models.TextField(null=True)),

('random\_keys', models.TextField(null=True)),

('file\_enc\_key', models.TextField(null=True)),

('file\_data', models.TextField(null=True)),

('upload\_date', models.DateField(auto\_now\_add=True)),

('status', models.TextField(default='pending')),

('owner', models.ForeignKey(null=True, on\_delete=django.db.models.deletion.CASCADE, to='ownerapp.ownermodel')),

],

options={

'db\_table': 'upload\_documents',

},

),

]

**Cloud server app:**

**Views.py:**

from pyexpat.errors import messages

from django.shortcuts import render,redirect,get\_object\_or\_404

from ownerapp.models import OwnerModel,UploadDocumentsModel

from userapp.models import DownloadsModel, UserModel,DownloadRequestModel

import datetime

from django.contrib import messages

# Create your views here.

def cloud\_login(request):

if request.method == "POST":

name= request.POST.get("name")

pwd = request.POST.get("pwd")

if name =='cloud' and pwd =='cloud':

messages.info(request,"login success")

return redirect('cloud-index')

else:

messages.warning(request,"wrong details")

return render(request,'cloud/cloud-login.html')

def cloud\_index(request):

# current time and date

# datetime object

date = datetime.date.today()

print(date)

# formating date using strftime

# print("After formating:", time.strftime("%b %d, %Y"))

users\_count = UserModel.objects.all().count()

owners\_count = OwnerModel.objects.filter(status="accepted").count()

doc\_count = UploadDocumentsModel.objects.filter(status="Accepted").count()

doc\_req\_count = UploadDocumentsModel.objects.filter(status="pending").count()

total\_downloads = DownloadsModel.objects.all().count()

m\_downloads = DownloadsModel.objects.filter(downloaded\_date=date).count()

down\_req\_count = DownloadRequestModel.objects.filter(download\_status='pending').count()

return render(request,'cloud/index.html',{

'u\_count':users\_count,

'o\_count':owners\_count,

'd\_count':doc\_count,

'd\_r\_count':down\_req\_count,

'doc\_req\_count':doc\_req\_count,

'tot\_doc\_count':total\_downloads,

'month\_downloads':m\_downloads,

})

def data\_owner(request):

owner = OwnerModel.objects.all().order\_by('-reg\_date')

return render(request,'cloud/cloud-data-owner.html',{'owner':owner})

def accept\_status(request,id):

accept = get\_object\_or\_404(OwnerModel,owner\_id =id)

accept.status = "Accepted"

accept.save(update\_fields=['status'])

accept.save()

return redirect("data-owner")

def reject\_status(request,id):

accept = get\_object\_or\_404(OwnerModel,owner\_id =id)

accept.status = "Rejected"

accept.save(update\_fields=['status'])

accept.save()

return redirect("data-owner")

def data\_user(request):

user = UserModel.objects.all()

return render(request,'cloud/cloud-data-user.html',{'user':user})

def view\_documents(request):

docs = UploadDocumentsModel.objects.all().order\_by('-upload\_date')

return render(request,'cloud/cloud-view-documents.html',{'docs':docs})

def accept\_document(request,id):

accept = get\_object\_or\_404(UploadDocumentsModel,doc\_id =id)

accept.status = "Accepted"

accept.save(update\_fields=['status'])

accept.save()

return redirect("view-documents")

def reject\_document(request,id):

accept = get\_object\_or\_404(UploadDocumentsModel,doc\_id =id)

accept.status = "Rejected"

accept.save(update\_fields=['status'])

accept.save()

return redirect("view-documents")

def view\_downloads(request):

downloads = DownloadsModel.objects.all()

user = UserModel.objects.all()

owner = UploadDocumentsModel.objects.all()

return render(request,'cloud/cloud-view-downloads.html',{'d':downloads,'owner':owner,'u':user})

**Apps.py:**

from django.apps import AppConfig

class CloudappConfig(AppConfig):

default\_auto\_field = 'django.db.models.BigAutoField'

name = 'cloudapp'

**Emailsettings.py :**

SET\_EMAIL\_USE\_TLS = True

SET\_EMAIL\_HOST='smtp.gmail.com'

SET\_EMAIL\_HOST\_USER='projects@codebook.in'

SET\_EMAIL\_HOST\_PASSWORD='frwqvhawrnsxetyk'

SET\_EMAIL\_PORT=587

SET\_EMAIL\_BACKEND = 'django.core.mail.backends.smtp.EmailBackend'

SET\_DEFAULT\_FROM\_EMAIL = 'projects@codebook.in'

**Settings.py:**

from pathlib import Path

import os

from .emailsettings import SET\_EMAIL\_USE\_TLS, SET\_EMAIL\_HOST, SET\_EMAIL\_HOST\_USER, \

SET\_EMAIL\_HOST\_PASSWORD, SET\_EMAIL\_PORT, SET\_EMAIL\_BACKEND, SET\_DEFAULT\_FROM\_EMAIL

from django.contrib.messages import constants as messages

# Build paths inside the project like this: BASE\_DIR / 'subdir'.

BASE\_DIR = Path(\_\_file\_\_).resolve().parent.parent

# Quick-start development settings - unsuitable for production

# See https://docs.djangoproject.com/en/4.0/howto/deployment/checklist/

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

SECRET\_KEY = 'django-insecure-xhkn\_dmpywn@5iq9r+i=2bp%3=976713lgvo46yvku0r9e35-w'

# 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',

'cloudapp',

'mainapp',

'ownerapp',

'userapp',

]

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 = 'FileSecure.urls'

