SOURCE CODE

from pathlib import Path

import os

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

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

INSTALLED\_APPS = [

    'django.contrib.admin',

    'django.contrib.auth',

    'django.contrib.contenttypes',

    'django.contrib.sessions',

    'django.contrib.messages',

    'django.contrib.staticfiles',

    'price'

]

ROOT\_URLCONF = 'gold\_price.urls'

TEMPLATES = [

    {

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

        'DIRS': [os.path.join(BASE\_DIR,'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 = 'gold\_price.wsgi.application'

# Database

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

DATABASES = {

    'default': {

        'ENGINE': 'django.db.backends.sqlite3',

        'NAME': BASE\_DIR / 'db.sqlite3',

    }

}

# 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,'source')

]

# 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 sending

EMAIL\_USE\_TLS = True

EMAIL\_HOST = 'smtp.gmail.com'

EMAIL\_PORT = 587

EMAIL\_HOST\_USER = 'example@gmail.com'

EMAIL\_HOST\_PASSWORD = 'password'

from django.contrib import admin

from django.urls import path,include

urlpatterns = [

    path('admin/', admin.site.urls),

    path('', include('price.urls'))

]

from django.shortcuts import render

from django.http import HttpResponse

from . models import price\_input, price\_prediction

from django.contrib.sessions.models import Session

from django.core.mail import send\_mail

from django.conf import settings

def mail(rate,eml):

    subject = "Gold Price"

    msg = " predict Gold price is greater than this shop rate " +str(rate)

    to=eml

    res=send\_mail(subject, msg,settings.EMAIL\_HOST\_USER, [to])

    if(res == 1):

        msg = "Mail Sent Successfuly"

    else:

        msg = "Mail could not sent"

    return msg

# Create your views here. https://www.goodreturns.in/silver-rates/

def shoprate():

    price=[]

    # Required module- pip install requests

    import requests

    # pip install Bs4

    from bs4 import BeautifulSoup

    # Passing Request

    # page = requests.get(

    #     "https://www.bhimagold.com/gold?utm\_source=google+search+&utm\_medium=brand&utm\_campaign=indxnl&gclid=Cj0KCQjwm6KUBhC3ARIsACIwxBgW31Yohxyrb2p01\_YdGJUPiaBP6fzbZtchm6L4kKEyFa4uOE9IRNEaApd3EALw\_wcB")

    url = 'https://www.bhimagold.com/'

    headers = { 'User-Agent' :

'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/101.0.4951.67 Safari/537.36'}

    page = requests.get(url,headers=headers)

    # Getting Content of Page

    Soup = BeautifulSoup(page.content, 'html.parser')

    # Getting info specific Class

    info = Soup.find\_all(class\_='gold-label-price')

    count = 0

    # Getting Date Which we Required...

    for items in info:

        # count += 1

        # if count==1:

        price.append(items.get\_text())

        pr=items.get\_text()

        return pr

def sliver():

    price=[]

    # Required module- pip install requests

    import requests

    # pip install Bs4

    from bs4 import BeautifulSoup

    # Passing Request

    page = requests.get(

        "https://www.moneycontrol.com/commodity/silver-price.html")

    # Getting Content of Page

    Soup = BeautifulSoup(page.content, 'html.parser')

    # Getting info specific Class

    info = Soup.find\_all(class\_='rd\_30')

    count = 0

    # Getting Date Which we Required...

    for items in info:

        count += 1

        if count==1:

            price.append(items.get\_text())

            print(items.get\_text())

            val=items.get\_text()

            val=val.split(" ")

            break

    for p in price:

        one=p

        return p

def spx():

    price=[]

    # Required module- pip install requests

    import requests

    # pip install Bs4

    from bs4 import BeautifulSoup

    # Passing Request

    page = requests.get(

        "https://www.google.com/finance/quote/.INX:INDEXSP?sa=X&ved=2ahUKEwjfk5mpyt33AhWQUGwGHRA5ALoQ3ecFegQIJxAY")

    # Getting Content of Page

    Soup = BeautifulSoup(page.content, 'html.parser')

    # Getting info specific Class

    info = Soup.find\_all(class\_='YMlKec')

    count = 0

    # Getting Date Which we Required...

    for items in info:

        count += 1

        if count==1:

            price.append(items.get\_text())

            print(items.get\_text())

            val=items.get\_text()

            val=val.split(" ")

            break

    for p in price:

        one=p

    one=one.replace(',',"")

    return one

def goldprice():

    price=[]

    # Required module- pip install requests

    import requests

    # pip install Bs4

    from bs4 import BeautifulSoup

    # Passing Request

    page = requests.get(

        "https://markets.businessinsider.com/commodities/gold-price")

