Urls.py

It will show urls of specific page

from django.urls import path

from . import views

urlpatterns = [

path('', views.index , name="PortalHome"),

path('register/' , views.register , name = "register"),

path('registerall/' , views.registerall , name = "registerall") ,

path('check/' , views.check , name = "check"),

path('teacherlogin/' , views.teacherlogin , name = "teacherlogin"),

path('studentlogin/' , views.studentlogin , name = "studentlogin"),

path('paperinfo/' , views.paperinfo , name = "paperinfo"),

path('teacherlogout/' , views.teacherlogout , name = "teacherlogout"),

path('answerinfo/' , views.answerinfo , name = "answerinfo"),

path('answerevaluation/' , views.answereval , name = "answereval"),

#path('design/' , views.design , name= "design")

]

Views.py

#contain diffent function

from django.shortcuts import render

from django.http import HttpResponse,HttpResponseRedirect

from portal.models import Student , Teacher , Paperinfo

from django.urls import reverse

from passlib.hash import pbkdf2\_sha256

from nltk.tokenize import sent\_tokenize, word\_tokenize

from nltk.corpus import stopwords,wordnet

from nltk.stem import WordNetLemmatizer

from itertools import product

import numpy

# Create your views here.

count = 0

def index(request):

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

def register(request):

return render(request , 'portal/register.html')

def registerall(request):

#print(request.POST)

username = request.POST['username']

email = request.POST['email']

password = request.POST['password']

position = request.POST['position']

print(username, email , password , position)

password = pbkdf2\_sha256.encrypt(password , rounds = 12000 , salt\_size=32)

if(position == "1"):

s = Teacher(teacher\_name = username , teacher\_password = password , teacher\_email = "T" +email)

else:

s = Student(student\_name = username , student\_password = password , student\_email = "S" + email)

s.save()

if s.id:

return HttpResponseRedirect(reverse('PortalHome'))

else:

return HttpResponse("error")

#return HttpResponse("username:{0}\ncountry:{1}".format(username,country))

def teacherlogin(request):

return render(request , 'portal/teacher\_login.html')

def studentlogin(request):

return render(request , 'portal/student\_login.html')

def check(request):

username = request.POST['username']

password = request.POST['password']

l1 = Teacher.objects.filter(teacher\_email = 'T' + username)

l2 = Student.objects.filter(student\_email = 'S' + username)

print(l1 , l2)

print(type(l1) , type(l2))

t = False

s = False

for i in l1:

t = isinstance( i , Teacher)

for i in l2:

s = isinstance(i , Student)

if(s == False and t == False):

return HttpResponseRedirect(reverse('register'))

else:

if(s):

#print(l2)

email = l2[0].student\_email

passw = l2[0].student\_password

else:

#print(l1)

email = l1[0].teacher\_email

passw = l1[0].teacher\_password

print(password)

print(passw)

print(pbkdf2\_sha256.verify(password,passw))

if (pbkdf2\_sha256.verify(password , passw)):

request .session['useremail'] = email

if(s):

return HttpResponseRedirect(reverse('studentlogin'))

else:

return HttpResponseRedirect(reverse('teacherlogin'))

#return HttpResponseRedirect(reverse('check'))

else:

return HttpResponseRedirect(reverse('register'))

def paperinfo(request):

cls = request.POST['cls']

divi = request.POST['divi']

subject = request.POST['sub']

qn1 = request.POST['qn1']

ans1 = request.POST['ans1']

qn2 = request.POST['qn2']

ans2 = request.POST['ans2']

qn3 = request.POST['qn3']

ans3 = request.POST['ans3']

s = Paperinfo(paper\_class = cls , paper\_division = divi , paper\_subject = subject , paper\_qn1 = qn1 , paper\_qn2 = qn2 , paper\_qn3 = qn3 , paper\_ans1 = ans1 , paper\_ans2 = ans2,paper\_ans3 = ans3)

s.save()

return HttpResponseRedirect(reverse('teacherlogin'))

def teacherlogout(request):

request.session.flush()

return HttpResponseRedirect(reverse('PortalHome'))

def answerinfo(request):

cls = request.POST['cls']

divi = request.POST['divi']

subject = request.POST['sub']

l1 = Paperinfo.objects.filter(paper\_class = cls , paper\_division = divi , paper\_subject = subject)

qn1 = l1[0].paper\_qn1

qn2 = l1[0].paper\_qn2

qn3 = l1[0].paper\_qn3

ans1 = l1[0].paper\_ans1

ans2 = l1[0].paper\_ans2

ans3 = l1[0].paper\_ans3

request.session['qn1'] = qn1

request.session['qn2'] = qn2

request.session['qn3'] = qn3

request.session['ans1'] = ans1

request.session['ans2'] = ans2

request.session['ans3'] = ans3

arg = {'cls':cls , 'divi':divi,'subject':subject,'qn1':qn1,'qn2':qn2,'qn3':qn3,'ans1':ans1,'ans2':ans2,'ans3':ans3}

return render(request , 'portal/student\_paper.html' , arg)

def answereval(request):

marklist = []

teacher\_answer = [request.session['ans1'],request.session['ans2'],request.session['ans3']]

student\_answer = [request.POST['ans1'],request.POST['ans2'],request.POST['ans3']]

'''

for i in teacher\_answer:

print(i)

for i in student\_answer:

print(i)

'''

'''

for i in range(0,3):

str1 = teacher\_answer[i]

str2 = student\_answer[i]

stop\_words = set(stopwords.words("english"))

##---------------Initialising Lists---------------##

filtered\_sentence1 = []

filtered\_sentence2 = []

lemm\_sentence1 = []

lemm\_sentence2 = []

sims = []

temp1 = []

temp2 = []

simi = []

final = []

same\_sent1 = []

same\_sent2 = []

