**SAMPLE CODE**

**Urls.py**

"""Taskontology URL Configuration

The `urlpatterns` list routes URLs to views. For more information please see:

https://docs.djangoproject.com/en/2.2/topics/http/urls/

Examples:

Function views

1. Add an import: from my\_app import views

2. Add a URL to urlpatterns: path('', views.home, name='home')

Class-based views

1. Add an import: from other\_app.views import Home

2. Add a URL to urlpatterns: path('', Home.as\_view(), name='home')

Including another URLconf

1. Import the include() function: from django.urls import include, path

2. Add a URL to urlpatterns: path('blog/', include('blog.urls'))

"""

from django.conf.urls import url

from django.contrib import admin

from django.urls import path

from django.conf import settings

from django.conf.urls.static import static

from Taskontology.views import adminlogin, adminloginaction, logout, userdetails, activateuser, userfiles, adminhome

from user.views import index, base, registration, user, userlogincheck, userhome, uploadfile, viewuserfiles, \

findvocabulary, detection, imagedetect

urlpatterns = [

url(r'^admin/', admin.site.urls),

url(r'^$', index, name="index"),

url(r'^index/', index, name="index"),

url(r'^base/', base, name="base"),

url(r'^registration/', registration, name="registration"),

url(r'^user/', user, name="user"),

url(r'^adminhome/', adminhome, name="adminhome"),

url(r'^adminlogin/', adminlogin, name="adminlogin"),

url(r'^adminloginaction/', adminloginaction, name="adminloginaction"),

url(r'^logout/', logout, name="logout"),

url(r'^userdetails/', userdetails, name="userdetails"),

url(r'^userfiles/', userfiles, name="userfiles"),

url(r'^activateuser/', activateuser, name="activateuser"),

url(r'^userlogincheck/', userlogincheck, name="userlogincheck"),

url(r'^userhome/', userhome, name="userhome"),

url(r'^uploadfile/', uploadfile, name="uploadfile"),

url(r'^viewuserfiles/', viewuserfiles, name="viewuserfiles"),

url(r'^findvocabulary/', findvocabulary, name="findvocabulary"),

url(r'^detection/', detection, name="detection"),

url(r'^imagedetect', imagedetect, name="imagedetect")

]

if settings.DEBUG:

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

Main **Views.py**

from random import randint

from django.shortcuts import render

from django.contrib import messages

from user.models import registrationmodel, uploadmodel

def adminhome(request):

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

def adminlogin(request):

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

def adminloginaction(request):

if request.method == "POST":

if request.method == "POST":

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

print(login)

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

if login == 'admin' and pswd == 'admin':

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

else:

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

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

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

def logout(request):

return render(request,'index.html')

def userdetails(request):

userdata = registrationmodel.objects.all()

return render(request,'admin/viewuserdetails.html', {'object': userdata})

def userfiles(request):

userfile = uploadmodel.objects.all()

return render(request, 'admin/viewfiles.html',{'object': userfile})

def activateuser(request):

if request.method=='GET':

usid = request.GET.get('usid')

authkey = random\_with\_N\_digits(8)

status = 'activated'

print("USID = ",usid,authkey,status)

registrationmodel.objects.filter(id=usid).update(authkey=authkey , status=status)

userdata = registrationmodel.objects.all()

return render(request,'admin/viewuserdetails.html',{'object':userdata})

def random\_with\_N\_digits(n):

range\_start = 10\*\*(n-1)

range\_end = (10\*\*n)-1

return randint(range\_start, range\_end)

User **views.py**

import nltk

from PIL import Image

from django.conf import settings

from django.shortcuts import render, HttpResponseRedirect, redirect

from django.contrib import messages

# Create your views here.

from nltk import word\_tokenize

from nltk.corpus import wordnet as wn

from user.forms import registrationform, UploadfileForm

from user.models import registrationmodel, uploadmodel

from msilib.schema import File

from nltk.corpus import wordnet

import numpy as np

import argparse

import time

import cv2

import os

def index(request):

return render(request,"index.html")

def base(request):

return render(request, "base.html")

def registration(request):

if request.method == 'POST':

form = registrationform(request.POST)

if form.is\_valid():