TEMPLATES = [

{

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

'DIRS': [os.path.join(BASE\_DIR,'assets/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 = 'FileSecure.wsgi.application'

# Database

# https://docs.djangoproject.com/en/4.0/ref/settings/#databases

DATABASES = {

'default': {

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

'NAME':'filesecure',

'USER':'root',

'PASSWORD':'',

'HOST':'localhost',

'PORT':'3306',

'OPTIONS': {

'init\_command': "SET sql\_mode='STRICT\_TRANS\_TABLES'" ,

}

}

}

# Password validation

# https://docs.djangoproject.com/en/4.0/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/4.0/topics/i18n/

LANGUAGE\_CODE = 'en-us'

TIME\_ZONE = 'UTC'

USE\_I18N = True

USE\_TZ = True

# Static files (CSS, JavaScript, Images)

# https://docs.djangoproject.com/en/4.0/howto/static-files/

STATIC\_URL = '/static/'

STATICFILES\_DIRS=[os.path.join(BASE\_DIR,'assets/static'),]

MEDIA\_URL = '/media/'

MEDIA\_ROOT= os.path.join(BASE\_DIR,'media/')

# Default primary key field type

# https://docs.djangoproject.com/en/4.0/ref/settings/#default-auto-field

DEFAULT\_AUTO\_FIELD = 'django.db.models.BigAutoField'

EMAIL\_USE\_TLS = SET\_EMAIL\_USE\_TLS

EMAIL\_HOST = SET\_EMAIL\_HOST

EMAIL\_HOST\_USER = SET\_EMAIL\_HOST\_USER

EMAIL\_HOST\_PASSWORD = SET\_EMAIL\_HOST\_PASSWORD

EMAIL\_PORT = SET\_EMAIL\_PORT

EMAIL\_BACKEND = SET\_EMAIL\_BACKEND

DEFAULT\_FROM\_EMAIL = SET\_DEFAULT\_FROM\_EMAIL

MESSAGE\_TAGS = {

messages.DEBUG : 'alert-secondary',

messages.INFO: 'alert-info',

messages.WARNING: 'alert-warning',

messages.SUCCESS: 'alert-success',

messages.ERROR: 'alert-danger',

}

**Check\_internet.py:**

import urllib.request

def connect(host='http://google.com'):

try:

urllib.request.urlopen(host)

return True

except:

return False

**Urls.py:**

from django.contrib import admin

from django.urls import path

from mainapp import views as mainviews

from userapp import views as userviews

from ownerapp import views as ownerviews

from cloudapp import views as cloudviews

from django.conf.urls.static import static

from django.conf import settings

urlpatterns = [

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

# main page url's

path('',mainviews.main\_index,name="index"),

path('about',mainviews.about,name="about"),

# user page url's

path('user-register',userviews.user\_register,name="user-register"),

path('user-otp-verify',userviews.otp\_verify,name="user-otp-verify"),

path('user-login',userviews.user\_login,name="user-login"),

path('user-home',userviews.user\_home,name="user-home"),

path('download-requests/<int:id>/',userviews.download\_requests,name="download-requests"),

path('user-view-docs',userviews.user\_search\_docs,name="user-view-docs"),

path('user-mydownloads',userviews.user\_mydownloads,name="user-mydownloads"),

path('user-profile',userviews.user\_profile,name="user-profile"),

path('user-response-docs',userviews.user\_response\_docs,name="user-response-docs"),

path('user-download-document/<int:id>/',userviews.download\_doc,name="user-download-document"),

path('user-decrypt-document/<int:id>/',userviews.decrypt\_document,name="user-decrypt-document"),

path('user-aboutus',userviews.user\_about,name="user-aboutus"),

# owner page url's

path('owner-register',ownerviews.owner\_register,name="owner-register"),

path('owner-otp-verify',ownerviews.owner\_otp\_verify,name="owner-otp-verify"),

path('owner-encrypt-file',ownerviews.owner\_encrypt\_file,name="owner-encrypt-file"),

path('owner-login',ownerviews.owner\_login,name="owner-login"),

path('owner-home',ownerviews.owner\_home,name="owner-home"),

path('upload-docs',ownerviews.owner\_upload\_docs,name="upload-docs"),

path('owner-download-requests',ownerviews.owner\_download\_requests,name="owner-download-requests"),

path('key-generator/<int:id>/',ownerviews.key\_generator,name="key-generator"),

path('reject-request/<int:id>/',ownerviews.reject\_request,name="reject-request"),

path('top-kdownloads',ownerviews.owner\_topk\_downloads,name="top-kdownloads"),

path('owner-profile',ownerviews.owner\_profile,name="owner-profile"),

path('owner-aboutus',ownerviews.owner\_about,name="owner-aboutus"),

#cloud page url's

path('clod-login',cloudviews.cloud\_login,name="cloud-login"),

path('cloud-index',cloudviews.cloud\_index,name="cloud-index"),

path('data-owner',cloudviews.data\_owner,name="data-owner"),

path('data-user',cloudviews.data\_user,name="data-user"),

path('view-documents',cloudviews.view\_documents,name="view-documents"),

path('accept-document/<int:id>/',cloudviews.accept\_document,name="accept-document"),

path('reject-document/<int:id>/',cloudviews.reject\_document,name="reject-document"),

path('view-downloads',cloudviews.view\_downloads,name="view-downloads"),

path('accept-owner/<int:id>/',cloudviews.accept\_status,name="accept-owner"),

path('reject-owner/<int:id>/',cloudviews.reject\_status,name="reject-owner"),

]

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