# ps = PorterStemmer()

##---------------Defining WordNet Lematizer for English Language---------------##

lemmatizer = WordNetLemmatizer()

# myfile = open('Text1.txt', 'r')

# data=myfile.read().replace('\n', '')

##print(sent\_tokenize(example\_text))

##

##print(word\_tokenize(example\_text))

##---------------Tokenizing and removing the Stopwords---------------##

for words1 in word\_tokenize(str1):

if words1 not in stop\_words:

if words1.isalnum():

filtered\_sentence1.append(words1)

##---------------Lemmatizing: Root Words---------------##

for i in filtered\_sentence1:

lemm\_sentence1.append(lemmatizer.lemmatize(i))

# print(lemm\_sentence1)

##---------------Tokenizing and removing the Stopwords---------------##

for words2 in word\_tokenize(str2):

if words2 not in stop\_words:

if words2.isalnum():

filtered\_sentence2.append(words2)

##---------------Lemmatizing: Root Words---------------##

for i in filtered\_sentence2:

lemm\_sentence2.append(lemmatizer.lemmatize(i))

# print(lemm\_sentence2)

##---------------Removing the same words from the tokens----------------##

##for word1 in lemm\_sentence1:

## for word2 in lemm\_sentence2:

## if word1 == word2:

## same\_sent1.append(word1)

## same\_sent2.append(word2)

##

##if same\_sent1 != []:

## for word1 in same\_sent1:

## lemm\_sentence1.remove(word1)

##if same\_sent2 != []:

## for word2 in same\_sent2:

## lemm\_sentence2.remove(word2)

##

##print(lemm\_sentence1)

##print(lemm\_sentence2)

##---------------Similarity index calculation for each word---------------##

for word1 in lemm\_sentence1:

simi = []

for word2 in lemm\_sentence2:

sims = []

# print(word1)

# print(word2)

syns1 = wordnet.synsets(word1)

# print(syns1)

# print(wordFromList1[0])

syns2 = wordnet.synsets(word2)

# print(wordFromList2[0])

for sense1, sense2 in product(syns1, syns2):

d = wordnet.wup\_similarity(sense1, sense2)

if d != None:

sims.append(d)

# print(sims)

# print(max(sims))

if sims != []:

max\_sim = max(sims)

# print(max\_sim)

simi.append(max\_sim)

if simi != []:

max\_final = max(simi)

final.append(max\_final)

# print(final)

# if max\_sim >= 0.7:

# print(word1)

# print(word2)

# print('\n')

# if word1 not in temp1:

# temp1.append(word1)

# if word2 not in temp2:

# temp2.append(word2)

# lemm\_sentence1.remove(word1)

# lemm\_sentence2.remove(word2)

# if wordFromList1 and wordFromList2: #Thanks to @alexis' note

# s = wordFromList1[0].wup\_similarity(wordFromList2[0])

# list.append(s)

# for word1 in temp1:

# lemm\_sentence1.remove(word1)

# for word2 in temp2:

# lemm\_sentence2.remove(word2)

# print(lemm\_sentence1)

# print(lemm\_sentence2)

##---------------Final Output---------------##

similarity\_index = numpy.mean(final)

similarity\_index = round(similarity\_index, 2)

print("Sentence 1: ", str1)

print("Sentence 2: ", str2)

print("Similarity index value : ", similarity\_index)

mark = ""

if similarity\_index >= 0.8:

mark = "100/100"

elif similarity\_index >= 0.75 and similarity\_index < 0.80:

mark = "90/100"

elif similarity\_index >= 0.70 and similarity\_index < 0.75:

mark = "80/100"

elif similarity\_index >= 0.70 and similarity\_index < 0.60:

mark = "70/100"

else:

mark = "60/100"

marklist.append(mark)

arg = {'mark1':marklist[0] , 'mark2':marklist[1] , 'mark3':marklist[2] , 'qn1':request.session['qn1'],'qn2':request.session['qn2'],'qn3':request.session['qn3']}

'''

marklist = ["100/100" , "100/100" , "100/100"]

arg = {'mark1': marklist[0], 'mark2': marklist[1], 'mark3': marklist[2], 'qn1': request.session['qn1'],'qn2': request.session['qn2'], 'qn3': request.session['qn3']}

return render(request , 'portal/student\_result.html' , arg)

'''

def design(request):

all = Teacherinfo.objects.filter( teacher\_email = request.session['useremail'])

return render(request , ')

'''

Models.py

#contain database information

from django.db import models

# Create your models here.

class Teacher(models.Model):

teacher\_id = models.AutoField

teacher\_name = models.CharField(max\_length = 50)

teacher\_email = models.CharField(max\_length = 50)

teacher\_password = models.CharField(max\_length = 256 )

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

return self.teacher\_name

class Student(models.Model):

student\_id = models.AutoField

student\_name = models.CharField(max\_length = 50)

student\_email = models.CharField(max\_length = 50)

student\_password = models.CharField(max\_length = 256 )

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

return self.student\_name

class Paperinfo(models.Model):

paper\_id = models.AutoField

paper\_class = models.CharField(max\_length = 10)

paper\_division = models.CharField(max\_length = 10)

paper\_subject = models.CharField(max\_length = 50)

paper\_qn1 = models.CharField(max\_length = 200)

paper\_ans1 = models.TextField()

paper\_qn2 = models.CharField(max\_length=200)

paper\_ans2 = models.TextField()

paper\_qn3 = models.CharField(max\_length=200)

paper\_ans3 = models.TextField()

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

return self.paper\_subject