# print("Hai Meghana")

form.save()

messages.success(request, 'you are successfully registred')

return HttpResponseRedirect('trainer')

else:

print('Invalid')

else:

form = registrationform()

return render(request, "user/registration.html", {'form': form})

def userhome(request):

return render(request, "user/userhome.html")

def user(request):

return render(request, "user/user.html")

def userlogincheck(request):

if request.method == 'POST':

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

print(usid)

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

print(pswd)

try:

check = registrationmodel.objects.get(loginid=usid, password=pswd)

# print('usid',usid,'pswd',pswd)

request.session['userid'] = check.loginid

status = check.status

if status == "activated":

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

#auth.login(request, usid)

return render(request,'user/userpage.html')

else:

messages.success(request, 'user is not activated')

return render(request,'user/user.html')

except Exception as e:

print('Exception is ', str(e))

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

return render(request,'user/user.html')

def uploadfile(request):

if request.method == 'POST':

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

if form.is\_valid():

form.save()

return redirect('user/upload\_list.html')

else:

form = UploadfileForm()

return render(request, 'user/uploadfile.html', {'form': form})

def upload\_list(request):

files = File.objects.all()

return render(request, 'upload\_list.html', {'files': files})

def viewuserfiles(request):

filedata = uploadmodel.objects.all()

return render(request, 'user/viewuserdata.html', {'object': filedata})

def findvocabulary(request):

if request.method == "GET":

file = request.GET.get('id')

try:

#check = uploadmodel.objects.get(id=usid)

#file = check.file

print("Path is ", settings.MEDIA\_ROOT+'/'+file)

raw = open(settings.MEDIA\_ROOT+'/'+file).read()

#print(raw)

tokens = word\_tokenize(raw)

#print(tokens)

words = set(w.lower() for w in nltk.corpus.words.words())

# tokens1 = word\_tokenize(words)

tokens1 = list(words)

# print(tokens1)

voc = set(tokens) & set(tokens1)

meg = str(voc)

#print('Word type ',meg)

word = meg.split(",")

for x in word:

#print('X = ',x)

line = nltk.re.sub('[^ a-zA-Z0-9]', '', x)

#print("Line ", line)

for x in line:

sysns = wn.synsets(x)

#print('Rslt ',sysns)

#texts = [[word.lower() for word in text.split()] for text in voc]

#syns = wn.synsets(meg)

#print("synsets:", syns)

dict = {

"file": file,

"voc": voc,

"sysns": sysns,

}

#print(dict)

katti = {}

vcData = dict['voc']

#print(vcData)

try:

for xword in vcData:

#print('for NLTK =',xword)

syn = wordnet.synsets(xword)

if len(syn) !=0:

description = syn[0].definition()

katti.update({xword:description})

else:

pass

except Exception as e:

print(e)

pass

#print('katti type ',katti)

dict1 = {

"katti": katti,

'dict':dict

}

#print("dict1:",dict1)

#return render(request, "user/vocabulary.html",{'dict':dict,'katti':katti})

#return render(request, "user/vocabulary.html", katti)

return render(request, "user/vocabulary.html", dict1)

except Exception as e:

print('Exception is ', str(e))

messages.success(request, 'Invalid Details')

return render(request, 'user/viewuserdata.html')

def detection(request):

if request.method == 'POST':

images = request.FILES.get('imgfile')

print("image:",images)

img = Image.open(images)

#print("meghana:", img)

image = img.save(settings.MEDIA\_ROOT + "/cropped\_picture.jpg")

args = {'yolo': 'yolo-coco', 'confidence': 0.5, 'threshold': 0.3} # vars(ap.parse\_args())

print("Volvorine Args ", type(args))

args.update({'image': image})

print("Dict Data ", args)

# load the COCO class labels our YOLO model was trained on

labelsPath = os.path.sep.join(["yolo-coco/coco.names"])

LABELS = open(labelsPath).read().strip().split("\n")

# initialize a list of colors to represent each possible class label

np.random.seed(42)