    # Getting Content of Page

    Soup = BeautifulSoup(page.content, 'html.parser')

    # Getting info specific Class

    info = Soup.find\_all(class\_='price-section\_\_current-value')

    count = 0

    # Getting Date Which we Required...

    for items in info:

count += 1

        if count==1:

            price.append(items.get\_text())

            # print(items.get\_text())

            # val=items.get\_text()

            # val=val.split(" ")

            break

    for p in price:

        one=p

    return one

def usdoller():

    price=[]

    # Required module- pip install requests

    import requests

    # pip install Bs4

    from bs4 import BeautifulSoup

    # Passing Request

    page = requests.get(

        "https://www.google.com/finance/quote/USD-RUB")

    # Getting Content of Page

    Soup = BeautifulSoup(page.content, 'html.parser')

    # Getting info specific Class

    info = Soup.find\_all(class\_='fxKbKc')

    count = 0

    # Getting Date Which we Required...

    for items in info:

        count += 1

        if count==1:

            price.append(items.get\_text())

            print(items.get\_text())

            # val=items.get\_text()

            # val=val.split(" "

            break

    for p in price:

        one=p

        return one

#euro

def euro():

    price=[]

    # Required module- pip install request

    import requests

    # pip install Bs4

  from bs4 import BeautifulSoup

    # Passing Request

    page = requests.get(

        "https://www.google.com/finance/quote/EUR-USD")

    # Getting Content of Page

    Soup = BeautifulSoup(page.content, 'html.parser')

    # Getting info specific Class

    info = Soup.find\_all(class\_='YMlKec fxKbKc')

    count = 0

    # Getting Date Which we Required...

    for items in info:

        count += 1

        if count==1:

    price.append(items.get\_text())

            # print(items.get\_text())

            # val=items.get\_text()

            # val=val.split(" ")

            break

    for p in price:

        one=p

        return one

def gold(request):

    rate=shoprate()

    return render(request,'index.html',{'goldrate':rate})

def user\_reg(request):

    if request.method=="POST":

        nam=request.POST.get("nam")

        eml=request.POST.get("eml")

        cnum=request.POST.get("cnum")

        pword=request.POST.get("paswd")

        usernam=request.POST.get("usrnam")

        obj=price\_prediction.objects.create(nam=nam,eml=eml,cnum=cnum,password=pword,uname=usernam)

        obj.save()

        return HttpResponse("<h4> Successfully registered</h4> <a href='/'>Login </a> ")

    return render(request,'registration.html')

def home(request):

    user=request.session['username']

    passw=request.session['password']

    obj=price\_prediction.objects.filter(uname=user,password=passw)

    if obj:

        for l in obj:

            idno=l.id

            eml=l.eml

        request.session['idn']=idno

        request.session['eml']=eml

        gld=goldprice()

        sp=spx()

        slv=sliver()

        ero=euro()

        us=usdoller()

        return render(request,'home.html',{'user':obj,'gl':gld,'spx':sp,'slv':slv,'ero':ero,'us':us})

def login(request):

    if request.method=="POST":

        user=request.POST.get("user")

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

        obj=price\_prediction.objects.filter(uname=user,password=passw)

        if obj:

            for l in obj:

                idno=l.id

                eml=l.eml

            request.session['idn']=idno

            request.session['eml']=eml

            request.session['username']=user

            request.session['password']=passw

            gld=goldprice()

            sp=spx()

            slv=sliver()

            ero=euro()

            us=usdoller()

            request.session['gld']=gld

            request.session['sp']=sp

            request.session['slv']=slv

            request.session['ero']=ero

            request.session['us']=us

            return render(request,'home.html',{'user':obj,'gl':gld,'spx':sp,'slv':slv,'ero':ero,'us':us})

        else:

            return render(request,'index.html',{'msg':'invalid username and password'})

    else:

        return render(request,'index.html')

def predict(data\_list):

    import joblib

    import numpy

    # import matplotlib.pyplot as plt

    # import pandas as pd

    # import seaborn as sms

    model2=joblib.load("model\goldprice\_md.joblib")

    prf=model2.predict([data\_list])

    print(prf[0])

    return prf[0]

def prediction(request):

     if request.method=="POST":

        spx=float(request.POST.get("spx"))

        gld=float(request.POST.get("gld"))

        usd=float(request.POST.get("usd"))

        slv=float(request.POST.get("slv"))

        eur=float(request.POST.get("eur"))

        datalist=[spx,usd,slv,eur]

        prd=predict(datalist)

        obj=price\_input.objects.create(spx=spx,gld=gld,usd=usd, slv=slv, eur=eur,predict=prd)

        obj.save()