COLORS = np.random.randint(0, 255, size=(len(LABELS), 3), dtype="uint8")

# derive the paths to the YOLO weights and model configuration

weightsPath = os.path.sep.join(["yolo-coco/yolov3.weights"])

configPath = os.path.sep.join(["yolo-coco/yolov3.cfg"])

# load our YOLO object detector trained on COCO dataset (80 classes)

print("[INFO] loading YOLO from disk...")

net = cv2.dnn.readNetFromDarknet(configPath, weightsPath)

# load our input image and grab its spatial dimensions

# image = "F:/Python/Alex Codes/yolo-object-detection/images/soccer.jpg"

# image = cv2.imread(args["image"])

image = cv2.imread(settings.MEDIA\_ROOT+"/cropped\_picture.jpg")

print("images:",image)

(H, W) = image.shape[:2]

# determine only the \*output\* layer names that we need from YOLO

ln = net.getLayerNames()

ln = [ln[i[0] - 1] for i in net.getUnconnectedOutLayers()]

# construct a blob from the input image and then perform a forward

# pass of the YOLO object detector, giving us our bounding boxes and

# associated probabilities

blob = cv2.dnn.blobFromImage(image, 1 / 255.0, (416, 416),

swapRB=True, crop=False)

net.setInput(blob)

start = time.time()

layerOutputs = net.forward(ln)

end = time.time()

# show timing information on YOLO

print("[INFO] YOLO took {:.6f} seconds".format(end - start))

# initialize our lists of detected bounding boxes, confidences, and

# class IDs, respectively

boxes = []

confidences = []

classIDs = []

# loop over each of the layer outputs

for output in layerOutputs:

# loop over each of the detections

for detection in output:

# extract the class ID and confidence (i.e., probability) of

# the current object detection

scores = detection[5:]

classID = np.argmax(scores)

confidence = scores[classID]

# filter out weak predictions by ensuring the detected

# probability is greater than the minimum probability

if confidence > args["confidence"]:

# scale the bounding box coordinates back relative to the

# size of the image, keeping in mind that YOLO actually

# returns the center (x, y)-coordinates of the bounding

# box followed by the boxes' width and height

box = detection[0:4] \* np.array([W, H, W, H])

(centerX, centerY, width, height) = box.astype("int")

# use the center (x, y)-coordinates to derive the top and

# and left corner of the bounding box

x = int(centerX - (width / 2))

y = int(centerY - (height / 2))

# update our list of bounding box coordinates, confidences,

# and class IDs

boxes.append([x, y, int(width), int(height)])

confidences.append(float(confidence))

classIDs.append(classID)

# apply non-maxima suppression to suppress weak, overlapping bounding

# boxes

idxs = cv2.dnn.NMSBoxes(boxes, confidences, args["confidence"],

args["threshold"])

# ensure at least one detection exists

if len(idxs) > 0:

# loop over the indexes we are keeping

for i in idxs.flatten():

# extract the bounding box coordinates

(x, y) = (boxes[i][0], boxes[i][1])

(w, h) = (boxes[i][2], boxes[i][3])

# draw a bounding box rectangle and label on the image

color = [int(c) for c in COLORS[classIDs[i]]]

cv2.rectangle(image, (x, y), (x + w, y + h), color, 2)

text = "{}: {:.4f}".format(LABELS[classIDs[i]], confidences[i])

cv2.putText(image, text, (x, y - 5), cv2.FONT\_HERSHEY\_SIMPLEX,

0.5, color, 2)

# show the output image

cv2.imshow("Image", image)

cv2.waitKey(0)

return render(request,"user/objectdetect.html")

def imagedetect(request):

return render(request, "user/imagedetect.html")

User **forms.py**

from django import forms

from user.models import registrationmodel, uploadmodel

class registrationform(forms.ModelForm):

loginid = forms.CharField(widget=forms.TextInput(), required=True, max\_length=100)

password = forms.CharField(widget=forms.PasswordInput(), required=True, max\_length=100)

email = forms.EmailField(widget=forms.TextInput(),required=True)

mobile = forms.CharField(widget=forms.TextInput(),required=True,max\_length=100)

place = forms.CharField(widget=forms.TextInput(),required=True,max\_length=100)

city = forms.CharField(widget=forms.TextInput(),required=True,max\_length=100)

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

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

class Meta:

model = registrationmodel

fields = ['loginid','password','email','mobile','place','city','authkey','status' ]

class UploadfileForm(forms.ModelForm):

class Meta:

model = uploadmodel

fields = ('filename','file')

user **Models.py**

from django.db import models

# Create your models here.

class registrationmodel(models.Model):

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

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

email = models.EmailField()

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

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

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

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

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

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

return self.email

class uploadmodel(models.Model):

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

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

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

return self.filename

**Adminbase.html**

{% load static %}

<!DOCTYPE html>

<html lang="en">

<head>

<title>Task Ontology</title>

<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

<link href="https://fonts.googleapis.com/css?family=Muli:300,400,700,900" rel="stylesheet">

<link rel="stylesheet" href="{% static 'fonts/icomoon/style.css' %}">

<link rel="stylesheet" href="{% static 'css/bootstrap.min.css' %}">

<link rel="stylesheet" href="{% static 'css/jquery-ui.css' %}">

<link rel="stylesheet" href="{% static 'css/owl.carousel.min.css' %}">

<link rel="stylesheet" href="{% static 'css/owl.theme.default.min.css' %}">

<link rel="stylesheet" href="{% static 'css/owl.theme.default.min.css' %}">

<link rel="stylesheet" href="{% static 'css/jquery.fancybox.min.css' %}">

<link rel="stylesheet" href="{% static 'css/bootstrap-datepicker.css' %}">

<link rel="stylesheet" href="{% static 'fonts/flaticon/font/flaticon.css' %}">

<link rel="stylesheet" href="{% static 'css/aos.css' %}">

<link rel="stylesheet" href="{% static 'css/style.css'%}">

</head>

<body data-spy="scroll" data-target=".site-navbar-target" data-offset="300">

<div class="site-wrap">

<div class="site-mobile-menu site-navbar-target">

<div class="site-mobile-menu-header">

<div class="site-mobile-menu-close mt-3">

<span class="icon-close2 js-menu-toggle"></span>

</div>

</div>

<div class="site-mobile-menu-body"></div>

</div>

<header class="site-navbar py-4 js-sticky-header site-navbar-target" role="banner">

<div class="container-fluid">

<div class="d-flex align-items-center">

<div class="site-logo mr-auto w-25"><a href="{% url 'index' %}">Task Ontologys</a></div>

<div class="mx-auto text-center">

<nav class="site-navigation position-relative text-right" role="navigation">

<ul class="site-menu main-menu js-clone-nav mx-auto d-none d-lg-block m-0 p-0">

<li><a href="{% url 'adminhome' %}" class="nav-link">AdminHome</a></li>

<li><a href="{% url 'userdetails' %}" class="nav-link">UserDetails</a></li>

<li><a href="{% url 'userfiles' %}" class="nav-link">Viewfiles</a></li>

</ul>

</nav>

</div>

<div class="ml-auto w-25">

<nav class="site-navigation position-relative text-right" role="navigation">

<ul class="site-menu main-menu site-menu-dark js-clone-nav mr-auto d-none d-lg-block m-0 p-0">

<li class="cta"><a href="{% url 'logout' %}" class="nav-link"><span>Logout</span></a></li>

</ul>

</nav>

<a href="#" class="d-inline-block d-lg-none site-menu-toggle js-menu-toggle text-black float-right"><span class="icon-menu h3"></span></a>

</div>

</div>

</div>

</header>

<!--<div class="intro-section" id="home-section">

<div class="slide-1" style="background-image: url({% static 'images/hero\_1.jpg' %});" data-stellar-background-ratio="0.5">

<div class="container">

<div class="row align-items-center">

<div class="col-12">

<div class="row align-items-center">

<div class="col-lg-6 mb-4">

<h1 data-aos="fade-up" data-aos-delay="100">Autonomous Machine Learning Modeling using a Task Ontology</h1>