        # return HttpResponse("<div style='width:40%; height:300px; background-color:yellow;'> <h3 style='text-algin:center;'>Gold Price Prediction:"+str(prd)+"</h3>  </div>")

        user=request.session['username']

        passw=request.session['username']

        obj=price\_prediction.objects.filter(uname=user,password=passw)

        if obj:

            for l in obj:

                idno=l.id

            request.session['idn']=idno

        gld=request.session['gld']

        sp=request.session['sp']

        slv=request.session['slv']

        ero=request.session['ero']

        us=request.session['us']

        shopr=shoprate()

        # if int(shopr)<prd

        return render(request,'home.html',{'user':obj,'gl':gld,'spx':sp,'slv':slv,'ero':ero,'us':us,'prd':prd})

     return render(request,'registration.html')

def logout(request):

    request.session['username']=" "

    request.session['passsword']="

    return Http Response("Lout out <br> <a href='/login'>Login again")

def gp (request):

    obj =price\_prediction.objects.all()

    return render(request,'gold\_reg.html',{'data':obj}) from django.urls

import path, include

from .import views

urlpatterns = [

    path('',views.gold),

    path('registration',views.user\_reg),

    path('gold\_reg',views.gp),

    path('login',views.login),

    path('prediction',views.prediction),

    path('home',views.home),

    path('logout',views.logout)

    ]

mt-5">Sizes :</h6>

      <button class="btn btn-primary btn-lg">Large</button>

      <button class="btn btn-primary ">Regular</button>

      <button class="btn btn-primary btn-sm">Small</button>

      <button class="btn btn-primary w-sm ml-3">w-sm</button>

      <button class="btn btn-primary w-md">w-md</button>

      <button class="btn btn-primary w-lg">w-lg</button>

      <h6 class="section-secondary-title mt-5">Colors :</h6>

      <button type="button" class="btn btn-primary">Primary</button>

      <button type="button" class="btn btn-secondary">Secondary</button>

      <button type="button" class="btn btn-success">Success</button>

      <button type="button" class="btn btn-danger">Danger</button>

      <button type="button" class="btn btn-warning">Warning</button>

      <button type="button" class="btn btn-info">Info</button>

      <button type="button" class="btn btn-light">Light</button>

      <button type="button" class="btn btn-dark">Dark</button>

      <h6 class="section-secondary-title mt-5">Buttons Tags :</h6>

      <button class="btn btn-primary" type="submit">Button</button>

      <input class="btn btn-primary" type="button" value="Input">

      <input class="btn btn-primary" type="submit" value="Submit">

      <input class="btn btn-primary" type="reset" value="Reset">

      <!-- devider -->

      <div class="py-5"></div>

      <!-- Inputs -->

      <h3>Inputs</h3>

      <hr>

      <div class="row mt-5">

         <div class="col-md-6">

            <h6 class="section-secondary-title">Form controls :</h6>

            <div class="form-group">

               <input type="email" class="form-control" id="exampleFormControlInput1" placeholder="Email address">

            </div>

            <div class="form-group">

               <input type="date" class="form-control" id="exampleFormControlInput1" placeholder="Date">

            </div>

            <div class="form-group">

               <input class="form-control" type="text" placeholder="Readonly input here…" readonly>

            </div>

            <div class="form-group">

               <input class="form-control" id="disabledInput" type="text" placeholder="Disabled input here..." disabled>

            </div>

            <div class="form-group">

               <select class="form-control" id="exampleFormControlSelect1">

                  <option>Example select</option>

                  <option>1</option>

                  <option>2</option>

                  <option>3</option>

                  <option>4</option>

                  <option>5</option>

               </select>

            </div>

            <div class="form-group">

               <select multiple class="form-control" id="exampleFormControlSelect2">

                  <option>Example multiple select</option>

                  <option>1</option>

                  <option>2</option>

                  <option>3</option>

                  <option>4</option>

                  <option>5</option>

               </select>

            </div>

from django.db import models

# Create your models here.

class price\_prediction(models.Model):

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

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

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

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

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

    def \_\_str\_\_(self):

        return self.nam

class price\_input(models.Model):

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

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

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

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

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

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

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta http-equiv="X-UA-Compatible" content="IE=edge">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

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

    <title>Document</title>

</head>

<body>

    <a href="/login">Login again</a>

  <script>

    $(document).ready(function() {

        function disableBack() { window.history.forward() }

        window.onload = disableBack();

        window.onpageshow = function(evt) { if (evt.persisted) disableBack() }

    });

 </script>

</body>

</html>

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta http-equiv="X-UA-Compatible" content="IE=edge">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

</body>

</html>