<!--<p class="mb-4" data-aos="fade-up" data-aos-delay="200">Lorem ipsum dolor sit amet consectetur adipisicing elit. Maxime ipsa nulla sed quis rerum amet natus quas necessitatibus.</p>

<p data-aos="fade-up" data-aos-delay="300"><a href="#" class="btn btn-primary py-3 px-5 btn-pill">Admission Now</a></p>

</div>-->

<!--<div class="col-lg-5 ml-auto" data-aos="fade-up" data-aos-delay="500">

<form action="" method="post" class="form-box">

<h3 class="h4 text-black mb-4">Sign Up</h3>

<div class="form-group">

<input type="text" class="form-control" placeholder="Email Addresss">

</div>

<div class="form-group">

<input type="password" class="form-control" placeholder="Password">

</div>

<div class="form-group mb-4">

<input type="password" class="form-control" placeholder="Re-type Password">

</div>

<div class="form-group">

<input type="submit" class="btn btn-primary btn-pill" value="Sign up">

</div>

</form>

</div>-->

</div>

</div>

</div>

</div>

</div>

</div>

{% block contents %}

{% endblock %}

<!--<footer >

<div class="row pt-5 mt-5 text-center">

<div class="border-top pt-5">

<p>

Link back to Colorlib can't be removed. Template is licensed under CC BY 3.0.

Copyright &copy;<script>document.write(new Date().getFullYear());</script> All rights reserved

<!-- Link back to Colorlib can't be removed. Template is licensed under CC BY 3.0.

</p>

</div>

</div>

</div>

</div>

</footer>-->

</div> <!-- .site-wrap -->

<script src="{% static 'js/jquery-3.3.1.min.js' %}"></script>

<script src="{% static 'js/jquery-migrate-3.0.1.min.js' %}"></script>

<script src="{% static 'js/jquery-ui.js' %}"></script>

<script src="{% static 'js/popper.min.js' %}"></script>

<script src="{% static 'js/bootstrap.min.js' %}"></script>

<script src="{% static 'js/owl.carousel.min.js' %}"></script>

<script src="{% static 'js/jquery.stellar.min.js"></script>

<script src="{% static 'js/jquery.countdown.min.js' %}"></script>

<script src="{% static 'js/bootstrap-datepicker.min.js' %}"></script>

<script src="{% static 'js/jquery.easing.1.3.js' %}"></script>

<script src="{% static 'js/aos.js' %}"></script>

<script src="{% static 'js/jquery.fancybox.min.js' %}"></script>

<script src="{% static 'js/jquery.sticky.js' %}"></script>

<script src="{% static 'js/main.js' %}"></script>

</body>

</html>

**Vocabulary.html**

{% extends 'userbase.html'%}

{% load static %}

{% block contents %}

<div class="intro-section" id="home-section">

<div class="slide-1" style="background-image: url({% static 'images/hero\_1.jpg' %});" data-stellar-background-ratio="0.5">

<div class="container">

<div class="row align-items-center">

<div class="col-12">

<div class="row align-items-center">

<div class="col-lg-6 mb-4">

<h1 data-aos="fade-up" data-aos-delay="100">&nbsp;</h1>

<!--<p class="mb-4" data-aos="fade-up" data-aos-delay="200">Lorem ipsum dolor sit amet consectetur adipisicing elit. Maxime ipsa nulla sed quis rerum amet natus quas necessitatibus.</p>

<p data-aos="fade-up" data-aos-delay="300"><a href="#" class="btn btn-primary py-3 px-5 btn-pill">Admission Now</a></p>-->

<table border="2px">

<tr style="color:RED"><th>S.No</th><th>Word</th><th>Defination</th></tr>

{% for key,value in katti.items%}

<tr style="color:orange">

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

<td>{{key}}</td>

<td>{{value}}</td>

</tr>

{% endfor %}

</table>

</div>

<div class="col-lg-5 ml-auto" data-aos="fade-up" data-aos-delay="500">

<form action="" method="POST" class="form-box">

<h3 class="h4 text-black mb-4">Vocabulary Words</h3>

<table >

Results = {{dict}}

</table>

</form>

</div>

{% endblock %}